SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

BIOINFORMATIKA

Faza razmještaja u OLC paradigmi sastavljanja genoma

Domagoj Boroš, Ivan Jurin, Mateja Škriljak

Voditelj: prof.dr.sc. Mile Šikić

Zagreb, siječanj, 2017.

**Sadržaj**

[1. Uvod 1](#_Toc472258994)

[2. OLC paradigma sastavljanja genoma 2](#_Toc472258995)

[3. *Miniasm* algoritam 3](#_Toc472258996)

[4. Zaključak 4](#_Toc472258997)

[5. Literatura 5](#_Toc472258998)

[6. Sažetak 6](#_Toc472258999)

# Uvod

Danas postoje razne metode sekvenciranja genoma koje nude rješenja prihvatljive kvalitete. Problem je složen i nesavršen, pri čemu različita rješenja nude različite duljine očitanja te varijabilnu točnost. Susrećemo se i s problemom različite razine pokrivenosti očitanja, pri čemu se javlja problem razlomljenosti slijeda u više dijelova s procijepima između te sastavljanjem ne možemo dobiti jedinstveno potpuno rješenje.

Poseban problem je *de novo* sastavljanje, kada se sastavlja genom koji nam nije unaprijed poznat te nemamo referentni genom za provjeru očitanja, čime je čitav postupak dodatno otežan.

SLIKA 1

Rezultantni sastavljeni genom nakon većine postupaka nije potpun kontinuirani slijed, stoga se predstavlja hijerarhijskom strukturom koja opisuje način mapiranja očitanih nizova u rekonstruirani ciljni genom, pri čemu je mapiranje zapravo dobiveno većim dijelom heuristički. U takvoj strukturi osnovna jedinica su očitanja, koja se zatim grupiraju u kontige, koji se grupiraju u skafolde. Kontig je pritom niz uzastopnih očitanja koja se preklapaju, a skafold skupina kontiga, pri čemu se definira njihov poredak, orijentacija i veličine procijepa između njih. Veličina i točnost kontiga i skafolda definira kvalitetu sastavljenog genoma. [Šikić]

SLIKA 2

# OLC paradigma sastavljanja genoma

Dvije glavne metode sastavljanja genoma temeljene na algoritmima nad grafovima su OLC (engl*. Overlap-Layout-Consensus*), odnosno Preklapanje-Razmještaj-Konsenzus metode te metode nad *de Bruijn* grafovima.

SLIKA 3

OLC pristup gradi graf preklapanja te iz njega zaključuje o obliku ciljnih struktura. Takav graf se sastoji od vrhova koji odgovaraju očitanjima i bridova koji povezuju očitanja (odnosno vrhove) ukoliko između njih postoji preklapanje. Putovi u grafu su tada potencijalni kontizi te svaki ima svoju „zrcalnu sliku”, odnosno niz koji mu je reverzni komplement. [Slika xx] prikazuje vrste preklapanja nizova. Preklapanje može ujedno biti i obuhvaćanje, ako je čitav *B* sadržan u *A*. Kod drugih preklapanja prefiks ili sufiks jednog niza je poravnat s prefiksom ili sufiksom drugoga.

SLIKA 4

Nakon izgradnje grafa slijedi njegovo pojednostavljenje. Cilj druge faze je smanjiti broj bridova i vrhova bez smanjenja prostora potencijalnih rješenja. Uklanjaju se tranzitivni bridovi te „mjehurići” i vršna očitanja.

Za generiranje preklapanja korišten je alat *minimap*. [minimap] Algoritam *miniasm* radi s neispravljenim očitanjima, te su i rezultantni kontizi neispravljeni.

# *Miniasm* algoritam

Neka se niz sastoji od znakova abecede te ako je , njegov Watson-Crick komplement. Tada je njegov reverzni komplement.

Promatraju se preklapanja nizova (očitanja) i . Ako se može preslikati na podniz unutar , sadrži , a ako se sufiks od i prefiks od preslikavaju, govorimo o preklapanju koje ujedno i definira brid između tih očitanja. Duljina preklapanja je tada duljina prefiksa koji se ne preklapa s . Graf preklapanja je usmjereni graf gdje je skup očitanja, je skup preklapanja među njima, a je funkcija koja svakom bridu pridružuje njegovu duljinu. Ulazni stupanj čvora je uznačen s , a izlazni s .

## Čišćenje očitanja

Prvi korak je reduciranje greški u očitanjima. Izravnavaju (engl. *trimming*) se očitanja pronalaženjem najdužeg slijeda pokrivenog s tri ili više preklapanja zadane duljine te se odbacuju baze u očitanjima koje su tada izvan pronađenog slijeda. Također se filtriraju kimerna preklapanja.

