SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

BIOINFORMATIKA

Pronalazak mutacija pomoću treće generacije sekvenciranja

Dorian Ivanković, Luka Jurić, Šime Pavlić

Zagreb, siječanj, 2019.

Sadržaj

1. Uvod

Sekvenciranje genoma područje je čiji je razvoj potaknut 1970-tih radom Fredericka Sangera, te se strijelovito počelo razvijati 90-ih godina prošlog stoljeća. U početku je sekvenciranje bilo visoke točnosti, ali jako skupo. Tijekom godina razvile su se metode za jeftino sekvenciranje genoma, koje su nažalost cijenu platile na točnosti. Ovaj rad temelji se na pronalasku mutacija na referentnom genomu uz pomoć skupa očitanja dobivenog sekvenciranjem mutiranog genoma. Sekvenciranje je dobiveno metodama treće generacije i ima pogrešku od oko 15%. Bitni algoritmi koji se koriste su 'seed-and-extend' koji koristi minimizirane k-mere ulaznih nizova kako bi našao u kojem se dijelu referentnog genoma nalazi pojedino očitanje mutiranog genoma. Nakon toga koristi se Hirschbergov algoritam za nalaženje optimalnog poravnanja između referentnog genoma i mutiranih očitanja, te nalaženje mutacija na navedenim dijelovima. U prvom dijelu biti će detaljnije opisani algoritmi i korišteni postupak. U drugom dijelu navedeni postupak prikazati ćemo i objasniti na jednostavnom primjeru. Nakon demonstracije navesti ćemo dobivene rezultate, odnosno točnost, vrijeme izvođenja te utrošak memorije. Navedene rezultate zatim ćemo usporediti s izvornom implementacijom.

2. Opis algoritma

2.1 Konfiguracija i ulaz u program

U svrhu demonstriranja postupka i rezultata ovog rada razvijen je program za traženje mutacija u programskom jeziku C++ i programskom okružju CLion. Program na ulaz dobiva referentni genom i skup očitanja dobiven sekvenciranjem mutiranog genoma. Obje ulazne datoteke su u FAST-a formatu. Kao prvi korak datoteke učitamo u program, referentni genom spremimo kao jedan znakovni niz, a mutirana očitanja kao listu znakovnih nizova. Nakon učitanja podataka, želimo pronaći na koje dijelove referentnog genoma možemo mapirati mutirana očitanja. Za to koristimo algoritam *seed and extend*.

2.2 Seed and extend

Za početak *seed and extend* algoritma trebaju nam svi k-meri, odnosno podnizovi duljine k referentnog genoma, te svakog mutiranog očitanja. Uz k-mere moramo naći i inverzne k-mere jer ne znamo u kojem smjeru su sekvencirana pojedina očitanja. Pošto bi spremanje svih k-mera i inverznih k-mera zauzimalo previše memorije, te bi i algoritam s tolikom količinom k-mera bio znatno sporiji, odabiremo samo one k-mere koji su nam najbitniji, odnosno minimizere. Minimizere možemo prikazati u obliku (*w*,*k*). Ovaj oblik govori nam da tražimo minimizer duljine *k* u okolici w susjednih k-mera. Kod minimizera bitna stavka je da ako dva niza imaju jednaku dovoljno dugačku sekvencu oni moraju odabrati iste minimizere na toj sekvenci. Naša metoda svakom znaku A, C, G ili T pridaje neku brojevnu vrijednost. Od *w* susjednih k-mera tada kao minimizer odabire onaj k-mer čiji zbroj vrijednosti znakova je najmanji. Na taj način, jednaki nizovi uvijek će odabirati jednake minimizere. Detaljnije o minimizerima može se pronaći u radu [1].

2.2.1 Longest increasing subsequence

Nakon što smo pronašli minimizere referentnog genoma i mutiranog očitanja, potrebno je pronaći najvjerojatnije područje na referentnom genomu na koje se mutirano očitanje mapira.To radimo tako da pronađemo indekse u referentnom

genomu na kojima se k-meri iz referentnog genoma podudaraju s k-merima iz mutiranog očitanja. Kada dobijemo indekse podudaranja, želimo pronaći područje najdulje sekvence podudaranja, a da je udaljenost između podudaranja susjedna dva k-mera relativno mala (<500). To se jednostavno nalazi tako da podudaranja k-merova sortiramo po poziciji te iteriramo uzlazno i bliske k-mere stavljamo u istu grupu. Dobivena grupa s najviše pogodaka, odnosno prvi i zadnji indeksi u grupi pokazuju nam na najbolje mapirano područje. Sada je nađeno područje na referentnom genomu potrebno poravnati s mutiranim očitanjem. Za to koristimo Hirschbergov dinamički algoritam.

