SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

LISA - Alat za poravnanje DNA očitanja

Filip Pavetić, Goran Žužić

Zagreb, svibanj 2013.

Ovaj rad izrađen je na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu, na Zavodu za elektroničke sustave i obradbu informacija pod vodstvom doc. dr. sc. Mile Šikića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2012. / 2013.

SADRŽAJ

1. Uvod

Ovaj rad opisuje sustav za mapiranje DNA sekvenci na referentni genom. Rad je podijeljen na 4 dijela:

- u prvom dijelu ćemo napraviti kratki uvod u probleme bioinformatike i pozadinu problema kojeg sustav rješava
- u drugom dijelu se bavimo modelom, algoritmom i implementacijom nad kojim je sustav ostvaren
- nakon toga prezentiramo rezultate testiranja i usporedba s najpoznatijim analognim alatima
- naposljetku dajemo kratki zaključak i smjernice za daljnji rad

Sustav je razvijen u sklopu natjecanja identifikacija organizama iz toka DNA sekvenci kojeg je objavila tvrtka *InnoCentive* pod sponzorstvom vlade Sjedinjenih Američkih Država i koje je u tijeku [1].

1.1. Bioinformatika

Bioinformatika je u posljednjih nekoliko godina u centru ogromne pažnje i entuzijazma zbog napredaka u razumijevanju živih bića i brzom razvoju tehnologije pomoću kojih ih možemo proučavati. Tijekom 20. stoljeća se bioinformatika koristila u svrhe proučavanja nasljednih svojstava, evolucije između vrsta te identifikacije patogenih bakterija, dok se u bliskoj budućnosti vjeruje kako će pomoći u proizvodnji boljih lijekova koji bi bili posebno prilagođeni zaraženoj osobi.

Molekularna biologija i biokemija su nam otkrili kako najveću funkcionalnu ulogu u gotovo svim biološkim procesima živih organizama imaju molekule zvane *proteini*, linearni nizovi aminokiselina spojenih peptidnim vezama. Njihova uloga u organizmima seže od katalizacijskih procesa (npr. kod metabolizma), signalizacije i adhezije stanica sve do imunološkog sustava. Proteini su konstruirani zasebno u svakoj stanici od svojih

[1] više informacija o samom natjecanju možete pronaći na *http://tinyurl.com/dna-dtra*

gradivnih elemenata (aminokiselina) prema uputstvu zapisanim u masivnoj staničnoj molekuli deoksiribonukleinske kiseline (DNA).

Dominantno područje bioinformatike se bavi sekvenciranjem (čitanjem) i analizom DNA, koja je po svojoj strukturi dugi linearni niz sastavljen od otprilike 3 milijarde[2] povezanih parova baza nukleinskih kiselina. Masivnost tih brojeva stvara potrebu za razvojem računalnih alata bez kojih bilo kakva obrada postaje nemoguća.

Nedavni je razvoj moderne tehnologije doveo do strmoglavog pada cijene sekvenciranja molekule DNA do te razine da je postalo jasno kako će u bliskoj budućnosti najsporiji i najteži dio bioinformatike biti u konstrukciji efikasnih algoritama koji brzo i pouzdano obrađuju rastuću količinu informacija na njihovom raspolaganju.

1.2. DNA i mutacije

Kao što je spomenuto, DNA modeliramo kao linearni niz povezanih parova baza. U DNA se pojavljuju točno četiri različite baze i njih označavamo slovima *A, G, T* i *C*[3]. Tako DNA reprezentiramo dugim nizom slova nad gornjom četvoroslovnom abecedom, primjerice:

```
... AGTGAGGAAAAAAAAAGGTCAATGCAGCACTTGAGCCAACATTGTAGAT
    GTTGTACTGCAAGGTCAGGTCTCGCCCCTCCACGGCGTATCTGTTCAG
    CAGTGACTTGGAGGCAAGAAAATCAAACCCGTGATCGATGGTACCGAGC ...
```

Kroz vrijeme se na molekuli DNA javljaju razne mutacije. Okvirno poznavanje mogućih mutacija je nužno za bolje razumijevanje izazova koji se javljaju u bioinformatici. Biolozi su izolirali nekoliko dominantnih vrsta:

1. Supstitucija - zamjena jedne baze drugom. Najčešća mutacija. Uzrokovana greškama u replikaciji i kemikalijama.

   A C G T T G A C
   A C G A T G A C

   Slika 1.1: Primjer supstitucije

2. Ispuštanje i umetanje - iz DNA se ukloni ili doda slijed baza. Ova mutacija je vrlo štetna i obično rezultira gubitkom funkcionalnosti dijela gena.

