SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6896

POPRAVLJANJE DJELOMIČNO SASTAVLJENOGA GENOMA POMOĆI HI-C OČITANJA

Filip Wolf

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6896

POPRAVLJANJE DJELOMIČNO SASTAVLJENOGA GENOMA POMOĆI HI-C OČITANJA

Filip Wolf

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6896

Pristupnik:	**Filip Wolf (0036510053)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	prof. dr. sc. Mile Šikić

Zadatak: **Popravljanje djelomično sastavljenoga genoma pomoći Hi-C očitanja**

Opis zadatka:

Današnji uređaji za očitanje DNA ili RNA nisu u stanju očitati cijele genome, već samo manje dijelove i pri tome unose određenu količinu pogreške. Jedan od osnovnih zadataka u bioinformatici jest sastaviti cijeli genom na temelju takvih očitanja. Alati za sastavljanje genoma većinom nisu u stanju u potpunosti sastaviti veće genome, već samo manje dijelove istih što rezultira fragmentiranim sastavljenim genomima. Za poboljšanje takvih djelomično sastavljenog genoma koriste se dodatne metode, koje koriste dodatne podatke poput Hi-C metode za očitanje kromatinskih interakcija. Potrebno je implementirati alat, koji će iskoristiti Hi-C podatke da bi djelomično sastavljene dijelove genoma grupirao u kromosome i odredio njihov međusobni raspored unutar pojedinog kromosoma. Alat testirati na stvarnim i simuliranim podacima. Rješenje mora biti napisano koristeći programski jezik C++. Programski kod je potrebno komentirati i pri pisanju pratiti neki od standardnih stilova. Napisati iscrpne upute za instalaciju i izvođenje. Kompletno programsko rješenje postaviti na GitHub pod jednom od OSI-odobrenih licenci.

Rok za predaju rada: 12. lipnja 2020.

Sadržaj

Uvod 1

1. Priprema podataka 3

1.1. O podatcima 3

1.2. Načini pripreme podataka 3

1.2.1. Vinska mušica (*Drosophila melanogaster*) 4

1.2.2. Čovjek (*Homo sapiens*) 6

1.3. Analiza ispravnosti podataka 6

1.4. Simuliranje kontiga 8

2. Algoritam 10

2.1. Općenito o postupku 10

2.2. Razvrstavanje u kromosome 10

2.3. Poredak kontiga unutar kromosoma 13

2.4. Određivanje orijentacije kontiga 15

2.5. Ograničenja i moguće nadogradnje 16

2.5.1. Ograničenja 16

2.5.2. Nadogradnje 17

3. Rezultati 19

3.1. Vinska mušica (*Drosophila melanogaster*) 19

3.1.1. Podatkovni skup 1 19

3.1.2. Podatkovni skup 2 20

3.2. Čovjek (*Homo sapiens*) 21

3.2.1. Podatkovni skup 1 21

3.2.2. Podatkovni skup 2 21

3.2.3. Podatkovni skup 3 22

Zaključak 23

Literatura 24

Sažetak 25

Summary 26

Uvod

Projekt Ljudskog Genoma [1] je početkom prošlog desetljeća postavio visoki standard za točnost sekvenciranja genoma. Tehnologije koje izvode takvo masovno sekvenciranje su danas sve jeftinije i dopuštaju primjenu na raznim eksperimentima. Iako su algoritmi za sekvenciranje znatno unaprijeđeni kroz godine, u svakodnevnom radu smo i dalje iznenađujuće daleko od standarda Projekta Ljudskog Genoma. Za Projekt Ljudskog Genoma su uložene milijarde dolara i godine razvoja i ne možemo se nadati dobiti takve rezultate sa djelićem uloženog, no cilj je približiti im se.

Tehnologije koje se danas koriste veće genome ne sastavlja do kraja nego nam daju fragmentirani konačan proizvod sastavljen od tisuća manjih dijelova koje nazivamo kontizi. Te metode su bazirane na kemijskim procesima i dolaze sa određenom greškom koja je mala, no i dalje dovoljno velika da se mora uzeti u obzir. Kada bismo pokušali tim metodama sekvencirati veće genome bi greška također bila prisutna, ali suočavamo se s dodatnim problemom da je broj ponavljajućih dijelova u njima mnogo veći, što alatima za sastavljanje predstavlja problem. Metode za sekvenciranje većih genoma postoje, no one rade sa već unaprijed izrezanim genomima koje je moguće onda pročitati i stoga je konačni proizvod niz kontiga kod kojih je prisutna mala greška, ali i dalje nisu sastavljeni u potpun genom. Određen standard potpunog sastavljanja smo uspjeli dostići samo kod bakterija i sličnih organizama s manjim genomom. Biolozima su informacije koje pruža djelomično sastavljeni genom korisne, no u nekim primjenama je potrebno imati potpuno sastavljeni genom i potpuno sastavljene kromosome. Stoga način sastavljanja potpunog genoma moramo pronaći nekim drugim metodama koje nisu samo bazirane na navedenom.

Ljudski genom je sekvenciran pomoću mnogo pomoćnih podataka koji ga čine vrlo točnim, ali i vrlo izvedbeno skupim. Razvijene su razne strategije za sastavljanje kontiga koje su često bazirane na pomoćnim podatcima unutar DNK. Metoda koju ćemo mi razmatrati se bazira na Hi-C podatcima. Hi-C podatci proizlaze iz trodimenzionalne strukture kromosoma unutar DNK i rasporeda kromatinskih veza unutar njih. Naime, gustoća kromatinskih veza je puno gušća unutar samih kromosoma što nam dopušta razvrstavanje kontiga u kromosome. Nadalje, gustoće kromatinskih veza su mnogo gušće među bližim dijelovima kromosoma nego među onima više udaljenim jedni od drugih, što nam dopušta lakše

razvrstavanje kontiga. Nakon što Hi-C podatke poravnamo sa sekvencama, možemo iščitati poveznice iz podataka i pomoću njih, odrediti gustoće poveznica između kontiga. Sa tim znanjem je razvijen algoritam sastavljanja kontiga u potpune kromosome s visokom točnošću. U poglavljima koja slijede je detaljno opisan navedeni postupak, kao i način pripreme podataka za testiranje. Ovaj rad je baziran na prijašnjem radu [2].

1. Priprema podataka

U ovom poglavlju će biti pobliže pojašnjeno koji su podatci korišteni, u kojem formatu su korišteni i kako su pripremljeni za obradu. Opisuju se skripte pisane u Perlu, naredbe komandne linije (bash) i vizualizacija u Pythonu.

1.1. O podatcima

Postoji mnoštvo metoda za sekvenciranje gena, pa tako postoji i mnogo različitih formata podataka. Pri izradi programa su korištene dvije vrste podataka: podatci o genskim sekvencama i podatci o Hi-C kromatinskim interakcijama. Za podatke o sekvencama je korišten široko korišteni FASTA format, dok je za Hi-C podatke korišten njemu komplementarni FASTQ format. Genske sekvence vinske mušice i njeni Hi-C podatci, kao i kromosomi čovjeka su preuzeti sa stranica NCBI-a [3], dok su Hi-C podatci čovjeka uručeni od strane mentora.

Hi-C podatci obično dolaze u dvije datoteke jednake veličine. Razlog tome je što su Hi-C podatci pročitani s dva različita dijela genoma i oni nam predstavljaju poveznicu između tih dijelova. Važno je i napomenuti da su poveznice unutar kromosoma gušće nego između različitih kromosomima i da su bliski dijelovi kromosoma gušće povezani od udaljenijih. Također je bitno obratiti pozornost na slijedeće: da bismo imali potpunu sliku svih poveznica unutar molekule DNK, moramo imati informaciju o svim Hi-C poveznicama unutar nje. Neke Hi-C poveznice će se poravnavati sa originalnom genskom sekvencom, dok će se neke druge Hi-C poveznice poravnavati sa reverznim komplementom te iste genske sekvence (reverzni komplement je u ovom slučaju druga zavojnica molekule DNK). Mi ćemo uzeti u obzir sve poveznice, neovisno o njihovoj orijentaciji te će se pri samom poravnanju na svaku gensku sekvencu pokušati vršiti poravnanje u oba smjera. Postoji i mogućnost zapisivanja rezultata u dvije odvojene datoteke, svaka za jednu Hi-C datoteku, ali ovdje to nije razmatrano.

1.2. Načini pripreme podataka

Pripremu podataka radimo na način da prvo Hi-C podatke poravnamo sa željenim genskim sekvencama, koje su u obliku jednog ili više kromosoma izrezanih na kontige, i rezultat

obradimo tako da izbacimo neporavnata Hi-C očitanja i neispravne parove podataka. To radimo sa namjerom da iz rezultantnih podataka o Hi-C poravnanjima sa genskom sekvencom iščitamo gustoće poveznica između kontiga. Nakon toga konstruiramo *heatmap* (toplinska mapa), grafičku reprezentaciju dobivenog poravnanja Hi-C podataka na gensku sekvencu. Ako je toplinska mapa ispravna, možemo nastaviti s radom i podatke koristiti za analizu pomoću algoritma opisanog u poglavlju 2.