## Sastavljanje grafa

SLIKA

Gradi se graf preklapanja, pri čemu se preklapanja klasificiraju prema prikazanom algoritmu. Ulazni parametri su: duljine očitanja *l*, početni indeks preklapanja *b*, završni indeks *e*, maksimalna duljina *overhang* područja *o* te maksimalan omjer duljine *overhang* područja i duljine preklapanja *r*.

|  |  |
| --- | --- |
| **KlasificirajPreklapanje(*l, b, e, o, r*):** | |
| *overhang*  min(*b[1], b[2]*) + min(*l[1] – e[1], l[2] – e[2]*) | |
| *maplen* max(*e[1] – b[1], e[2] – b[2]*) | |
| **ako** *overhang* > min(*o, maplen \* r*): | |
|  | **vrati** UNUTARNJE\_PREKLAPANJE |
| **ako** *b[1] b[2]* **i** *l[1] – e[1] l[2] – e[2]:* | |
|  | **vrati** PRVI\_SADRŽAN |
| **ako** *b[1] b[2]* **i** *l[1] – e[1] l[2] – e[2]:* | |
|  | **vrati** DRUGI\_SADRŽAN |
| **ako** *b[1] > b[2]* | |
|  | **vrati** PREKLAPANJE\_PRVI\_PREMA\_DRUGOM |
|  | **vrati** PREKLAPANJE\_DRUGI\_PREMA\_PRVOM |

Unutarnja preklapanja (definirana ovisno o duljini vanjskog (engl. *overhang*) područja) i očitanja sadržana u drugima se ignoriraju, a ostala očitanja se dodaju u graf. [miniasm]

## Čišćenje grafa

### Uklanjanje tranzitivnih bridova

SLIKA

Brid je tranzitivan ako postoji i . Uklanjanje takvih bridova ne utječe na povezanost očitanja u grafu.

### Uklanjanje šiljaka

SLIKA

Čvor je šiljak ako i . Većina šiljaka je uzrokovana pogrešnim očitanjem ili nedostajućim preklapanjima. [miniasm]

### Uklanjanje mjehurića

SLIKA

Mjehurić u grafu je aciklički podgraf s jednim izvorom i jednim ponorom , pri čemu postoje barem dva puta između i . Algoritam pronalaženja mjehurića iz [miniasm] započinje pretragu iz potencijalnog izvora i posjećuje čvor kad posjeti sve njegove ulazne bridove.

Ulazni parametri su: graf početni čvor i maksimalna dubina istraživanja *d.* Algoritam vraća ponor mjehurića ako je pronađen, ili *nil* ako nije.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PronađiMjehurić(*V, E, v0, d*):** | | | | | | | |
| **ako** *deg+(v0) < 2:* **vrati** *nil* | | | | | | | |
| **za** *v V: [v]* | | | | | | *minimalna udaljenost od v0 do v* | |
| *[v0]* | | | | | | | |
| *prazan stog* | | | | | *čvorovi čiji su ulazni bridovi posjećeni* | | |
| PUSH(S, *v0*) | | | | | | | |
|  | | | | | | *broj posjećenih čvorova izvan S* | |
| **dok** S nije prazan: | | | | | | | |
|  | *v* POP(S) | | | | | | |
|  | **za svaki** *v w E:* | | | | | | |
|  |  | **ako** *w = v0:* **vrati** *nil* | | | | | *ciklus* |
|  |  | **ako** *[v] + l(v w) > d:* **vrati** *nil* | | | | |
|  |  | **ako** *[w] = :* | | | | | *još neistraženo* |
|  |  |  | [*w*] *deg*-(*w*) | | | | |
|  |  |  | *p p + 1* | | | | |
|  |  | **ako** *[v] + l(v w) <[w]:* | | | | |
|  |  |  | *[w] [v] + l(v w)* | | | | |
|  |  | [*w*] [*w*] – 1 | | | | |  |
|  |  | **ako** [*w*] *=0:* | | | | |
|  |  |  | **ako** *deg+(w)* 0: | | | | |
|  |  |  |  | PUSH*(S, w)* | | | |
|  |  |  | *p p - 1* | | | | |
|  | **ako** |*S*| = 1 **i** *p = 0:* | | | | | | |
|  |  | **vrati** POP(*S*) | | | | |
| **vrati** *nil* | | | | | | | |

### Sastavljanje unitiga

Neka je put u grafu, duljine *k.* Spajanjem podnizova u čvorovima dobivamo , gdje je podniz određen indeksima i , uključivo, a je operator konkatenacije nizova. U grafu s uklonjenim tranzitivnim bridovima, unitig je put gdje te ili i . Unitig je zapravo maksimalni put na kojem se susjedna očitanja mogu združiti tako da se ne utječe na povezanost u originalnom grafu. [miniasm]

# Rezultati i zaključak

*Miniasm* algoritam ne ispravlja pogrešna očitanja. U najboljem slučaju, pogreška u sastavljanju nizova koji predstavljaju pojedinačne unitige je jednaka pogrešci u samom isčitavanju početnih sekvenci. Teorijski zaključci pokazuju [Li] potencijalna poboljšanja u sastavljanju unitiga iskorištavanjem znanja o preklapanjima u očitanjima. No, takav alat nije definiran u sklopu ove metode.

# Literatura

# Sažetak