[2] kod ljudi

[3] skraćeno od adenin, guanin, timin i citozin, vrsta nukleinskih kiselina koje se pojavljuju

A C G T T G A C

A C G A C

Slika 1.2: Primjer ispuštanja

3. Udvostručavanje - ponavljanje dijela sekvence dva ili više puta zaredom. Ova mutacija je iznimno rijetka, ali se smatra da je unatoč tomu imala veliku ulogu u povećanju genetskog koda živih bića.

A C G T T G A C

A C G T T G A T T G A C

Slika 1.3: Primjer udvostručavanja

4. Inverzija - okretanje dijela sekvence. Iznimno rijetka mutacija koja se najčešće zanemaruje pri analizi.

A C T C A A G G

A C A A C T G G

Slika 1.4: Primjer inverzije

1.3. Sekvenciranje DNA

Prvi pokušaj sekvenciranja ljudskog DNA počinje 1990. američkim *Human Genome Project*-om koji je trajao 15 godina i uspješno identificirao većinu ljudskog genoma. Ukupna cijena projekta je procijenjena na 3.000.000.000$. [2] Od tada cijena sekvenciranja DNA strmoglavo pada i očekuje se će u sljedećih nekoliko godina dostići 1.000$ za cjelokupni ljudski genom.

Sekvenciranje DNA se odvija u nekoliko faza: (1) prikupljanje kratkih očitanja, (2) poravnavanje s referentnim genomom, (3) izlistavanje varijanti i (4) filtriranje i označavanje. [8]

Prikupljanje kratkih očitanja je biološki dio sekvenciranja tijekom kojeg se DNA višestruko replicira, slučajno lomi na dijelove duljine 100-3000 baza te se potom očitava složenim metodama. Te metode su podložne pogreškama uslijed krivog

očitanja dijela sekvenci, gubljenja informacija i dvosmislenosti podataka koji se javljaju tijekom loma.

Poravnavanje s referentnim genomom je proces mapiranja očitanih fragmenata na referentni genom. Uobičajeni volumen podataka od otprilike desetke milijuna očitanja i tri milijarde startnih pozicija čini ovaj posao vrlo algoritamski zahtjevnim. Još kritičnijim ovaj korak čini svojstvo da se sve greške na ovom koraku propagiraju i ruše kvalitetu svim preostalim dijelovima sekvenciranja.

Izlistavanje varijanti je korak u kojem se uspoređuju poravnata očitanja s referentnim genomom. Pronađene mutacije mogu biti uzročnici nasljednih bolesti ili mogu jednostavno biti genski šum bez funkcionalnih svojstava.

Filtriranje i označavanje procjenjuje koja od potencijalnih razlika između interesantnih gena pridonosi patološkom procesu kojeg proučavamo. Filtriraju se mutacije bez funkcionalnih značajki te se prepoznaju česte populacijske mutacije.

1.4. Obrazloženje teme

Izgrađeni sustav se bavi problematikom *poravnavanja očitanja s referentnim genomom*. Specifično, potrebno je za svaki od nekoliko desetaka milijuna očitanja koja su skupljena u FASTQ datoteku mapirati na poziciju u referentnom genomu. Problem su pritom gore opisane mutacije, pogreške u kratkim očitanjima, dvosmisleni podaci, regije niske složenosti te repetitivne regije koje se mogu mapirati na više mjesta u referentnom genomu.