1.2.1. Vinska mušica (*Drosophila melanogaster*)

Radi vremenskih i resursnih ograničenja, za testiranje nisu korišteni svi raspoloživi podatci, već je napravljen niz testnih skupova različite veličine sa namjerom da dobro reprezentiraju potpune podatke, ali su i dovoljno mali za testiranje. Prvi skup je sadržavao samo 4. kromosom, pošto je vrlo malen i lagano je raditi početne testove na njemu. Nadalje je napravljen skup od kromosoma X i 2L. Oba su mnogo veći od kromosoma 4 i prikladni su za naprednije testiranje programa. Hi-C podatci vinske mušice su bili namijenjeni za cijeli genom umjesto samo za gore navedene kromosome, no to nije problem pošto se podatci koji su namijenjeni za preostale kromosome jednostavno neće poravnati jer program za poravnanje ne može poravnati Hi-C podatke na kromosome s kojima ne raspolažemo.

Za poravnanje Hi-C podataka na genske sekvence koristi se paket BWA (Burrows-Wheeler Aligner) [4], konkretno naredba `bwa mem`. Iako paket nije namijenjen za zadaću Hi-C poravnanja, može se koristiti za *pair – end* [9] očitanja kakva mi koristimo (očitanja gdje se poravnanje vrši s oba kraja genske sekvence). Naši podatci zapravo nisu pair – end, no poravnavaju se na vro sličan način. Podatci su oblika Hi-C i dolaze u dvije datoteke te su korištene bwa mem opcije -5SP [5], no datoteke nisu bile jednako imenovane i imale su nastavak na imenu koji je valjalo maknuti te je stoga bila potrebna prilagodba podataka pomoću naredbe `sed`. Valja napomenuti i da bwa mem zahtijeva da su podatci prije poravnanja indeksirani (zahtijeva postojanje datoteka sa informacijama o sekvencama poput imena, duljine itd.) i zato valja prije poravnanja izvršiti naredbu `bwa index` nad sekvencom na koju poravnavamo željena Hi-C očitanja (na FASTA datoteku).

Format datoteke korišten za ispis je SAM. SAM je *tab separated* (odvojen znakom tabulatora) format sa mnogo mogućnosti za ispis, no nama su zanimljiva samo određena polja. Konkretno, zanimljivo nam je prvo polje jer iz broja nakon točke možemo iščitati kojem paru spada koji redak (broj je jednak za parove), gdje parovi predstavljaju Hi-C očitanja iz dvije različite datoteke. Zanimljivo nam je i treće polje u kojem je naziv reference

na koju se poravnava, gdje ako je vrijednost polja * znači da se Hi-C očitanje nije poravnalo na tu referencu te nam je zanimljivo polje četiri u kojem je zapisana lokacija na sekvenci gdje je Hi-C očitanje poravnato. Nakon poravnanja dobiva se datoteka koja izgleda kao u nastavku:

SRR389762.17 77 * 0 0 * * 0 0 TTCCCATGTCTAGAGAAAACGAAAAATGCTCGAGAA HHHHHHHHHHHHHHDHDHHHHHHHHFHHHFHHHHGFE AS:i:0 XS:i:0

SRR389762.17 141 * 0 0 * * 0 0 TGAATCACAGCGAATGCAAAGCTTTTTGGAACAAGT HHDHHHHHHHHEHHHHHHHHHHHHHHHHHHHHFHHHHE AS:i:0 XS:i:0

SRR389762.18 117 NC_004353.4 910196 0 * = 910196 0 CAGTTCTTTTCCTTTTTAGGGAGGTCCTTTATTTCT ?G@=BGGGAEGGFE@G=GGGGBEEB=GGGGFGGGFG MC:Z:36M AS:i:0 XS:i:0

SRR389762.18 185 NC_004353.4 910196 60 36M = 910196 0 ACACCGCATTTTTCTGTATGCTTCTCGCTTTCGTAC <EB>FED9?EEE+FFGDFFEFFFDHFGHGDFHHGFH NM:i:1 MD:Z:27T8 AS:i:31 XS:i:0

SRR389762.19 77 * 0 0 * * 0 0 TTCTCTTCTCTTCTCTTCTCTTCTCTTCTCTTCTCT HHHHHHHHHHHHHHHHHHHHHHHHGHHHHHHHHHHHHH AS:i:0 XS:i:0

SRR389762.19 141 * 0 0 * * 0 0 GAGAAGAGAAGAGAAGAGAAGAGAAGAGAAGAGAAG HHHHHHHHHGHFHFHHHHHHHHH@HHEHEEGEF<DG AS:i:0 XS:i:0

Kao već navedeno, nas zanimaju podatci koji na 3. poziciji imaju ime zapisa (ovdje NC_004353.4) jer to znači da je došlo do poravnanja. Osim toga, po indeksima na prvoj poziciji vidimo da su redci uvijek u parovima i zato želimo da je 5. pozicija popunjena za oba retka, pošto to znači da je Hi-C očitanje poravnato s oba kontiga na koje se pokušao poravnati. Da je poravnato samo jedno Hi-C očitanje, značilo bi da nema poveznice između te dvije lokacije i taj nam podatak nije koristan. U svrhu vađenja potrebnih linija korištena je jednostavna naredba komandne linije grep. Nakon te naredbe ostaju samo dvije linije:

SRR389762.18 117 NC_004353.4 910196 0 * = 910196 0 CAGTTCTTTTCCTTTTTAGGGAGGTCCTTTATTTCT ?G@=BGGGAEGGFE@G=GGGGBEEB=GGGGFGGGFG MC:Z:36M AS:i:0 XS:i:0

SRR389762.18 185 NC_004353.4 910196 60 36M = 910196 0 ACACCGCATTTTTCTGTATGCTTCTCGCTTTCGTAC <EB>FED9?EEE+FFGDFFEFFFDHFGHGDFHHGFH NM:i:1 MD:Z:27T8 AS:i:31 XS:i:0

Sada je još potrebno iz datoteke izdvojiti one linije kojima je 5. pozicija popunjena i u svrhu toga je napisana kraća Perl skripta. Konačno dobivena datoteka ima samo algoritmu korisne linije i sada se može krenuti u analizu ispravnosti podataka. Navedeni proces opisuje slučaj

kada imamo samo jednu sekvencu (kromosom) u datoteci na koju poravnavamo (genskoj sekvenci). U slučaju više sekvenci samo moramo nadovezati sadržaj jedne datoteke na drugu i proces se dalje ne mijenja.

1.2.2. Čovjek (*Homo sapiens*)

Za analizu ljudskog genoma je korištena kombinacija dva podatkovna skupa: kromosom 19 i kromosom 21 te kromosom 17 i kromosom 18. Razlika podataka za čovjeka s kojim raspolažemo od onih za vinsku mušicu je da su Hi-C očitanja za čovjeka lokalizirana samo na spomenute kromosome, što proces poravnanja čini mnogo bržim. Inače to ne mora biti slučaj. Također, podatci su dolazili u odvojenim datotekama, posebno za svaki kromosom i posebno za njihova zajednička Hi-C očitanja, te je te podatke prije početka rada trebalo spojiti u jednu datoteku (jednu za svaki smjer Hi-C očitanja opisanim u 1.1). Hi-C podatke je također trebalo prilagoditi za rad, no ovaj put sa malo drukčijom naredbom `sed`. Osim toga, proces je identičan onome za vinsku mušicu opisanom u 1.2.1.

1.3. Analiza ispravnosti podataka

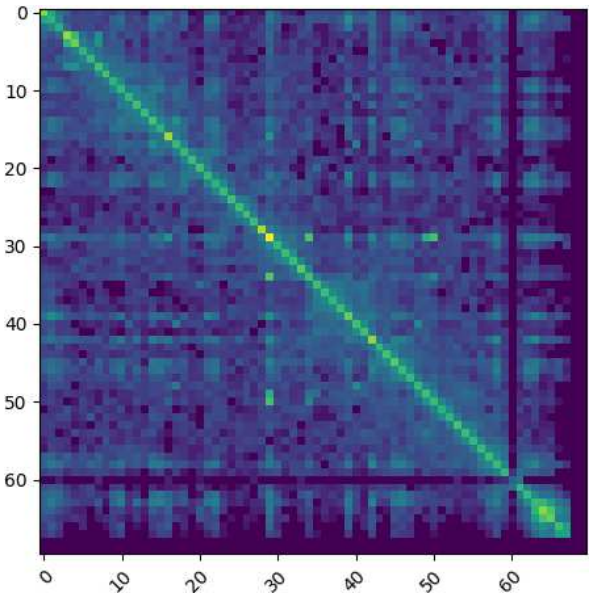
Da bismo potvrdili jesu li nam podatci ispravni, moramo konstruirati *heatmap* (toplinsku mapu). Pri poravnanju, želja nam je da su parovi Hi-C podataka poravnati na točne lokacije, od kojih je većina blizu jedna drugoj. No to se neće uvijek dogoditi. Program će naći poveznice i na drugim lokacijama unutar kromosoma i tamo ih zapisati. Iako je to iz perspektive korisnika greška, zapravo je očekivano ponašanje. Ono što treba provjeriti je jeli taj broj manje vjerojatnih poveznica dovoljno mali. U svrhu toga pišemo program koji generira vizualni prikaz koji nazivamo toplinska mapa. Na apscisi se označuje pozicija na koju se mapira jedan član para Hi-C poravnanja, a na ordinati drugi.

