SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**SEMINAR**

# Faza konsenzusa u OLC paradigmi sastavljanja genoma

*Ivan Paljak, Ivan Sekulić*
Voditelj: *Mile Šikić*

Zagreb, siječanj 2017.

# SADRŽAJ

# 1. Uvod

Proučavanje strukture DNA u fokusu je znanstvenika od samog otrkića DNA. Precizno čitanje genoma predstavlja velik izazov te su s vremenom razvijani sve brži i jeftiniji uređaji za što bolja očitanja. Danas su uređaji i dovoljno brzi i jeftini, ali se javlja problem kratkih očitanja. Bioinformatika, između ostalog, teži spojiti ta kratka očitanja u jedinstvenu sekvencu (genom).

Prilikom sastavljanja genoma, javlja se nekoliko problema. Prvi je nepreciznost uređaja za sekvenciranje, što zahtjeva višestruka očitanja jednog genoma kako bi se, određenim metodama, mogao ustanoviti stvarni niz. Također, danas se koriste metode temeljene na *shotgun* sekvenciranju cijelog genoma pri čemu nemamo nikakvu informaciju o poretku pojedinih očitanja. Treći otežavajući faktor uspješnog sastavljanja jedinstvene sekvence jest varijabilna duljina očitanja. Uređaji druge generacije, koji treutno prevladavaju, rade očitanja veličine od nekoliko desetaka do par stotina nukleotida. Treća generacija uređaja proizvodi dulja očitanja, od nekoliko tisuća nukleotida, ali imaju velik postotak pogreške – od 15% do čak 40%.

Razvijeno je nekoliko algoritama koji se bave problemom sastavljanja genoma, a najkorišteniji su oni temeljeni na algoritmima nad grafovima. Najčešće se koristi jedna od dviju osnovnih metoda: Preklapanje-Razmještaj-Konsenzus *engl. Overlap-Layout-Consensus, OLC* metode temeljene na grafu preklapanja ili metode temeljene na *de Bruijn* grafovima (Šikić i Domazet-Lošo, 2013). U ovom projektu, bavimo se konsenzus fazom OLC paradigme.

Ovaj dokument organiziran je na sljedeći način: u sljedećem poglavlju opisan je algoritam koji smo koristili za dobivanje konsenzusa. Poglavlje 3 kratko opisuje našu konkretnu implementaciju i karakteristike korištenih računala. U poglavlju 4 dana je usporedba vlastite implementacije s referentnim radom. Poslijednje poglavlje sadrži zaključak cjelokupnog projekta.

# 2. Algoritam Sparc

Cilj ovog projekta bio je implementirati algoritam Sparc (Ye i Ma, 2016) koji se koristi u konsenzus fazi preklapanje-razmještanje-konsenzus (engl. Overlap-Layout-Consensus, OLC) pristupa. Sparc je algoritam za konsenzus fazu OLC pristupa, temeljen na *k-mer/de Bruijn* (Hannenhalli et al., 1996) grafovima.

## 2.1. Ulaz u algoritam

Algoritam koristi izlaz iz faze razmještanja OLC pristupa (u .fasta formatu) te sva početna očitanja genoma (u .fastq formatu). Očitanja su mapirana na kontigu iz faze razmještanja i pohranjena u datoteku formata .sam, ukratko opisanom u nastavku.

Kako bi uspješno izgradili graf i proveli algoritam, potrebno je znati gdje se pojedina očitanja mapiraju na osnovnu kontigu. Te informacije dobivano iz .sam datoteke koju smo generirali alatom *GraphMap* (Sović et al., 2016). Nama najvažnije informacije u datoteci jesu:

**POS** pozicija na osnovnoj kontizi na kojoj počinje mapiranje pojedinog očitanja

**CIGAR** operacije obavljene nad očitanjem kako bi se dobilo mapiranje (dodavanje, brisanje, pomicanje, itd.)

**SEQ** očitana sekvenca u prvotnom obliku (bez obavljenih CIGAR operacija)

**QUAL** kvaliteta očitanja (iz .fastq datoteke očitanja)

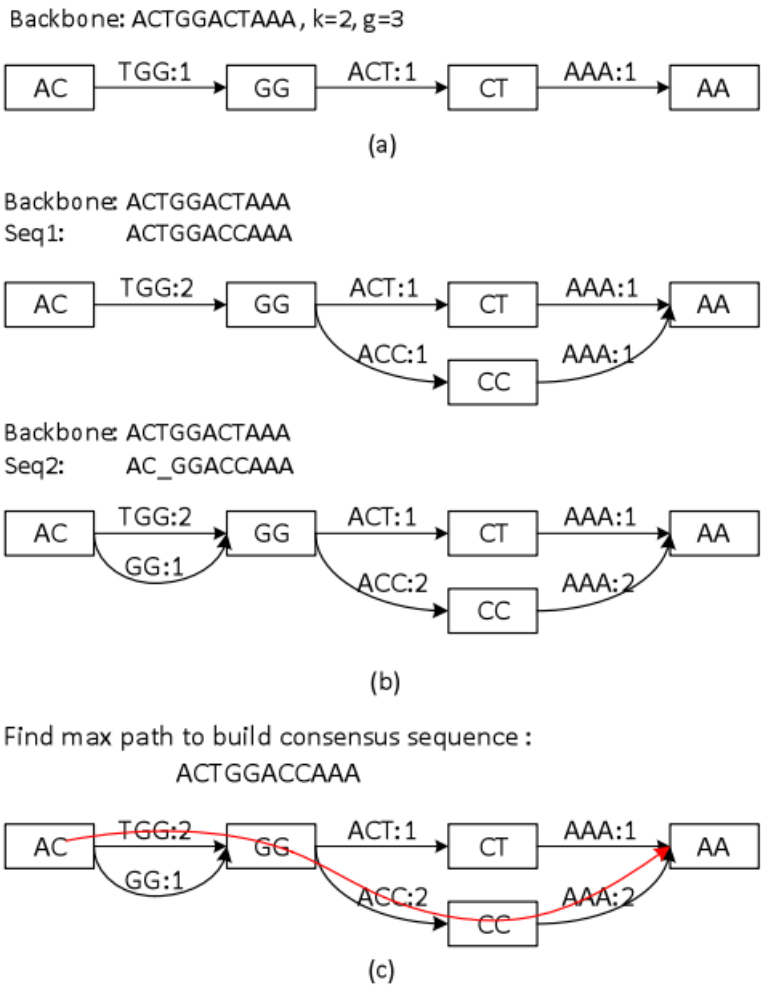
## 2.2. Opis algoritma

Temeljni dio algoritma je izgradnja grafa $k$-mera. Prvi korak u izgradnji spomenutog grafa jest pretvorba referentong genoma (engl. *backbone*) u sparse lanac $k$-mera kao što je prikazano u $(a)$ dijelu slike. Gustoću grafa korigiramo hiperparametrom $g$, dok

je veličina $k$-mera određena parametrom $k$. U čvorovima lanca nalaze se $k$-meri dok se na vezama nalaze dijelovi genoma koji ih povezuju.

Uz referentni genom imamo i niz očitanja koja se djelomično ili potpuno mapiraju na taj genom, a opis jednog takvog očitanja vidljiv je u prethodnom odlomku. Koristeći očitanu sekvencu [SEQ] i niz operacija nad tom sekvencom [CIGAR] gradimo mapiranje na referentni genom koje u odnosu na njega može imati neke elemente zamijenjene, obrisane ili dodane. Od tog mapiranja gradimo lanac k-mera sličan lancu referentnog genoma uz dodatak što svaki puta kada novoizgrađeni čvor već postoji u stablu k-mera, prethodni čvor ćemo, uz odgovarajući prijelaz, samo spojiti na njega. Ovaj dio algoritma lijepo je vidljiv u (b) i (c) dijelovima slike 2.1.