Ovaj sustav se koristi kao prvi korak prilikom rješavanja problema identifikacije organizama iz toka DNA sekvenci koji se pojavio na natjecanju *Innocentive*. Autori ovog rada sudjeluju na tome natjecanju u sklopu tima Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

2. Pristup

2.1. Pregled postojećih rješenja

Ovo poglavlje počinjemo kratkih pregledom trendova u razvoju mappera te kasnije dajemo detaljan opis našeg pristupa. Jedan od prvih algoritama za poravnanje očitanja je algoritam globalnog poravnanja pod nazivom Smith-Waterman[11]. Rješenje dobiveno tim algoritmom može se smatrati veoma točnim zbog toga što ima statistički opravdano značenje, međutim zbog velike vremenske složenosti algoritma nije praktično za korištenje.

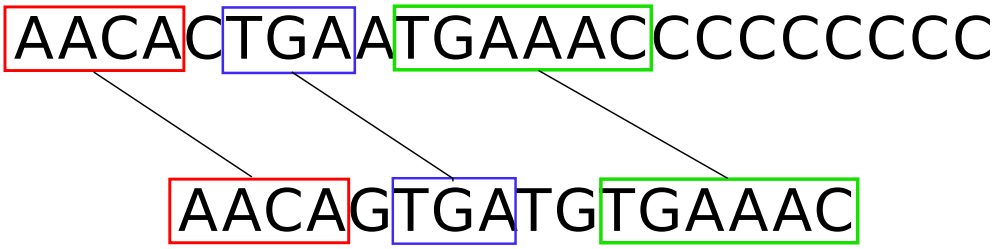
Većina boljih modernih alata za mapiranje - *mappera* (poput SNAP-a[13] i SeqAlto-a[9]) koriste *seed-and-extend* pristup. U tom pristupu prvo se od referentnog genoma gradi indeks koji mapira niz od svakih uzastopnih k-znakova genoma na sve pozicije unutar genoma na kojima se taj podniz nalazi. Ukupan broj takvih različitih podnizova je 4^k, što je i prva praktična opaska, jer za $k \leq 32$ možemo takav podniz zakodirati u 64-bitni cjelobrojni tip podataka. Očitanja se obrađuju na način da se ponovno promatraju podnizi uzastopnih znakova duljine k te se vrši upit u prethodno izgrađeni indeks. Pozicije dobivene takvim upitom daju kandidatne pozicije u čijoj se okolini nalazi potencijalno poravnanje očitanja na genom. Razlike između mappera ovog tipa su primarno u načinu izgradnje indeksa i načinu ocjene kvalitete poravnanja na nekoj kandidatnoj poziciji. Tako primjerice SNAP koristi tablicu raspršenog adresiranja kao strukturu podataka nad kojom je izgrađen indeks, dok SeqAlto koristi sortirano polje. Za ocjenu kvalitete SNAP koristi *edit-distance* metriku, dok SeqAlto koristi općenitiju *Needleman-Wunsch*[1] metriku. Oba pristupa daju značajno poboljšanje efikasnosti u odnosu na pristup spomenut na početku ove sekcije, međutim ti su mapperi i dalje nepraktični budući da kroz vrijeme strojevi za izradu očitanja napreduju i duljine očitanja postaju sve veće čime pretpostavke koje osiguravaju efikasnost ovih alata postaju prejake (glavni razlog je što u njihovim implementacijama izračun tih metrika ima vremensku složenost linearno proporcionalno s duljinama očitanja).

Naš mapper indeks izgrađuje po uzoru na SeqAlto, međutim uvodimo novu metriku

ocjene kvalitete koja za red složenosti poboljšava postojeće algoritme bez gubitka točnosti što ćemo potkrijepiti eksperimentima u kasnijim poglavljima. Metrika kvalitete bazira se na traženju najduljeg uzlaznog podniza[1]

2.2. Opis pristupa

Glavna inspiracija prema našem algoritmu proizlazi iz nekoliko svojstava. Prvo, za malu količinu grešaka (koje predstavljaju realan slučaj, očitanja s velikim postotkom greške nisu korisna za ikakve zaključke) unutar očitanja postojat će mnogo nizova uzastopnih znakova koji nisu dotaknuti greškom. Mi želimo iskoristiti tu činjenicu pa dizajniramo algoritam koji nastoji poravnati dijelove netaknute greškom, a one koji je sadrže ignorira. Pokazuje se da je za veće duljine očitanja ovaj pristup iznimno uspješan[2]