Pošto je broj poveznica velik, određene raspone pozicija poravnanja stavljamo u istu „kutiju“ i na taj način smanjujemo broj potrebnih elemenata za crtanje i određujemo rezoluciju. Taj broj nam označava gustoću poveznica za određen raspon pozicija i prevedemo ga u intenzitet boje. Pošto će neke pozicije imati puno više poveznica od drugih, znatno će se isticati i „gušiti“ ostale pozicije. Zato kao dodatni korak prije crtanja logaritmiramo dobivene intenzitete boja. Dobivena slika je prikazana na slici (Slika 1.2). Na slici vidimo jasno intenzivniju boju na središnjoj dijagonali što upućuje na ispravnost naših podataka jer, kako smo napomenuli, želimo da je gustoća Hi-C očitanja veća među približno jednakim

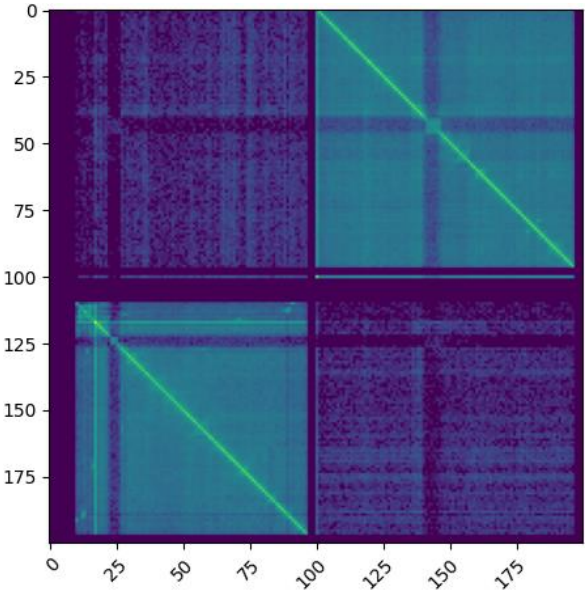
lokacijama na sekvenci. To ponašanje je očekivano, pošto je genska sekvenca već sastavljena, no pomoći će nam kada ćemo uspoređivati dobivenu toplinsku mapu sa našim vlastitim simuliranim podatcima u 1.4.

Pri korištenju sekvenci sa kombiniranim kromosomima u svrhu testiranja programa, toplinska mapa bi trebao biti vrlo sličan, no trebala bi i postojati 4 jasno odvojena područja. Dva će biti svjetlija i imat će dijagonalu. To su 2 kromosoma s poravnatim Hi-C podatcima. Izvan dijagonale se nalaze interakcije pojedinih kontiga unutar samog kromosoma. Druga dva područja neće imati dijagonalu i biti će puno tamnija. To su poveznice između pojedinih kromosoma (poveznice između kontiga različitih kromosoma). Takav primjer je prikazan na slikama (Slika 1.1**,** Slika 1.4Error! Reference source not found.). Na drugoj slici se i jasno vidi problem na koji nailazimo. Pošto krajevi kromosoma imaju veću gustoću poveznica (područje intenzivnije zelene boje), zapravo na tim lokacijama postoji više poveznica između 2 različita kromosoma nego unutar kromosoma. Taj problem se nastoji riješiti funkcijom normalizacije koja pokušava ujednačiti gustoće poveznica na pojedinim kontizima.

Za generiranje prikazanih toplinskih mapa napisan je Python program koji učitava očišćene podatke (podatke nakon izvršenja Perl skripte iz 1.2.1), pronalazi gustoće, sprema ih u matricu i iscrtava toplinsku mapu.



Slika 1.2 Toplinska mapa za *Drosophila melanogaster*, kromosom 4



Slika 1.1 Toplinska mapa za *Homo sapiens*, kromosomi 19 i 21

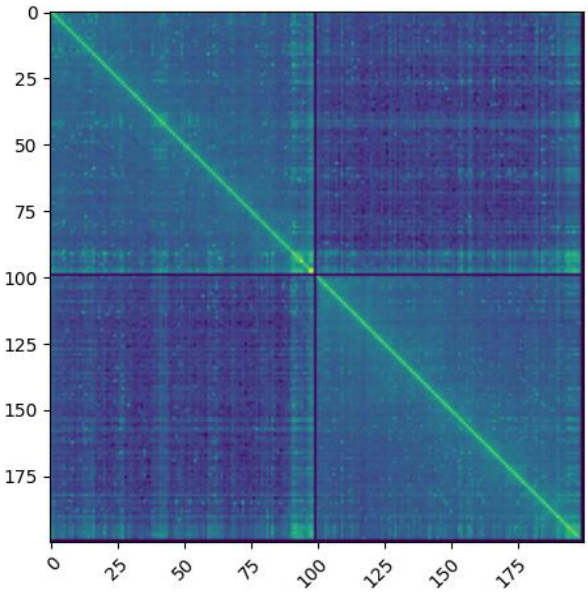
1.4. Simuliranje kontiga

Dostupni podatkovni skupovi genoma sadrže prevelik broj kromosoma i kontiga za testiranje našeg programa. Također, program nije dovoljno brz da bi radio sa tako velikim podatkovnim skupovima. Stoga moramo simulirati svoje vlastite podatke. Podatci s kojima raspolažemo su već sastavljeni (sadrže potpune kromosome), tako da je na nama da ih „izrežemo“, to jest, od njih stvorimo niz kontiga koje ćemo potom izmiješati. To radimo na način da prvo odredimo željene duljine kontiga i zatim na početku svakoga definiramo da je kontig, kako je opisano u FASTA dokumentaciji. Pomoću takvih podataka sada možemo testirati program.

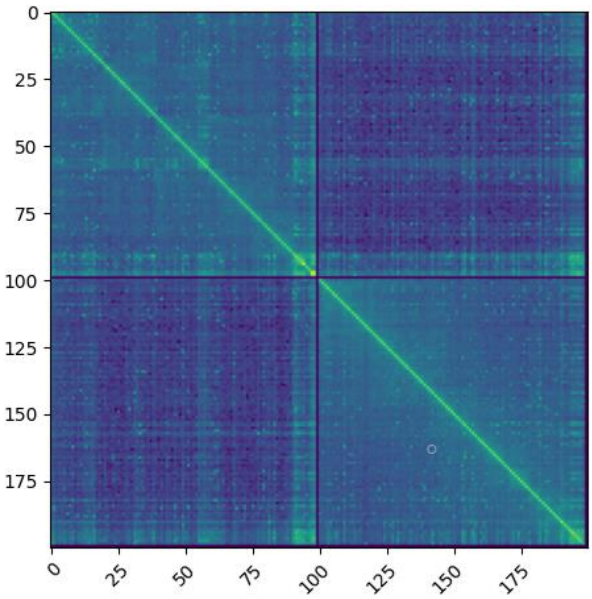
U svrhu zadanog postupka je napisana Perl skripta koja učitava podatke, razdvaja pojedine kontige i zatim ih zapiše u novu datoteku. Nakon ponovnog poravnanja Hi-C podataka na nove, simulirane podatke, moramo filtrirati iz podataka neispravne i nama nepotrebne linije na isti način kao što je opisano u poglavlju 1.2.1 i zatim moramo nacrtati toplinsku mapu. Pošto nismo mijenjali redoslijed kontiga, toplinska mapa bi trebala izgledati jednako ili vrlo slično kao kod originalnih nesimuliranih podataka (pošto smo, da bismo stvorili rupe, morali izbrisati neke genske baze, ipak se očekuje mala razlika u izgledu toplinske mape). Ako nova toplinska mapa izgleda jednako prošloj do razumne mjere, možemo krenuti analizirati dobivene podatke i pisati algoritam.

Ako smo mijenjali orijentaciju nekih kontiga za ispitivanje 3. koraka algoritma, toplinska mapa će izgledati kao na slici (Slika 1.3). Vidimo malu razliku na mjestima gdje se nalaze reverzno komplementirani kontizi. U svrhu reverznog komplementiranja kontiga je, jednako kao i za prošle korake, napisana Perl skripta. Reverzni komplement kontiga je taj isti kontig, ali zapisan unatrag i sa svim bazama zamijenjenim njihovim komplementarnim bazama (A – T, G – C) [6].