Primijetimo sada da je graf k-mera usmjeren i acikličan (engl. *DAG - Directed Acyclic Graph*), a skup svih lanaca tog grafa nam predstavlja prostor rješenja. Kako bismo izabrali najizgledniji lanac, uz oznaku prijelaza svake veze pamtit ćemo i neku mjeru izglednosti da je baš ta veza dio optimalnog lanca. Ovu mjeru možemo modelirati na razne načine, a u našoj implementaciji je ona jednaka broju očitanja čiji lanci prolaze tom vezom. Izglednost nekog lanca modelirana je sumom težina svih veza tog lanca. Dakle, problem smo sveli na traženje najduljeg puta u DAG-u. Primijetimo da nam usmjerenost i acikličnost dopuštaju nešto efikasniji pronalazak najduljeg puta u grafu. Naime, svaki DAG ima barem jedan topološki poredak čvorova (engl. *Topological sort*) pa se problem najdužeg puta svodi na jednostavno dinamičko programiranje. Neka $dist_i$ označava najdulji put nekog lanca koji završava u čvoru koji se u topološkom poretku nalazi na poziciji $i$. Tada vrijedi relacija $dist_i = max\{dist_j + w_{ji}\}$ za svaki čvor koji se u topološkom poretku nalazi na indeksu $j$ i direktno je povezan s čvorom na indeksu $i$. Postojanje topološkog poretka nam garantira da ćemo prije $dist_i$ već imati izračunate sve vrijednosti $dist_j$ pa najdulji put nalazimo u složenosti $O(V + E)$.




**Slika 2.1:** Izgradnja grafa (Ye i Ma, 2016).

# 3. Implementacija

## 3.1. Organizacija koda

Cjelokupni algoritam implementiran je u programskom jeziku C++11. Implementacija je podijeljena u dva osnovna dijela. Prvi modul *reading.cpp* učitava sve potrebne podatke (backbone i očitanja u .sam formatu) te primjenjuje CIGAR operacije na pojedina očitanja. Time generiramo datoteku vlastitog formata. U prvoj liniji zapisan je backbone, a svake sljedeće tri linije predstavljaju jedno očitanje na način:

1. očitanje na koje su primijenjene CIGAR operacije,
2. kvaliteta očitanja na koju su primijenjene CIGAR operacije,
3. pozicija u backbone-u na koju se mapira očitanje.

Drugi modul *sparc.cpp* direktna je implementacija algoritma Sparc prema ideji objašnjenoj u odgovarajućem poglavlju.

## 3.2. Kofiguracija korištenog računala

Računalo korišteno pri pokretanju cjelokupnog pipelinea ima sljedeću konfiguraciju:

**OS** Linux 14.04.1-Ubuntu x86_64

**Procesor** Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz (CPUs: 12)

**RAM** 32Gib @ 2133 MHz

# 4. Evaluacija

Testni podatci dobiveni su na kolegiju[1]. Evaluacija je obavljena na genomima dviju bakterija: escherichia coli te fuge lambda. Za svaku bakteriju dostupna su očitanja, genom nakon faze razmještanja te referentni genom s kojim uspoređujemo performanse svoga algoritma. Cilj je bio generirati genom konsenzus fazom kako bi njegovo preklapanje s referentnim bilo što veće od preklapanja genoma nakon faze razmještanja s referentnim.

## 4.1. DnaDiff iz MUMmer paketa

*MUMmer* (Kurtz et al., 2004) je široko korišten paket otvorenog koda (engl. *Open Source*) za razne podzadatke bioinformatike. Implementira razne module, a mi smo koristili *dnadiff* – modul dizajniran za evaluaciju slijedova dvaju vrlo sličnih genoma. Pruža detaljne informacije o razlikama između dvaju genoma, ali generira i high-level datoteku s kvantificiranim razlikama. Svoj algoritam vrednovali smo prema *AvgIdentity* polju u navedenoj datoteci (out.report). Broj predstavlja prosječno poklapanje referentnog genoma s našim. Postoje podatci za 1-na-1 mapiranje (gdje su ponavljanja zanemarena) te M-na-M mapiranje. Uglavnom su oba broja jednaka, pa stoga nije posvećeno previše pažnje na odabir određenoga. U svim rezultatima u nastavku nalazi se *AvgIdentity* polje iz 1-na-1 mapiranja.