Slika 2.1: Primjer poravnanja - dobro poravnanje ima jako puno preklapajućih dijelova

2.2.1. Indeks

U pregledu poglavlja spomenuli smo da za izgradnju indeksa pratimo smjernice SeqAlto mappera. Svaki niz i od $k = 20$ uzastopnih znakova na poziciji p_i unutar genoma kodiramo jednim cijelim brojem h_i koji predstavlja hash bez kolizije. Indeks nam čini polje uređenih parova (h_i, p_i). To je polje sortirano pa koristimo binarno pretraživanje kako bi dohvat pozicija za neki zadani hash bio efikasan. Dodatno, one h_i-ove koji se pojavljuju jako velik broj puta njihove parove mičemo iz niza jer takvi ne donose korisnu informaciju, a samo povećavaju vremensku i memorijsku složenost.

[1] engl. Longest Increasing Subsequence; odatle dolazi inspiracija za naziv LISA - LIS Aligner

[2] Trenutna pretpostavka koju radimo je da dugačka očitanja imaju svoje jedinstveno mjesto u genomu. Neki drugi mapperi ne rade tu pretpostavku pa je u daljnjim planovima razvoja LISA-e dopustiti mogućnost poravnanja na više različitih mjesta unutar genoma.

A	C	G	T	A	G	A	C	G	T	A	G	A	T	A	G	A	A
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

(a) DNA nad kojom gradimo indeks, uz $k = 5$

ACGTA	1,7
AGACG	5
AGATA	11
ATAGA	13
CGTAG	2,8
GACGT	6
GATAG	12
GTAGA	3,9
TAGAA	14
TAGAC	4
TAGAT	10

(b) Indeks

Tablica 2.1: Konceptualni prikaz indeksa

2.2.2. LISA algoritam poravnanja

Algoritam poravnanja sastoji se od dva koraka:

1. Grubo određivanje pozicije očitanja unutar genoma
2. Fino određivanje pozicije očitanja unutar genoma

2.2.3. Grubo određivanje pozicije očitanja

U ovoj osnovnoj fazi algoritma poravnanja određujemo poziciju unutar genoma oko koje je očitanje koncentrirano. Opća ideja iza algoritma je konstrukcija niza brojeva (kojeg ćemo uskoro opisati) nad kojim pokrećemo algoritam nalaženja najduljeg rastućeg podniza[3]. Nađeni podniz ugrubo opisuje poziciju poravnanja i koristi se u sljedećoj fazi za traženje točne pozicije poravnanja.

Potreban niz P konstruiramo tako da promatramo sve nizove uzastopnih znakova i duljine k unutar očitanja s lijeva na desno. Za svaki od njih vršimo upit za pozicije unutar kojih se nalazi u indeksu. Na kraj niza P za svaku od dobivenih pozicija x

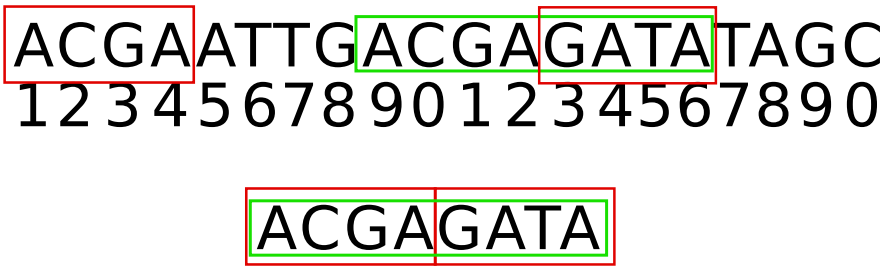
[3] algoritam je opisan u dodatku

dodajemo par (x, i). Kada smo to učinili za sve nizove i, sortiramo P po prvom broju u paru i pokrećemo algoritam pronalaska najduljeg zajedničkog podniza L nad nizom drugih brojeva parova u P. L koristimo u sljedećoj fazi. Pseudokod algoritma dan je u 1.