Pomoću programa Gepard [7] se može dobiti ispis na kojem se vide reverzno komplementirani kontizi. Kontizi prate dvije međusobno okomite dijagonale, gdje reverzno komplementirani kontizi prate glavnu dijagonalu, dok nekomplementirani kontizi prate sporednu dijagonalu. Kod podataka za vinsku mušicu je reverzno komplementirano 2 kontiga, dok je kod podataka za čovjeka reverzno komplementirano 3. Za čovjeka je također stvoren još jedan podatkovni skup gdje je reverzno komplementirano 2 kontiga.



Slika 1.4 Toplinska mapa za *Drosophila melanogaster*, kromosomi 2L i X



Slika 1.3 Toplinska mapa za *Drosophila melanogaster*, kromosomi 2L i X, kontizi 2 i 3 kromosoma X reverzno komplementarni

2. Algoritam

U ovom poglavlju će biti opisan algoritam za stvaranje uređenih kromosoma i njegovo izvršavanje. Program je u cijelosti pisan u Pythonu i sastoji se od 3 modula koji svaki predstavlja jedan od tri koraka algoritma za stvaranje cjelovitih kromosoma.

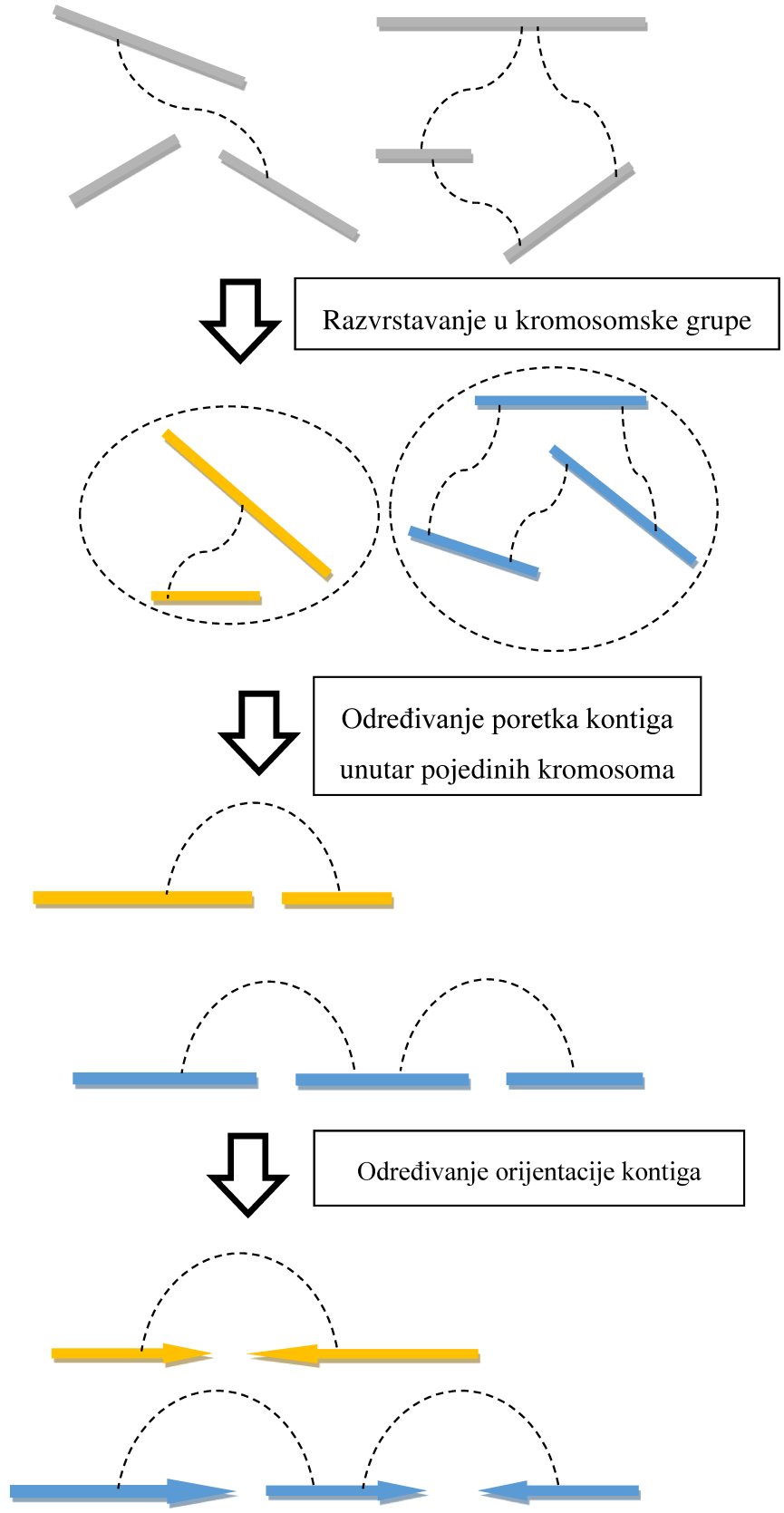
2.1. Općenito o postupku

Slika 2.1 prikazuje općeniti postupak algoritma. U prvom koraku neraspoređene kontige raspoređujemo u pojedine kromosome. Zatim unutar svakog pronađenog kromosoma poredamo kontige u pravilan redoslijed. Konačno, kada su kontizi poredani, svaki kontig pravilno orijentiramo. Prva dva koraka rade na bazi gustoća poveznica među kontizima, dok treći korak radi na principu gustoća poveznica na samim repovima (lijevi i desni kraj) kontiga. Iako je zadnji korak dovoljan za pravilno raspoređivanje kontiga, za velike setove podataka je prespor i zato zahtijevamo prijašnje korake da ubrzamo sam postupak.

2.2. Razvrstavanje u kromosome

Nakon što imamo podatke koje smo pripremili za uporabu (opisano u poglavlju 1.2) možemo početi s implementacijom algoritma. Pošto podatci sadrže višak informacija, cilj nam je stvoriti vremenski i prostorno efikasne podatkovne strukture za rad sa velikim količinama podataka koji su nam na raspolaganju. Program je pisan u Pythonu i stoga su podatci većinom spremani u rječnike standardne biblioteke.

Pošto poravnanja u SAM datoteci dolaze u parovima kako je navedeno u poglavlju 1.2.1, postupak se svodi na ispitivanje pozicije prvog člana para i zatim drugog. Ispitivanjem pozicije možemo utvrditi kojem kontigu pripada koji član para i spremiti ga kao poveznicu ta dva kontiga. Time dobivamo rječnik svih mogućih parova kontiga i njihovih međusobnih poveznica. Na temelju toga možemo iterativno prolaziti kroz rječnik poveznica kontiga i spajati kontige gdje svaki krug iteracije spojimo 2 kontiga s najviše poveznica i zajedno ih spremimo u novi kontig. Dva stara kontiga izbrišemo iz popisa kontiga. Potom izračunamo sve njegove poveznice s preostalim kontizima. To radimo na način da zbrojimo vrijednosti broja poveznica od kontiga koji tvore novi kromosom sa preostalim kontizima. Postupak se



Slika 2.1 Okvirni postupak algoritma. Duguljasti pravokutnici predstavljaju kontige. Isprekidane linije su poveznice između njih. Slika precrtana iz [2]

izvršava iterativno dok ne prođemo kroz sve preostale kontige. Postupak je prikazan na slici (Slika 2.2).

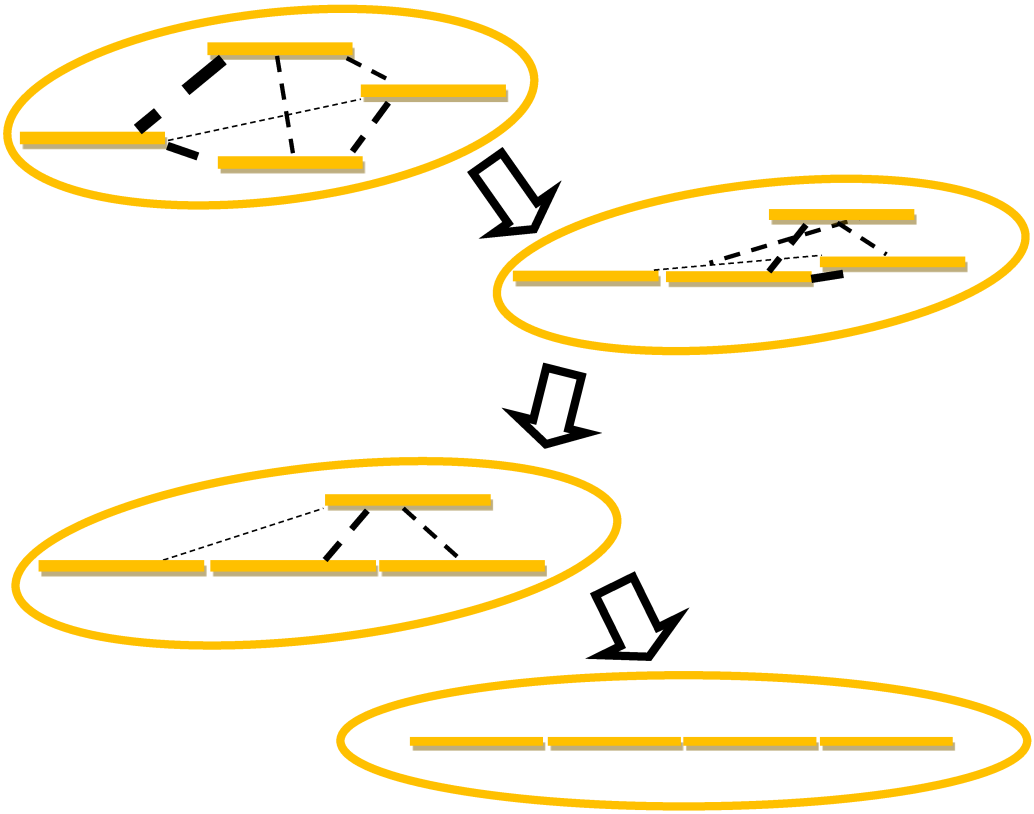
Program zahtijeva da se unaprijed zada predodređeni broj kromosoma i kontiga, kao i duljina svakog kontiga te ime kromosoma. Taj podatak nam je potreban da bismo znali od koje do koje granice indeksa podataka pripada kojem kontigu. Bez tog podatka bi sve spremao u isti kromosom pošto ne zna kojem kromosomu pripada redni broj očitanja (redni brojevi su jednaki za svaki kromosom). Kada je postupak gotov, imat ćemo popis kromosoma sa kontizima koji ga sačinjavaju.

