Bioinformatika

Faza razmještaja u OLC paradigmi sastavljanja genoma implementirana u Canu asembleru

FER, 2016/2017

Ivan Soldo; 0036471227  
Tomislav Vulić; 0036460396

# Problem

Razvojem strojeva za očitavanje početkom 21. stoljeća u fokus su se vratili asembleri genoma koji kao ulazne podatke primaju relativno kratka očitanja (red veličine 20 kb). Problem kod rekonstruiranja genoma iz takvog skupa ulaznih podataka je taj što zbog relativno malih očitanja proces postaje vrlo osjetljiv na dugačka ponavljanja te šum očitanja. Međutim, ulazni skup očitanja je redundantan te je moguće usrednjavanjem i filtriranjem dobiti rezultate relativno visoke točnosti.

OLC paradigmu čine tri faze: preklapanje (overlap), razmještaj (layout) te konsenzus (consensus). U prvoj se fazi iz ulaznih podataka pronalaze očitanja sa najboljim preklapanjem (najdulje preklapanje sa najvišom razinom sličnosti po nekoj mjeri, npr. Levenshteinova udaljenost). U drugoj fazi, kojom se ovaj rad i bavi, uzimaju se podaci o preklapanju te se izgrađuje graf koji prikazuje moguće putanje između očitanja. Algoritam razmještaja filtrira očitanja koja su nepovoljna za izgradnju grafa, odabire najbolja preklapanja te razrjeđuje inicijalni graf susjedstva očitanja. Nakon razmještanja, OLC treba proglasiti konsenzusni slijed na osnovu višestrukog poravnanja sljedova dobivenih u svakom bloku dobivenom u fazi razmještanja.[[1]](#footnote-1)

# 

# Algoritam

Algoritam je implementiran po uzoru na bogart funkciju Canu asemblera, koji je temeljen na *greedy* BOG (*best overlap graph)* algoritmu (Miller et al. 2008.). Taj algoritam učitava samo najbolja, tj. najdulja preklapanja između očitanja. BOG ignorira obuhvaćena (*contained)* preklapanja, tj. ona preklapanja u kojem je jedno očitanje nadniz drugoga[[2]](#footnote-2).

Inicijalno filtriranje provodi se na slijedeći način:

*// C = informacije o preklapanjima sa nekim očitanjem r*

*// n = broj preklapanja*

*// C\_f = informacije o filtriranim preklapanjima sa r*

**dok je** i ≤ n **radi**

**ako je** C(i) obuhvaćen **onda**

**preskoči**

**inače**

dodaj C(i) u C\_f

**vrati** {C\_f}

Nakon što su izbačena nepovoljna očitanja, nastavlja se sa procjenom kvalitete preostalih preklapanja. Preklapanja za svako očitanje se soritraju silazno prema sličnosti preklapajućih podnizova. Sada imamo n očitanja i njihova preklapanja. Preklapanja se dalje mogu razvrstati na lijevo, odnosno desno preklapanje s obzirom na promatrano očitanje. Ovime se stvara redundancija u smislu da i-ti čvor koji se preklapa sa j-tim čvorom, ima u svojim preklapanjima preklapanje sa j-tim čvorom, te j-ti čvor ima u svojim preklapanjima preklapanje sa i-tim čvorom, samo na drugoj strani.

Čvor je struktura koja se identificira rednim brojem očitanja iz ulaze .fasta datoteke, te sadrži i dvije strukture koje opisuju lijeva, odnosno desnta preklapanja

Čvor

Indeks očitanja

Popis lijevih preklapanja

Popis desnih preklapanja

Struktura preklapanja sadrži sve što se nalazi u zapisu u .mhap formatu

Preklapanje

Sličnost // [0.000000,1.000000]

Broj zajednlčkih min-mera

Indeksi dvaju preklapajućih očitanja

Koordinate preklapanja prvog očitanja

Koordinate preklapanja drugog očitanja

Duljina prvog očitanja

Duljina drugog očitanja

BOG algoritam zatim izabire najbolju putanju između svih čvorova u grafu (nije nužno obići sve čvorove).

U ovom je radu implementiran vrlo naivan način konstrukcije BOG-a. Kao početni čvor uzima se onaj koji nema lijeva preklapanja, tj. onaj koji ima globalno najmanju lokalno najveću sličnost sa lijevim preklapanjem. Ti se čvorovi stavljaju u listu starters te se iz svakog od tih čvorova pokušava napraviti BOG.

**za sve** a **iz** starters **radi**

trenutna\_putanja **= {}**

**dok** a nije prazan

**ako je** a.desni\_čvor neiskorišten **onda**

**spremi** a.desni\_čvor **u** trenutna\_putanja

a = a.desni\_čvor

**spremi** trenutna\_putanja **u** moguće\_putanje

Zatim se iz liste mogućih putanja po nekom kriteriju bira ona najbolja. U ovom je radu implementirano naivna metrika, da najdulja putanja ujedno znači i najbolju.

# Rezultati

# Zaključak

# Literatura

1. M. Šikić, M. Domazet-Lošo: Bioinformatika, 2013. [↑](#footnote-ref-1)
2. „Aggressive assembly of pyrosequencing reads with mates“, Miller et al. 2008. [↑](#footnote-ref-2)