Algoritam 1 $GruboPoravnanje(O)$

$P \leftarrow \{\}$
$L \leftarrow \{\}$
for $i = 1...duljina(O) - k + 1$ **do**
  for all $p \in dohvatiPozicijeIzIndeksa(O[i...i + k - 1])$ **do**
    $P \leftarrow P \cup (p, i)$
  end for
  $P \leftarrow sortirajPoPrvomBroju(P)$
  $L \leftarrow najduljiRastuciPodnizPoDrugomBroju(P)$
  return L
end for

2.2.4. Fino određivanje pozicije očitanja



Slika 2.2: Višestruka poravnanja

Razlog zašto je ova faza potrebna je vidljiv na 2.2. Očito je da je pravo poravnanje smješteno oko pozicije 9, dok bismo pogrešnom naivnom procjenom možda rekli da je ono ipak na poziciji 1. Pseudokod 2 opisuje rad ove faze.

Algoritam 2 $FinoPoravnanje(L)$

$brojac(-\infty...\infty) \leftarrow 0$

for all $l \in L$ **do**

$++brojac(prvi(l) - drugi(l))$

end for

$finaProcjena \leftarrow \arg\max_i brojac(i)$

return $finaProcjena$

Mjera pouzdanosti

Mjerenje pouzdanosti je još uvijek otvoren problem za LISA-u. Trenutno je u uporabi omjer duljine pronađenog rastućeg niza i duljine očitanja.

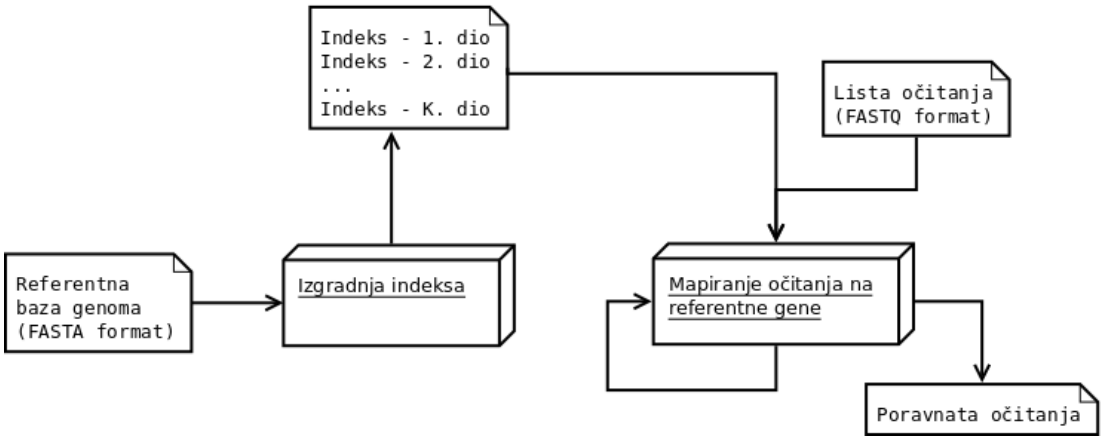
3. Implementacija

Sustav je implementiran u programskom jeziku C++ (zbog efikasnosti i ogromnog volumena podataka) koristeći nedavni standard C++0x/C++11 pomoću kompatibilnog GNU g++ prevoditelja.

Implementacija sustava može biti vrlo prirodno razložena na dva osnovna dijela:

1. izgradnja indeksa nad referentnom bazom gena (potrebno je pripremiti bazu genoma)
2. mapiranje očitanja na referentnu bazu (potrebno je pripremiti datoteku s očitanjima)

Prije bilo kakvog mapiranja potrebno je jednokratno izgraditi indeks, ali je nakon toga moguće neograničeno puta koristiti izgrađeni indeks pri mapiranju podataka, kao što je prikazano na dijagramu:



Slika 3.1: Dijagram komponenata s njihovih ulazima/izlazima

Ostatak poglavlja će detaljnije obraditi pojedine komponente opisanog algoritma te njihove ulazne/izlazne podatke.

3.1. Priprema podataka

3.2. Izgradnja indeksa

Zbog iznimnog volumena referentne baze od 47 GB i procijenjene veličine izlaznog indeksa od 340 GB memorije nije pogodno graditi indeks od cijele baze odjednom. Odabrali smo pristup gdje bazu dijelimo na više jednakih dijelova (čija se veličina može specificirati preko komandne linije) te gradimo zasebni indeks na svakom od tih dijelova.