Važno je i spomenuti funkciju normalizacije koja se u ovom koraku koristi prikazan jednadžbom (Jednadžba 1). Naime, dolazi do problema da gustoća Hi-C poveznica nije uniformna po cijelom kromosomu. Neki kontizi imaju puno više poveznica s ostalim kromosomima od drugih te statistički je veća vjerojatnost da imaju više poveznica s nesrodnim kontizima (onima iz drugog kromosoma). Zbog toga nastaje problem da će se neki kontizi krivo razvrstati po kromosomima. Taj problem se nastoji razriješiti korištenjem funkcije koja svodi gustoće Hi-C poveznica svih kontiga na jednaku vrijednost. Takvih funkcija ima mnogo, no ovdje je izabrana ona koja zbroj poveznica 2 kontiga dijeli s umnoškom zbroja svih njihovih poveznica sa svim ostalim kontizima (u biti njihova gustoća). Broj ukupnih poveznica brojimo i spremamo i nakon kraja izvršavanja osnovnog postupka traženja svih poveznica između parova kontiga, sve parove dijelimo s umnoškom tih brojeva. Na taj način kažnjavamo velike kontige i smanjujemo broj krivo razvrstanih kontiga.

Valja napomenuti da je u ovom koraku i implementirano traženje gustoće na repovima (krajevima) kontiga, pošto je dodatni vremenski utrošak za tu operaciju minimalan s obzirom da se vrši pri učitavanju podataka zajedno s traženjem općenite gustoće cijelih kontiga. Sa ovim korakom završenim, dobivene kromosome dalje šaljemo na drugi Python modul koji kontige nastoji poredati unutar kromosoma.

$$x_{i1,j1} = \frac{x_{i,j} + x_{j,i}}{n1 * n2}$$

Jednadžba 1 Jednadžba funkcije normalizacije



Slika 2.2 Postupak određivanja poretka kontiga unutar kromosoma

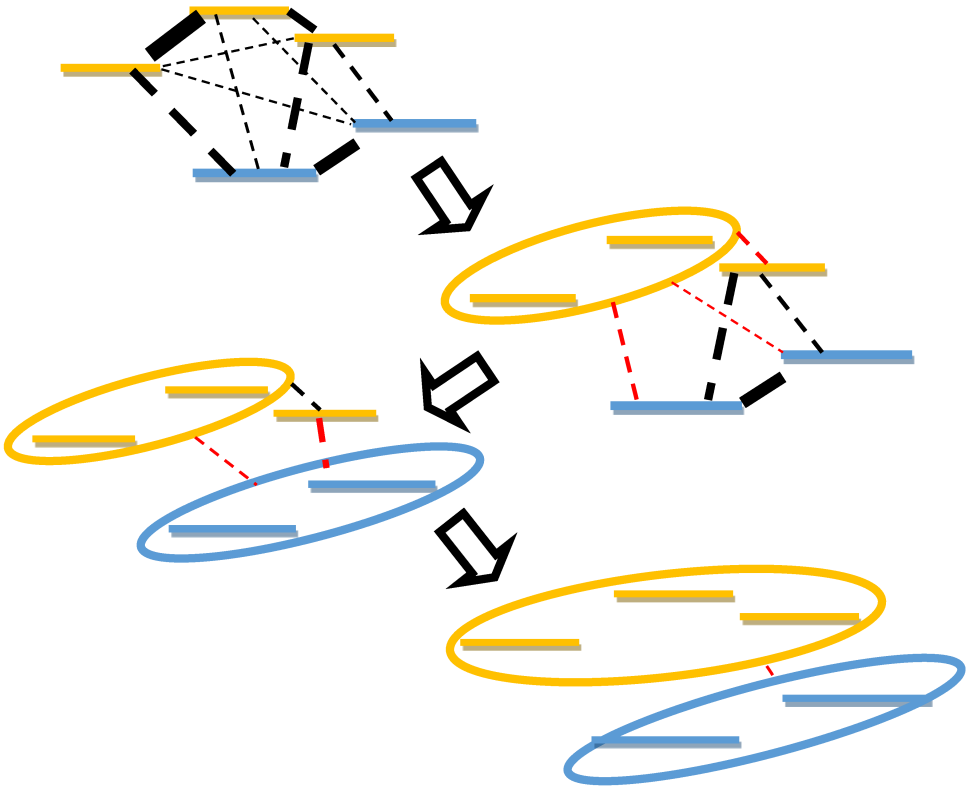
2.3. Poredak kontiga unutar kromosoma

U drugom koraku algoritma nam je cilj kontige sortirati unutar samog kromosoma pošto trenutno samo znamo u kojem se kromosomu nalaze, ne i kako su raspoređeni unutar njega. U tu svrhu ćemo koristiti informacije stečene u prošlom koraku (poglavlje 2.2).

Mi znamo točan broj poveznica između svakog kontiga te na taj način slično kao i u prošlom koraku, možemo odrediti poredak. Naime, susjedni kontizi imaju više međusobnih poveznica od udaljenijih kontiga. Program prvo pronalazi dva kontiga sa najviše poveznica i proglasi da se nalaze jedan pokraj drugoga. Njih makne iz popisa kontiga za koje ne znamo poredak. Zatim za svaki kontig u već poredanim kontizima traži sa kojim drugim kontigom koji još nije poredan on ima najviše poveznica. Kada je našao novi kontig, prilijepi ga uz onaj kontig s kojim ima najviše poveznica, pazeći pritom da je to jedan od kontiga na rubovima već poredanih kontiga.

Ako novo nađeni kontig ima najviše poveznica sa nekim od kontiga koji se ne nalaze na rubu, novi kontig ćemo nastojati „ugurati“ između nekih već postojećih kontiga. To radimo na način da izaberemo s koje strane kontiga s kojim novi kontig ima najviše poveznica ćemo ugurati novi kontig i to na način da biramo onu stranu na kojoj susjedni kontig ima više poveznica s novim kontigom. Taj kontig također maknemo iz popisa neporedanih.

Postupak radimo dok nismo maknuli sve kontige iz popisa neporedanih kontiga (popis kontiga koji je bio rezultat prošlog koraka algoritma). Ovo radi i kada je neki kontig sam po sebi kromosom, pošto će biti dodan zadnji zbog njegovog malog broja poveznica s ostalim kontizima. Proces vršimo za sve nađene kromosome. Konačni rezultat je rječnik lista gdje ključevi predstavljaju kromosome, dok su vrijednosti rječnika liste poredanih kontiga. Postupak je prikazan na slici (Slika 2.3).