2.3 Hirschbergov algoritam i pronalazak mutacija

Prednost Hirschbergovog algoritma je u linearnoj ovisnosti o zauzeću memorije. Mnogi drugi algoritmi poravnanja imaju kvadratnu ovisnost o memoriji, te kod dugih očitanih sekvenci mogu biti jako zahtjevni za RAM. Glavna ideja Hirschbergovog algoritma je da je poravnanje dva niza *s* i *t* jednako poravnanju prve polovice niza *s* s nizom *t* i reverznog poravnanja druge polovice niza *s* s nizom *t*. Cilj je pronaći indeks *k* niza *t* za koji je zbroj dva poravnanja maksimalan. Praktično algoritam dijeli Needleman-Wunsch matricu na dvije polovice te radi na principu spajanja polovica putova rekonstrukcije u jednu cjelinu. Nakon riješenja obje matrice, u zadnjem redu svake matrice odabiru se oni susjedni elementi matrice čiji zbroj rezultata je najveći. Kada odredimo te susjedne elemente znamo da se oni nalaze na putu optimalnog poravnanja i onda možemo rekurzivno dalje odrediti sve ostale dijelove poravnanja. Tim postupkom dobijemo i najvjerojatnije mutacije koje su se dogodile između tog dijela referentnog genoma i mutiranog očitanja. Detaljniji opis Hirschbergovog algoritma može se pronaći u radu [4].

Nakon što smo našli najvjerojatnije mutacije između dijelova referentnog genoma i svakog mutiranog očitanja, dovoljno je provjeriti koje se mutacije najčešće pojavljuju na pojedinom indeksu referentnog genoma, te je najvjerojatnije da su se te mutacije zaista i dogodile.

2.4 Izlaz programa i evaluacija rješenja

Program kao izlaz daje datoteku u kojoj su u CSV formatu navedene sve pronađene mutacije. Oblik CSV formata je slijedeći:

Mutacija	**Linija u CSV datoteci**		
Substitucija	X	Pozicija u referenci na kojoj se dogodila substitucija	Zamjenska nukleotidna baza
Umetanje	I	Pozicija u referenci prije koje se dogodilo umetanje	Umetnuta nukleotidna baza
Brisanje	D	Pozicija u referenci na kojoj se dogodilo brisanje	-

Evaluacija rezultata programa provodi se pomoću Jaccardovog indeksa. Jaccardov indeks uspoređuje članove dva seta i provjerava koji članovi si isti, a koji različiti. Formula računanja Jaccardovog indeksa je slijedeća:

$$J(X,Y) = |X \cap Y| / |X \cup Y|$$

3. Demonstracija postupka

Okvirnu demonstraciju postupka opisanog u prethodnom poglavlju prikazati ćemo na znakovnom nizu *s* koji predstavlja referentni genom, te skupu znakovnih nizova *t[]* koji će predstavljati mutirana očitanja.

Napomena: Zadnji korak u algoritmu simuliran je, jer da bi programska implementacija dobro radila, veličina referentnog genoma treba biti viša od nekog minimuma te pokrivenost svake pozicije u genomu mutiranim očitanjima treba biti velika. Takav primjer bio bi prevelik za ovakvu jednostavnu demonstraciju.

Umetanje 4, G

Supstitucija 13, G

Umetanje, 17, A

s = GTCTAGGATTATGAGTC

t[] = GTCTGAGGATT

TTATGAGTA

ATTATG

TGAGGATT

CTGAGGATT

ATTATGAGTA

ATTATGAG

TTATGAG

TCTGAGGAT

ATTATG

Slijedeći korak je da za referentni genom i mutirana očitanja nađemo k-mere i inverzne k-mere te ih minimiziramo. Tražimo 4-mere te od njih (4,4)-minimizere od 4 susjedna 4-mera. Zbog količine podataka prikazati ćemo samo dobivene minimizirane k-mere referentnog genoma i prvog mutiranog očitanja.

Minimizeri za referentni genom:

AGTC, TAGG, TGAG, GAGT, AATC, TCAT, TCCT, ATAA, ATCC, TAAT, CATA

Minimizeri za prvo mutirano očitanje:

TGAG, AATC, TCCT, ATCC, GAGG

Kada smo našli minimizere možemo, kako smo gore opisali, pomoću grupa indeksa podudaranja bliskih minimizera, dobiti mapirano područje.

U slučaju prvog mutiranog očitanja, kao mapirano područje dobivamo pozicije od 4 do 9 u referentnom genomu, te od 5 do 10 u mutiranom očitanju. Vidimo da je algoritam našao dobro područje jer ono je u oba niza jednako AGGATT.