Implementacijski gledano, indeks je binarna datoteka koja sadrži sortiranu hash tablicu svih genskih podsekvenci duljine N nukleotidnih baza[1]. Specifično, svaku podsekvencu duljine N možemo interpretirati kao broj u bazi 4 te tako dobiveni broj uzimamo kao ključ dotične pozicije koji se sortira i sprema u indeks. Ovakav odabir omogućuje brzo pretraživanje po ključu pomoću logaritamskog binarnog pretraživanja. Dodatno, moguće je i pretraživati podsekvence po ključu koji je manji od N iz razloga što za $N < 32$ i 64-bitni tip podataka nema preljeva.

Sama struktura binarne indeks datoteke je najbolje opisana opisnikom pretinaca hash tablice, prikazanim u sljedećem odsječku:

```
struct Entry {
  size_t position;
  hash_t hash;

  friend bool operator < (const Entry& a, const Entry& b) {
    return a.hash < b.hash;
  }
};
```

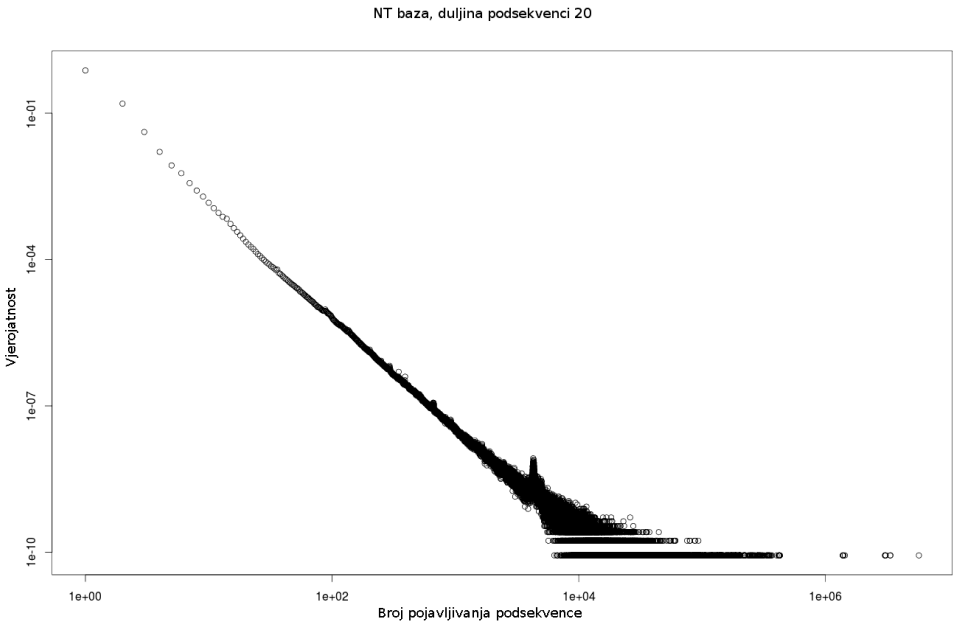
Bitno je napomenuti kako koristimo razne implementacijske i algoritamske optimizacije radi efikasnijeg baratanja podataka ogromnog volumena s kakvima radimo u ovom projektu. Ovdje ćemo ukratko objasniti neke od njih:

Paralelno sortiranje pretinaca se pokazalo kao prirodna optimizacija nakon profiliranja programskog koda. Ispostavilo se da je upravo ovaj korak najsporiji dio izgradnje indeksa. Za ubrzanje koristimo paralelni stabilni sort implementiran u OpenMP biblioteci (verzije 1.3).