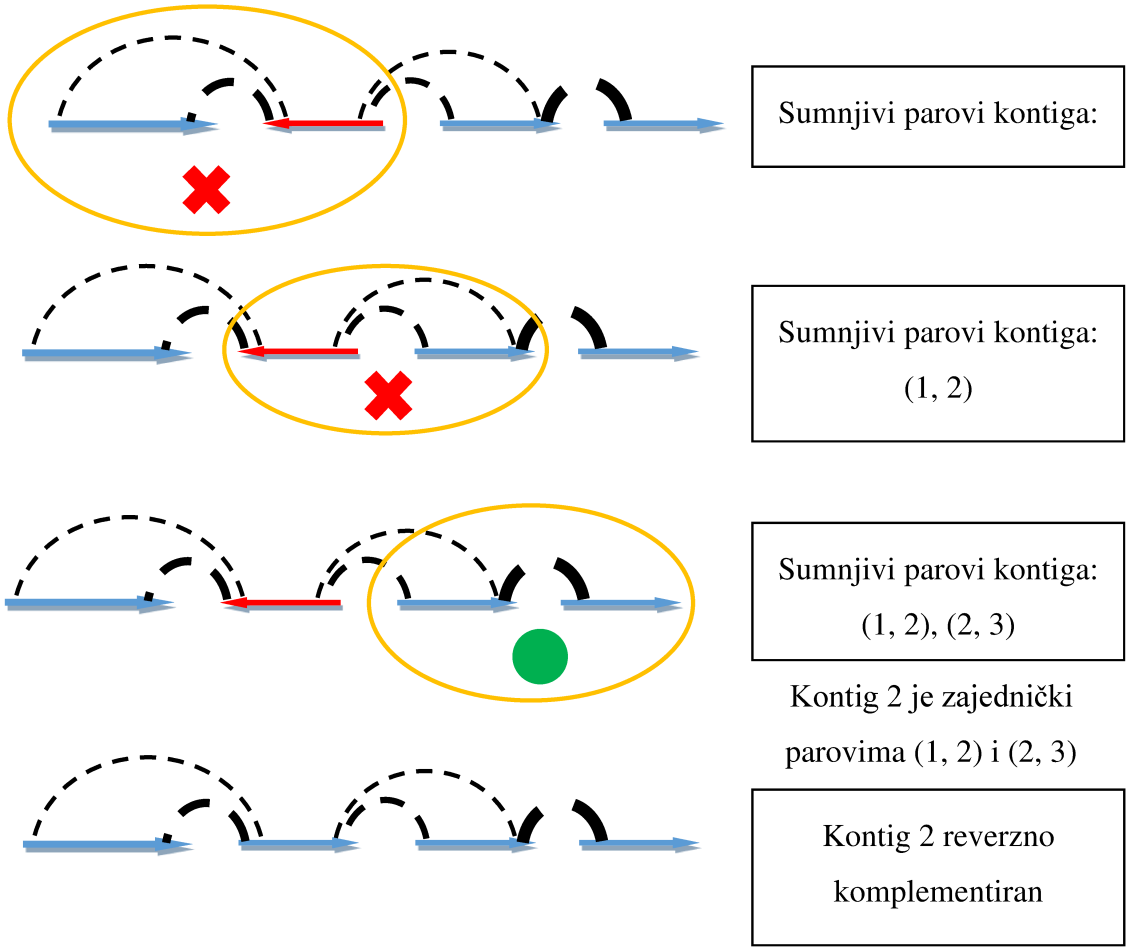
Slika 2.3 Postupak algoritma razvrstavanja u kromosome. Ovali predstavljaju novi kontig. Crvene isprekidane linije su novo izračunate gustoće poveznica. Debljina isprekidanih linija reprezentira gustoću poveznica između kontiga. Kontizi različite boje pripadaju različitim kromosomima. Program prekida s radom kada dostigne predodređeni broj kromosoma. Rezultat su novo nađeni kromosom. Slika precrtana iz [2]

2.4. Određivanje orijentacije kontiga

Zadatak posljednjeg koraka algoritma je otkrivanje kontiga koji su okrenuti s obzirom na orijentaciju ostalih kontiga. Da je kontig okrenut znači da je u podatcima s kojima radimo zapisan njegov reverzni komplement [6]. Otkriti reverzno komplementarne kontige iz podataka je vrlo teško i zahtijeva prijašnje znanje statistike pravilnih kontiga. Kod pravilnih kontiga je broj Hi-C poveznica između susjednih krajeva promatranog kontiga i njegovih susjednih kontiga vrlo visok, dok udaljeni krajevi tih istih kontiga imaju vrlo malo poveznica. Kod reverzno komplementarnih kontiga primjećujemo da je broj poveznica između bliskih krajeva manji nego kod ispravnih kontiga, no i dalje visok, dok je broj poveznica između udaljenih krajeva viši nego kod ispravnih kontiga, no i dalje je mnogo niži nego broj poveznica kod susjednih krajeva.

Da bismo te podatke iskoristili u našu korist, potrebno nam je analizirati podatke i otkriti točne pragove iz kojih možemo detektirati kada je došlo do obrtanja kontiga. Te pragove dobijemo tako da za svaka dva susjedna kontiga podijelimo najveći broj poveznica koji se pojavljuju kod udaljenih krajeva sa najvećim brojem poveznica koji se pojavljuju kod bliskih krajeva. Na taj način dobijemo broj koji možemo koristiti kao filtar za detekciju reverzno komplementarnih kontiga. Kod čovjeka je ta vrijednost 0.15 (15%) za neke testirane kromosome, dok je kod vinske mušice ona 0.05 (5%). Ovaj prag se posebno izračunava za svaki kromosom. Valja napomenuti da ova metoda zahtijeva prijašnje znanje koji su kontizi reverzno komplementarni kao takva nije primjenjiva na pravim skupovima podataka, no navedeni prag se ipak može dobiti iz podataka detekcijom stršećih vrijednosti, što spada u područje umjetne inteligencije (specifično, strojnog učenja) i nadilazi okvire ovog rada.

Navedeni proces izolira parove kontiga na koje se sumnja da bi mogli biti reverzno komplementarni, a same reverzno komplementarne kontige izdvojimo tako da nađemo parove izoliranih kontiga gdje se dva puta pojavljuje isti kontig. To znači da je broj ispravnih poveznica sa oba susjedna kontiga za nađeni kontig smanjen i navodi nas na zaključak da je izolirani kontig reverzno komplementaran. Proces je opisan slikom u nastavku (Slika 2.4). Valja napomenuti ograničenje algoritma. Pošto algoritam zahtijeva da je neki kontig nađen u oba susjedna para sumnjivih kontiga, algoritam ne može klasificirati kontige na samim krajevima kromosoma kao reverzno komplementarne.



Slika 2.4 Postupak okretanja reverzno komplementiranih kontiga

2.5. Ograničenja i moguće nadogradnje

2.5.1. Ograničenja

Algoritam opisan u prijašnjim poglavljima dolazi s određenim ograničenjima, proizašlim iz vremenskih i tehničkih aspekata. Prije svega treba reći da program radi samo za kromosome čiji su svi kontizi jednakih duljina. Ovo je ujedno i najveći limitirajući faktor algoritma pošto u radu sa pravim podatcima ćemo često imati kontige različitih duljina. Ovaj problem je vrlo lako riješiti i originalno je bio planiran biti riješen, ali zbog manjka vremena nije bio riješen. To bi bila prva nadogradnja ako bi se program išao nadograđivati i vjerojatno bi zahtijevao prijašnje znanje svih duljina kontiga i pozicija kontiga, što je nerealno i zapravo nam ukazuje da je problem već riješen, ili neki drugi način koji bi nam jasno iz podataka ukazao kada se radi o različitim kontizima pomoću kojeg bi pri samom učitavanju podataka odvojeno učitali

svaki kontig. S tim rečeno, tim podatcima i metodama se trenutno za ovaj rad ne raspolaže, no program bi se lako nadogradio u slučaju dostupnosti takvih podataka.

2.5.2. Nadogradnje

Dva ograničenja navedena u poglavlju 2.5.1 su aspekti programa koji ne rade. Program nema tu funkcionalnost i nju tek treba implementirati. No što sa stvarima koje rade, a mogu se poboljšati?

Prije svega bih naveo da je program pisan u Pythonu. Iako je Python odličan programski jezik za brzo i efikasno pisanje koda koji radi i dobar je za prezentiranje ideja, za rad s ovako velikim podatcima je jednostavno prespor. U ovom radu se trudilo maksimalno optimizirati performanse korištenjem efikasnih struktura podataka i minimiziranjem praznog rada i konačni rezultati izvršavanja su vrlo dobri (diskutirano u poglavlju 3).

Podatkovni skupovi na kojima se radilo su za standarde pravih podatkovnih skupova vrlo mali. Rješenje ovog problema bi bilo program prepisati u C++-u. C++ je mnogo brži i efikasniji što proizlazi iz činjenice da je mnogo manje *high – level* programski jezik od Pythona i zahtijeva od programera da sam napiše puno koda kojeg Python vrši sam. Jednako kao i Python, raspolaže vrlo efikasnim strukturama podataka te je objektno orijentiran, što ga čini savršenim za implementaciju ovakvog zadatka koji je računalno zahtijevan. Mane C++-a su što zahtijeva mnogo veću ekspertizu i znanje programiranja, a i samog jezika i kao takav je u njemu puno teže otkriti i ispraviti greške. Također je i teže implementirati odvojene module i povezati ih zajedno.

Osim C++-a, smislen izbor bi bio i programski jezik Java koja ima slične prednosti i mane uz dodatak da je još više objektno orijentirana, no preferira se C++.