[1] fer.unizg.hr/predmet/bio

**Tablica 4.1:** Usporedba vlastitog rješenja (SpaCRO) sa sekvencom iz faze razmještanja (default) i referentnim radom (Sparc).

| | lambda | ecoli |
|---|---|---|
| default | 86.16 | 88.57 |
| Sparc | **95.41** | **98.17** |
| SpaCRO | 90.97 | 95.69 |

**Tablica 4.2:** Potrošnja memorije i vrijeme izvođenja našeg algoritma na testnim skupovima podataka.

| | lambda | ecoli |
|---|---|---|
| mem [MB] | 30 | 2464 |
| time [s] | 0.45 | 62.8 |

## 4.2. Usporedba vlastitog rješenja s referentnim radom

Tablica 4.1 prikazuje naše rezultate u usporedbi s referentnim radom. Također, prikazana je mjera preklapanja sekvence iz faze razmještanja s konačnom, referentnom sekvencom.

Možemo uočiti kako je postignuto značajno poboljšanje sekvence genoma, ali rezultati dobiveni referentnim algoritmom su ipak bolji. Razlog nepostizanja visokih rezultata je najvjerojatnije zbog malih razlika u implementaciji algoritma. Također, originalni *Sparc* provodi algoritam u nekoliko iteracija. Na takav način svakom iteracijom malo raste točnost generirane sekvence (zasićenje je već nakon 3-5 iteracija). Mi nismo uočili poboljšanje performansi s povećanjem iteracija.

Naš algoritam najbolje rezultate postiže za hiperparametre $k = 3$ i $g = 4$. Originalan algoritam bolje radi s manjim vrijednostima. Rezultati u tablici 4.1 su za $k = 1$ te $g = 1$.

## 4.3. Memorija i vrijeme izvođenja

Kao dio projekta, potrebno je izmjeriti potrošnju memorije i vrijeme izvođenja našega algoritma te zadovoljiti određena ograničenja. Mjerenje je obavljeno pomoću alata cgmemtime[2], a rezultati su prikazani u tablici 4.2. I utrošena memorija i vrijeme izvođenja zadovoljavaju zadana ograničenja.

[2] github.com/isovic/cgmemtime

# 5. Zaključak

U okviru ovoga projekta upoznali smo se s osnovama bioinformatike. Proučili smo OLC paradigmu sastavljanja genoma te implementirali algoritam temeljen nad grafovima za konsenzus fazu paradigme. Uočili smo određene probleme sastavljanja genoma te se upoznali s postupcima i najčešće korištenim alatima u bioinformatici.

Impementiran je algoritam *Sparc* te je napravljena analiza rezultata. Postignuto je značajno poboljšanje u odnosu na početnu sekvencu, ali nisu dostignuti rezultati originalnog algoritma. Kompletan kod, upute za instalaciju i pokretanje mogu se pronaći na github.com/ipaljak/bioinfo.

# 6. Literatura

Sridhar Hannenhalli, William Feldman, Herbert F Lewis, Steven S Skiena, i Pavel A Pevzner. Positional sequencing by hybridization. *Computer applications in the biosciences: CABIOS*, 12(1):19–24, 1996.

Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, i Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):1, 2004.

Ivan Sović, Mile Šikić, Andreas Wilm, Shannon Nicole Fenlon, Swaine Chen, i Niranjan Nagarajan. Fast and sensitive mapping of nanopore sequencing reads with graphmap. *Nature communications*, 7, 2016.

Chengxi Ye i Zhanshan Sam Ma. Sparc: a sparsity-based consensus algorithm for long erroneous sequencing reads. *PeerJ*, 4:e2016, 2016.

Mile Šikić i Mirjana Domazet-Lošo. *Bioinformatika*. Fakultet Elektrotehnike i Računarstva, Sveučilište u Zagrebu, 2013.

# 7. Sažetak

Ovaj projekt napravljen je za kolegij Bioinformatika na FER-u. Implementiran je algoritam *Sparc* za generiranje konsenzusa u OLC paradigmi sastavljanja genoma. Dobiveni su rezultati te uspoređeni s referentnim radom. Kompletan kod, upute za instalaciju i pokretanje mogu se pronaći na github.com/ipaljak/bioinfo.