Osnovna ideja svodi na sekvenciranje dijelova podataka u cjelinu, koja predstavlja genom. Cilj algoritma za sastavljanje genoma jest pretvoriti očitanja, dobivenih iz sekvenci, u izlazni niz tj. genom, koristeći međusobna preklapanja tih očitanja. Preklapanje se svodi na preklapanje sufiksa jednog niza sa prefiksom drugog niza, odnosno preklapanje očitanja. Moderni algoritmi za sastavljanje genoma temelje se na algoritmima nad grafovima. Jedna od danas poznatih metoda jest metoda Preklapanje-Razmještaj-Konsenzus, a to je upravo tema ovog rada. Zamislimo graf, njega čine vrhovi i bridovi koji povezuju vrhove. Bridovima se može proći samo u jednom smjeru, u takvom slučaju graf zovemo usmjereni graf. Svaki usmjereni brid predstavlja vezu između izvorišnog i odredišnog vrha. Skupina bridova tvori šetnje prilikom kojih prolazimo preko vrhova u nekom redoslijedu, a sve na način da je odredišni vrh jednog brida zapravo izvorišni vrh za slijedeći brid. Graf konstruiramo na način da vrhovima pridjelimo očitanja, a ako postoji preklapanje između tih očitanja, onda se to očitanje manifestira u bridu između ta dva vrha.

OLC pristupa se svodi na traženje puta u grafu koji prolazi kroz sve vrhove u grafu jedanput, te pritom prelazeći preko očitanja i preklapanja tvori konačni niz, tj genom. Osnovni problem traženja takvoga puta, tzv. Hamiltonovog puta je vrijeme rješavanja problema. Ovaj problem kategorizira se kao NP potpun problem, nije ga moguće obaviti u dostupnom vremenu. Štoviše, za NP potpune probleme dokazano je da se ne mogu rješiti pomoću računala danas. Iz tog se razloga koriste heuristike kojima će se graf što je moguće više pojednostaviti i na taj način doći do rješenja.

U ovom radu programski je implementirana jedna od tri faze OLC-a, faza razmještaja. Faza razmještaja ostvarena je nekolicinom metoda, objašnjavat će se u istom redoslijedu kao što se one izvode u programu.

Metode preklapanja

U ovom poglavlju navode se metode koje se koriste prilikom faze razmještaja. Metode će se navoditi slijedno, istim redoslijedom kao što se izvršavaju u programu. Proces započinje s prilagodbom izvornih podataka, zatim slijedi pročišćavanje odnosno filtriranje kako bismo dobili najmanju moguću potrebnu količinu podataka – za lakše izvođenje programa i manje korištenje resursa. U konačnici se sastavlja graf i nad njim se izvode funkcije kako bismo došli do kontiga, čiji skup čini reprezentaciju DNA.

**1Pretvaranje preklapanja u oblik sa „lastinim repom“**

Ovu radnju potrebno je obaviti jer graf zahtjeva zapise oblikovane kao tzv. lastini repovi. Pritom prefiks ili sufiks jednog brida mora biti poravnat s prefiskom ili sufiksom drugoga.

Programski odsječak učitava početne i krajnje koordinate preklapanja, a vraća tzv hang-ove.

Ovim procesom nismo sto posto sigurni od pogreške, ali one će biti ispravljene slijedećim algoritmima.

**2Filtriranje očitanja**

Moguća je situacija u kojoj se jedno očitanje u potpunosti preklapa sa drugim očitanjem, tj jedno očitanje sadrži u sebi i neko drugo očitanje. Pritom je moguće u potpunosti maknuti očitanje koje je sadržano u većem očitanju. Ovaj korak jako pridonosi uklanjanju velike količine podataka, a da se pritom ne izgube nikakve bitne informacije.

**3Prilagođavanje lastinih repova**

Ovaj algoritam izravno se nastavlja na prethodni, i to na na način da nastoji skratiti ili produljiti dijelove preklapanja. Ovaj korak ispravlja forsirano jednake duljine lastinih repova koje je stvorio prethodni korak. Prvotno se računaju grube veličine preklapanja, zatim ovisno o tipu preklapanja nastojimo maksimalno smanjiti krajeve preklapanja. Računaju se hangovi novog preklapanja, stopa pogreške trenutnog preklapanja i ukupna stopa pogreške.