Osim nadogradnje pomoću drugog programskog jezika, program bi mogao profitirati i od nekih preinaka u pogledu točnosti. Znanstveni rad na kojem je ovaj rad baziran [2] opisuje postupke koji opširno koriste matematičko područje teorije grafova koje dopušta veću točnost i zapravo je idealno za probleme s kojima se ovdje suočavamo. Iako trenutni algoritmi već jesu donekle bazirani na toj teoriji, oni ju ne implementiraju u potpunosti što bi za velike podatkovne skupove vjerojatno bilo obvezno. Naime, metode opisane pomoću grafova su vrlo efikasne i dopuštaju naknadno provjeravanje točnosti i ispravljanje grešaka što trenutni program ne čini. Nadalje, algoritmi za dane grafove su već detaljno opisani i široko dostupni i jedini razlog zašto nisu korišteni je što pisac ovog rada nije imao znanje

potrebno za razumijevanje tih postupaka, a postupci zahtijevaju male preinake da bi radili na konkretnom problemu s kojim se suočavamo što je vrlo teško ostvariti bez prijašnjeg iskustva i razumijevanja. Ipak, trenutni algoritmi rade kako trebaju i sa malo optimizacije bi se približili performansama algoritama opisanim u radu.

Osim navedenih poboljšanja u algoritmu, u poglavlju 2.4 je opisano da bi taj korak algoritma bio poboljšan korištenjem strojnog učenja. Općenito se u cijelom algoritmu opširno koriste razne metode aproksimacije koje služe za razvrstavanje kontiga i njihovo orijentiranje. Većina tih postupaka je dobivena eksperimentalno metodom pokušaja i pogrešaka i mnogo tih funkcija i parametara bi se znatno poboljšali korištenjem metoda optimizacije parametara i traženja funkcija koje strojno učenje pruža. Na primjer, u poglavlju 2.2 se spominje funkcija normalizacije namijenjena ispravljanju gustoća Hi-C poveznica kontiga. Funkcija koja se trenutno koristi je daleko od savršene, no najbolje je što se moglo naći i pomoću strojnog učenja bi se mogla naći savršena funkcija za skoro svaki podatkovni skup, pa čak i pojedini kromosom. Nadalje, u poglavlju 2.4 se spominje parametar filtra koji razdvaja kontige u one koje treba i koje ne treba okrenuti. Taj parametar je dobiven ručnim podešavanjem za svaki kromosom i zahtijeva prijašnje znanje koji su kontizi reverzno komplementarni. Korištenjem područja strojnog učenja koje nazivamo detekcija stršećih vrijednosti bi se taj parametar mogao mnogo točnije naći bez potrebe za ručnim namještanjem na način da nađe kontige koji imaju gustoće spojnica na krajevima drukčije od ostalih kontiga. Te kontige bi mogao izabrati kao sumnjive ili iz njih stvoriti parametar filtra. Time su opisani mogući postupci optimizacije algoritma.

3. Rezultati

U ovom poglavlju se detaljno objašnjavaju i diskutiraju rezultati provedenih testova za testiranje ispravnosti rada algoritma. Prvo se prolazi kroz podatkovne skupove za vinsku mušicu i nakon toga kroz podatkovne skupove za čovjeka. Za svaki podatkovni skup je napravljena tablica za lakšu usporedbu podataka.

3.1. Vinska mušica (*Drosophila melanogaster*)

Za vinsku mušicu su pripremljena 2 podatkovna skupa za koje će se opisati rezultate, te je algoritam još testiran na vrlo malom kromosomu 4 pogodnom za rana testiranja i pojedino na nekim od navedenih kromosoma da bismo uvidjeli ispravnost samih kromosoma.

3.1.1. Podatkovni skup 1

U tablici (Tablica 1) su navedeni rezultati ispitivanja za kromosome X i 2L bez reverzno komplementiranih kontiga. Vidimo da je kromosom 2L potpuno točno poredan, no fali mu jedan kontig koji je prebačen u kromosom X. To se dogodilo zbog već spomenute nesavršenosti funkcije normalizacije (poglavlja 2.2 i 2.5.2) koja je, zbog gustoće poveznica kontiga 5 i 10 koji se stoga privlače, ih oba prebacila u kromosom X. Osim toga je greška da je krivo okrenut kontig 2, što je posljedica parametra filtriranja reverzno komplementarnih kontiga, opisanog u poglavljima 2.4 i 2.5.2. Parametar je namješten globalno za sve kromosome, umjesto lokalno za svaki pojedini. Greška se lako ispravlja skupa s ostalim ograničenjima opisanim u poglavlju 2.5.2 na način da se parametar filtra dinamički kalkulira za svaki kromosom tokom izvođenja te se stoga ova greška može i ignorirati, pogotovo zato što znamo da u ovom podatkovnom skupu nije došlo do krivog okretanja kontiga.

Nasuprot kromosomu 2L, kromosom X ima mnogo više grešaka. Većina tih grešaka dolazi iz manjka optimalnosti funkcije normalizacije. Nadalje, algoritam koji izvršava korak traženja poretka kontiga je vrlo kaotičan. I za malu promjenu u rang listi gustoća po kojima on radi se poredak može dramatično promijeniti. Ovdje vidimo da su neki kontizi (8, 7 i 9, 10) zapravo dobro poredani, no između njih su krivo ugurani ostali kontizi. Opet, ovaj problem se rješava boljom funkcijom normalizacije. Osim navedenih grešaka se pojavljuje

i krivo okrenuti kontig 8, koji je krivo okrenut zbog jednakih razloga kao i krivo okrenuti kontig 2 kod kromosoma 2L i također bi se trebao ignorirati.

3.1.2. Podatkovni skup 2

Ovaj podatkovni skup je jednak prethodnom (poglavlje 3.1.1) osim toga što su kontizi 2 i 3 kromosoma X reverzno komplementirani u svrhu testiranja trećeg koraka algoritma. Što se tiče kromosoma 2L, vidimo da nema razlike u odnosu na prošli podatkovni skup. Opet je kontig 5 prebačen u kromosom X i opet je krivo okrenut kontig 2 kojeg možemo ignorirati. Zanimljiviji su rezultati koje vidimo kod kromosoma X. Ovdje je on mnogo točnije i ljepše poredan nego u prošlom podatkovnom skupu i to je zato što su reverzno komplementarni kontizi prevagnuli funkciju normalizacije da točnije klasificira gustoće poveznica kontiga. Jedina greška kod poretka je zapravo kontig 6, koji je zalijepljen za kontig 10, te je naknadno između njih uguran kontig 5 što je posljedica visokih gustoća kontiga 5 i 10, a i kontiga 6 i 10.

Kada promatramo okretanje reverzno komplementarnih kontiga, kontigu 8 je pravilno ispravljena orijentacija, no kontigu 7 nije. To je za očekivati, pošto algoritam gleda samo susjedne kontige zbog brzine izvođenja, a kontigu 7 nedostaje jedan susjedni kontig (kontig 6), te stoga nije mogao biti okrenut, ali algoritam je ispravno našao sumnju između kontiga 7 i 8 što upućuje na njegov ispravan rad. Podatkovni skup je prikazan u tablici (Tablica 2).

Rezultati testiranja	**Točni rezultati**
Kromosom 2L: 4, 3, 2!, 1 Kromosom X: 6, 9, 10, 5, 8!, 7	Kromosom 2L: 1, 2, 3, 4, 5 Kromosom X: 6, 7, 8, 9, 10

Tablica 1 Rezultati za *Drosophila melanogaster*, kromosomi X i 2L, nema reverzno komplementarnih kontiga

Rezultati testiranja	**Točni rezultati**
Kromosom 2L: 4, 3, 2!, 1 Kromosom X: 7, 8!, 9, 10, 5, 6	Kromosom 2L: 1, 2, 3, 4, 5 Kromosom X: 6, 7!, 8!, 9, 10

Tablica 2 Rezultati za *Drosophila melanogaster*, kromosomi X i 2L, kontizi 2 i 3 kromosoma X reverzno komplementirani

3.2. Čovjek (*Homo sapiens*)

Već iz samih tablica možemo vidjeti da se podatkovni skupovi za čovjeka ponašaju puno bolje nego oni za vinsku mušicu. Uočavamo značajno manju grešku sortiranja kontiga u kromosome i poretka kontiga unutar kromosoma i manju grešku detekcije reverzno komplementarnih kontiga. Valja napomenuti da su kromosomi ovih podatkovnih skupova mnogo veći od onih kod vinske mušice.

3.2.1. Podatkovni skup 1

Podatkovni skup je prikazan u tablici (Tablica 3). Kontizi su svi pravilno raspoređeni u njihove odgovarajuće kromosome. Dok su kontizi kromosoma 19 savršeno poredani, kod kontiga 21 vidimo samo jednu grešku koja je proizašla iz snažne povezanosti kontiga 6 i 10. Što se tiče okretanja reverzno komplementarnih kontiga, program pravilno nije okrenuo niti jedan kontig, pošto u ovom podatkovnom skupu nema reverzno komplementarnih kontiga.