Sva mapirana područja referentnog genoma i odgovarajućih očitanja sada provlačimo kroz Hirschbergov algoritam, kako bi našli optimalno poravnanje i moguće mutacije između genoma i očitanja. Za gornji primjer poravnanja nizova AGGATT i AGGATT ne nalazimo ni jednu potencijalnu mutaciju, što je i ispravno jer su nizovi potpuno jednaki. Potencijalne mutacije na pojedinim indeksima pamtimo, te na kraju odlučujemo o mutaciji metodom glasanja. Odabrana je najčešća mutacija nađena na nekom indeksu. U našem primjeru nakon što smo poravnali sva mapirana područja pojavljuju se slijedeće najčešće mutacije:

Umetanje, pozicija 4, ‘G’

Supstitucija, pozicija 13, ‘G’

Umetanje, pozicija 17, ‘A’

Na ovako jednostavnom primjeru lako možemo vidjeti da su ove mutacije ispravne usporedbom referentnog genoma i mutiranih očitanja.

4. Rezultati

Programska implementacija testirana je na dva referentna genoma, *lambda* i genom bakterije *escherichia coli*. Dobivene rezultate zatim smo usporedili s rezultatima izvorne implementacije. Kod testiranja na genomu *lambde*, program na memorijski najzahtjevnijem dijelu zauzima oko 13MB memorije. Vrijeme izvođenja programa paraleliziranog na 8 dretvi iznosi oko 3 minute. Na genomu *lambde* dobiven je Jaccardov indeks od 0.729729729730 što zaključujemo da je vrlo dobro jer očitanja tog genoma imaju vrlo visoku razinu pogreške, na nekim mjestima i do 60%. Zaključak o uspješnosti rezultata možemo dobiti i usporedbom s Jaccardovim indeksom izvorne implementacije koji iznosi 0.453846153846. Na genomu *e.coli,* program na memorijski najzahtjevnijem dijelu zauzima oko 800 MB memorije, što je vrlo malo sudeći po količini podataka koje obrađuje. Vrijeme izvođenja isnosi oko 5 sati. Dobiven je Jaccardov indeks od 0.847675008552. Za usporedbu, referentna implementacija dobila je Jaccardov indeks jednak 0.824952583615. Po prikazanim rezultatima vidimo da je točnost algoritma vrlo visoka što nam pokazuje uspješnost implementacije navedenih algoritama.

5. Zaključak

Sekvenciranje genoma i lociranje mutacija bitan je dio Bioinformatike i to područje još uvijek je u velikom razvoju. U ovom radu osvrnuli smo se na neke poznate algoritme koji nam pomažu u lociranju mutacija, te smo napravili implementaciju u programskom jeziku C++ kako bi isto i demonstrirali. Iz dobivenih rezultata i usporedbom s izvornom implementacijom možemo vidjeli smo da postupci koje smo koristili daju zadovoljavajuće rezultate. Memorijska složenost programa vrlo je malena zbog korištenja Hirschbergovog algoritma za poravnanje nizova. Hirschbergov algoritam s druge strane negativno utječe na vremensku složenost, no s obzirom na razinu kojom smanji memoriju zaključili smo da je korištenjeg istog isplativo. Točnost programa vrlo je visoka i u slučajevima s visokom pogreškom metoda sekvenciranja te nam pokazuje da korišteni algoritmi vrlo dobro rade na detekciji mutacija.

6. Sažetak

Ovaj rad opisuje problematiku traženja mutacija koje su se dogodile na genomu pomoću referentnog genoma i skupa očitanja dobivenog sekvenciranjem mutiranog genoma. U postupku pronalaženja mutacija koristili smo neke poznate algoritme u području Bioinformatike kao 'Seed-and-Extend' metodu mapiranja nizova pomoću minimiziranih k-mera, te Hirschbergov dinamički algoritam za pronalaženje optimalnog poravnanja između dva niza. Postupak lociranja mutacija i korištene algoritme objasnili smo zatim na jednostavnom primjeru. Implementacija programa testirana je na genomu bakterije escherichije coli te genomu lambda, te su za oba genoma prikazani dobiveni rezultati te njihova usporedba s izvornom implementacijom.

7. Literatura

[1] Michael Roberts, Wayne Hayes, Brian R. Hunt, Stephen M. Mount, James A. Yorke. Reducing storage requirements for biological sequence comparison, 15 Srpanj 2004.

[2] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences, 19 Ožujak 2016.

[3] Mile Šikić, Mirjana Domazet-Lošo. Skripta za predmet Bioinformatika, Prosinac 2013.

[4] Dan Hirschberg. A linear space algorithm for computing maximal common subsequences, 1975.

[5] Jaccard Index / Similarity Coefficient, December 2, 2016. https://www.statisticshowto.datasciencecentral.com/jaccard-index/.

[6] Longest increasing subsequence, July 29, 2018.

https://en.wikipedia.org/wiki/Longest_increasing_subsequence