**4Filtriranje tranzitivnih preklapanja**

Zamislimo tri očitanja, i sva imaju preklapanje jedno s drugim. Kako su nam za definiciju preklapanja dovoljna dva očitanja, treće očitanje na istom području ne donosi nikakvu novu informaciju, stoga ga je sigurno maknuti. Informacija o preklapanju će ostati sačuvana. Ovim postupkom neće se promijeniti šetnje u grafu.

**5Filtriranje kratkih preklapanja**

Ovaj jednostavni algoritam uklanja preklapanja koja sačinjavaju jako mali postotak ukupnog očitanja, odnosno za nekoliko redova veličine su kraća od samog očitanja.

**6Filtriranje pogrešnih preklapanja**

Ovaj proces svodi se na uklanjanje pogrešno stvorenih preklapanja. Pogrešno stvorena prelapanja prestavljaju isto veliki problem kao i pogreške prilikom očitanja. Prilikom stvaranja preklapanja zapisuje se vjerojatnost pogreške, i ovaj algoritam uklanja sva preklapanja sa vjerojatnosti pogreške većom od dozvoljenje.

**7Kreiranje grafa**

Nakon što su podaci pripremljeni prethodnim algoritmima, vrijeme je za implementaciju grafa. Svako očitanje čini jedan čvor, a svako preklapanje je prikazano dvama bridovima. Vezano uz strukturu, svaki čvor sadrži dva distinktna polja bridova, jedan za početak očitanja i drugi za kraj očitanja. Svaki brid ima svoj par, drugi brid, koji je izgrađen od istog preklapanja ali sa obrnutom orjentacijom.

**8Pojednostavnjenje grafa**

Izgrađeni graf moguće sadrži neke oblike pogrešaka ili strukture koje nam ne odgovoraju. Tada je potrebno primjeniti niz metoda koje će graf dovesti u odgovarajuć oblik, i samim time rješiti pogreške.

8.1Uklanjanje čvorova bez bridova

8.2Uklanjanje ogranaka

Ogranci su zapravo čvorovi „spojeni na graf“ samo sa bridovima u jednom smjeru. Zamislimo slijedeću situaciju: u čvoru dolazi do račvanja na dvije strane, na jednu stranu se nastavlja veza sa drugim čvorom koji se više ne nastalja ni na što, a na drugu stranu se nastavlja čvor koji je dalje povezan na ostatak grafa. Čvor sa prve spomenute strane, ogranak, je problematičan, i njega se u ovom koraku uklanja.

8.3Uklanjanje mjehurića

Mjehurići su strukture koje se sastoje od dva ili više odvojenih puteva koji dijele isti početni čvor zvan korijen, i isti konačni čvor. Kroz mjehurić se prolazi i traži se najpokriveniji put. Svi ostali putevi se uklanjaju.

Ove metode mogu se zvati više puta za redom u alternirajućem slijedu, i na taj način se graf maksimalno pojednostavljava.

**9Ekstrahiranje kontiga**

Graf je prethodnim koracima maksimalno pojednostavljen, a da sadrži kvalitetnu informaciju. Slijedeći korak je, kao što naslov ovog procesa daje naslutiti, izlučivanje podnizova. Postoji mogučnost da nakon svih ovih metoda graf još sadrži problematične dijelove. Stoga uzimamo podnizove, ograničene dijelove niza u kojem nema grananja, to su slijedno povezani čvorovi koji imaju samo jednog susjeda prije i jednog susjeda poslije sebe. Algoritam funkcionira na način da se krene od proizvoljnog čvora i „putuje“ po grafu sve dok se ne dođe do prvog grananja. Takvi putevi se spremaju i rangiraju. Potom se za bolje kandidate početnih čvorova ponovno prolazi kroz graf, a u trenutku dolaska do prvog grananja odlučuje se kojom granom krenuti. U konačnici se od svih kandidata za najdulji put odabire najdulji.