Rezultati testiranja	**Točni rezultati**
Kromosom 19: 5, 4, 3, 2, 1 Kromosom 21: 9, 8, 7, 6, 10	Kromosom 19: 1, 2, 3, 4, 5 Kromosom 21: 6, 7, 8, 9, 10

Tablica 3 Rezultati za *Homo sapiens*, kromosomi 19 i 21, nema reverzno komplementarnih kontiga

3.2.2. Podatkovni skup 2

Ovaj podatkovni skup prikazan u tablici (Tablica 4) je jednak prošlom podatkovnom skupu (3.2.1), sa jedinom razlikom da je kontig 2 kromosoma 19 reverzno komplementaran i da su kontizi 2 i 4 kromosoma 21 reverzno komplementirani. Sortiranje u kromosome i poredak kontiga unutar kromosoma je jednako kao u prošlom podatkovnom skup i vrlo točno.

Zanimljiva stvar ovdje su reverzno komplementirani kontizi. Program je ispravno okrenuo kontig 2 kromosoma 19 i ispravno je okrenuo kontig 2 kromosoma 21. Jedina greška se pojavljuje kod kontiga 4 kromosoma 21, kojeg algoritam nije ispravno identificirao kao reverzno komplementiranog. To proizlazi iz činjenice da je kromosom 21 krivo poredan i kontig nije mogao biti ispravno okrenut pošto njegov lijevi susjed ne postoji (ovo

ograničenje je opisano u poglavlju 2.4). No, i s ovim greškama na umu, rezultati su zadovoljavajući.

Rezultati testiranja	**Točni rezultati**
Kromosom 19: 5, 4, 3, 2!, 1 Kromosom 21: 9, 8, 7!, 6, 10	Kromosom 19: 1, 2!, 3, 4, 5 Kromosom 21: 6, 7!, 8, 9!, 10

Tablica 4 Rezultati za *Homo sapiens*, kromosomi 19 i 21, kontig 2 kromosoma 19 reverzno komplementaran, kontizi 2 i 4 kromosoma 21 reverzno komplementarni

3.2.3. Podatkovni skup 3

Rezultati testiranja ovog podatkovnog skupa su prikazani u tablici (Tablica 5). Kontizi su pravilno raspoređeni po kromosomima i savršeno poredani, do sada najbolji rezultati. Točnost podataka znatno ovisi o funkciji normalizacije i „ponašanju“ kontiga i ovdje su se podatci dobro prilagodili. Nažalost, program nije detektirao niti jedan reverzno komplementarni kontig i pri analizi podataka se ne vidi značajna razlika između krivo i točno okrenutih kontiga, stoga program nije niti mogao detektirati reverzno komplementarne kontige. Usprkos tome, rezultati su zadovoljavajući i u globalu su rezultati za čovjeka mnogo bolji nego rezultati za vinsku mušicu.

Rezultati testiranja	**Točni rezultati**
Kromosom 19: 5, 4, 3, 2, 1 Kromosom 21: 10, 9, 8, 7, 6	Kromosom 19: 1, 2, 3!, 4, 5 Kromosom 21: 6, 7, 8!, 9, 10

Tablica 5 Rezultati za *Homo sapiens*, kromosomi 17 i 18, kontizi 3 kromosoma 17 i 18 reverzno komplementarni

Zaključak

Metode sastavljanja cjelovitog genoma pomoću Hi-C podataka baziranim na kromatinskim gustoćama među pojedinim kontizima su se pokazale kao vrlo efektivan i točan način sastavljanja genoma. Algoritmi i metode implementirane u ovom radu odstupaju donekle od metoda prezentiranim u referentnom radu [2] i ne dostižu stupnjeve točnosti i skalabilnosti prezentiranom u istome. No, usprkos tome, ovisno o korištenom podatkovnom skupu, ovdje implementirane metode dostižu zadovoljavajuće stupnjeve točnosti i ponašaju se očekivano s obzirom na testne podatke. Ta činjenica govori u prilog robusnosti i svestranosti Hi-C podataka, budući da neovisno o načinu korištenja podataka, dobiveni rezultati mogu biti zadovoljavajući, pa čak u nekim podatkovnim skupovima skoro savršeni. Još jedna prednost korištenja Hi-C metoda je njihova mogućnost nadogradnje modernim metodama. U poglavlju 2.5.2 navedene su neke mogućnosti nadogradnje algoritama i jedna od njih koristi područje strojnog učenja, konkretno metode detekcije stršećih vrijednosti, čime bi se znatno povećala točnost podataka. Neovisno o tome, Hi-C podatci su se dokazali kao efektivna metoda u sastavljanju djelomično sastavljenih genoma i u godinama koje slijede sigurno će biti jedna od glavnih metoda korištena u donošenju brzog i jeftinog sastavljanja velikih genoma. Naime, za takve genome je i dalje teško dobiti točna očitanja bez velikog utroška vremena i novca, što za neke eksperimente nije dopustivo. Smanjenjem tih ograničenja će se znatno unaprijediti znanost genetike i genskog modificiranja što će otključati nova vrata u preciznom modificiranju i analiziranju gena.

Literatura

[1] https://www.genome.gov/human-genome-project

[2] Joshua N Burton, A. A. (2013). Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. *Nature Biotechnology*.

[3] https://www.ncbi.nlm.nih.gov/

[4] http://bio-bwa.sourceforge.net/

[5] https://phasegenomics.github.io/2019/09/19/hic-alignment-and-qc.html

[6] https://en.wikipedia.org/wiki/Complementarity_(molecular_biology)

[7] https://cube.univie.ac.at/gepard

[8] https://en.wikipedia.org/wiki/Chromosome_conformation_capture

[9] https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html

[10] https://www.ncbi.nlm.nih.gov/genbank/

[11] https://www.illumina.com/

Sažetak

Prilikom očitavanja i sekvenciranja gena, ne možemo odmah sekvencirati cijeli genom. U pravilu se dobije više manjih očitanja koje nazivamo kontizima. To su djelomično sastavljeni kraći dijelovi koje još treba spojiti u cijeli genom (niz kromosoma). Jedna od metoda korištena za taj proces koristi podatke o kromatinskim poveznicama između kontiga, specifično, ovdje korištena metoda to radi pomoću Hi-C podataka [8]. Hi-C podatke moramo poravnati sa kontizima i zabilježiti međusobni broj poveznica između kromosoma, između kontiga i unutar samih kontiga. Te poveznice, a i sam raspored kromatina, prati određenu distribuciju na temelju koje možemo vući zaključke. Tako dobivene podatke o poveznicama analiziramo i pomoću dobivenih saznanja o njihovim vezama kontige prvo raspoređujemo u pojedine kromosome, zatim unutar kromosoma poredamo kontige i na kraju orijentiramo kontige. Prvi korak radimo na osnovi toga što kontizi unutar kromosoma kojem pripadaju međusobno imaju više kromatinskih poveznica. Drugi korak proizlazi iz činjenice da među susjednim kontizima ima više kromatinskih interakcija nego među više udaljenim kontizima i ta povezanost linearno opada s udaljenošću. Treći i zadnji korak ostvarujemo pomoću znanja da je više interakcija među krajevima (repovima) kontiga koji su susjedni nego onih koji su na udaljenijim krajevima kontiga. Pri cijelom procesu treba paziti na činjenicu da kromatin nije jednako gusto raspoređen među svim kontizima i neki će statistički imati više međusobnih interakcija zbog jednostavne činjenice da imaju gušće (češće) Hi-C veze. Konačno, dobivene rezultate spajamo u kromosome i uspoređujemo točnost sa referentnim podatcima. Program je testiran na podatcima vinske mušice (*Drosophila melanogaster*) i čovjeka (*Homo sapiens*). Navedeni rad je izrađen prema znanstvenom radu [2].

Ključne riječi: bioinformatika, Hi-C, sastavljanje genoma, sastavljanje kromosoma, kromatin

Summary

During the act of genome sequence alignment, we are not able to sequence the entire genome. Normally, the result is a number of smaller sequences which we call contigs. Contigs are partially assembled smaller pieces which yet need to be assembled into an entire genome (a set of chromosomes). One of the methods used for this process uses data about chromatin interactions between contigs, specifically, the method used here uses Hi-C data [8]. The Hi-C data needs to be aligned along the contigs and we need to measure the number of links between chromosomes, between contigs and inside the contigs themselves. These links, along with the very distribution of chromatin, follows a set distribution with the help of which we can draw conclusions. The produced data is analyzed and, using the knowledge we gathered about the links, the contigs are first separated into specific chromosomes, then ordered within the chromosomes and lastly, oriented into the right direction. The first step is based on the fact that the contigs have more mutual links within the chromosomes they belong to. The second step follows the idea that neighboring contigs have more mutual links that those contigs that are further apart, and this linkage density falls linearly with contig distance. The third and last step comes from the knowledge that neighboring contig ends (tails) have more links that contig ends that are further apart. During the whole process, it is vital to take note of the fact that chromatin density is not equally distributed among all contigs and some will have a statistically higher likelihood of having an interaction with other contigs simply from the fact that they have more dense (frequent) Hi-C links. Finally, the output data is assembled into chromosomes and we compare the results with reference data. The program has been tested on fruit fly (*Drosophila melanogaster*) datasets and human (*Homo sapiens*) datasets. This thesis is based on a paper [2].

Keywords: bioinformatics, Hi-C, genome assembly, chromosome assembly, chromatin