Micanje podsekvenci s visokom frekvencijom pojavljivanja . Intuitivno je jasno kako podsekvence s vrlo visokom frekvencijom pojavljivanja (preko 6 redova

[1] tipično, uzimamo $N = 16$ ili $N = 20$

veličine više od prosjeka) kodiraju repetitive regije ili regije niske složenosti i pritome nimalo ne pridonose kvaliteti poravnanja. Kako bismo što smanjili obujam podataka i ubrzali fazu poravnavanja, sustav izbacuje one podsekvence čija frekvencija pojavljivanja je 5 puta veća od prosjeka [2]. Kako bismo opravdali takvu redukciju podataka, analizirali smo frekvenciju pojedinih podsekvenci i potvrdili kako se pojedine jedinke pojavljuju više redova veličina od prosjeka, kao što je prikazano na sljedećem grafu:



Slika 3.2: Vrijednost y-ona predstavlja vjerojatnost da se slučajno odabrana podsekvenca u bazi javlja točno x puta.

Cjelokupna izgradnja indeksa nakon optimizacija traje otprilike 4 sata računalu performansa 192 GB RAM-a i Intel(R) Xeon(R) procesorom s 12 jezgara koje rade na 2.40 GHz.

3.3. Mapiranje očitanja

Mapiranje očitanja na referentnu bazu zahtjeva izgrađeni dio indeksa koji se u potpunosti učitava u radnu memoriju. Iz ovog razloga je potrebno unaprijed odrediti prihvatljivu veličinu indeksa koju je moguće u potpunosti učitati.

[2] postiže se smanjenje količine podataka od oko 15%

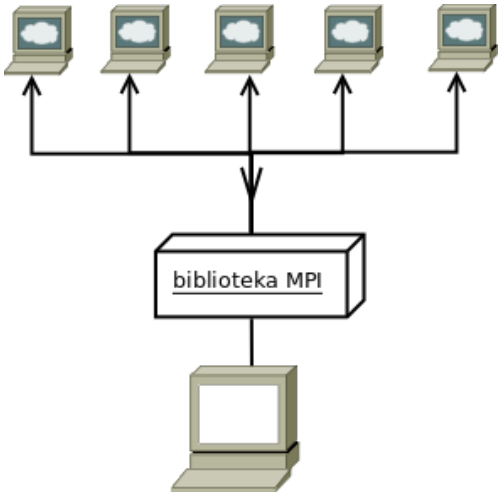
Algoritam mapira pojedino očitanje tako da pokušava pronaći najbolje poravnanje s referentnom bazom. Za svaku podsekvencu pronalaze se sve pozicije u genima gdje se navedena podsekvenca pojavljuje. Pretraga se, zbog sortiranosti podataka vrši binarnim pretraživanjem, a može se zbog velike količine podataka i uniformne hash funkcije ubrzati interpoliranim binarnim pretraživanjem. [12]

Za svaki gen u kojem je pronađena barem jedna odgovarajuća podsekvenca se poravnava s fragmentom pomoću opisanog algoritma najdužeg rastućeg podniza. Implementacija algoritma je ostvarena u složenosti $O(n \log n)$ pomoću vrlo efikasnog Fenwickovog stabla koje podržava sve potrebne operacije kao i klasično balansirano binarno stablo, ali uz puno bolju vremensku i memorijsku konstantu. [3]

Kako bi se ubrzao proces mapiranja, očitanja su ravnomjerno podijeljena među jezgrama računala (sustav automatski detektira broj jezgara i ravnomjerno dijeli posao). Osiguranje ravnomjerne zaposlenosti je ostvareno korištenjem reda dretvenih zadataka (engl. *ThreadPool*) uz rotirajući binarni semafor (engl. *Spinlock*) kako bi se fragmenti fino granulirali i radi osiguranja što efikasnije sinkronizacije među dretvama. Koristimo biblioteku *pthreads* kako bi omogućili višedretveno mapiranje.

3.4. Distribucija mapiranja

Veliki volumen podataka prisutan pri rješavanju ovog problema nas je potakao da distribuiramo algoritam na više računala. Specifično, korisnik pokreće sustav za mapiranje na jednom računalu koje potom dijeli ulaznu datoteku očitanih fragmenata na jednake dijelove te ih šalje na farmu servera koji potpuno paralelno obrađuju dane podatke i pri završetku šalju rezultate natrag na originalno računalo. Distribucija se vrši preko biblioteke MPI[5], kao što je prikazano na slici:



Slika 3.3: Distribucija mapiranja na farmu servera

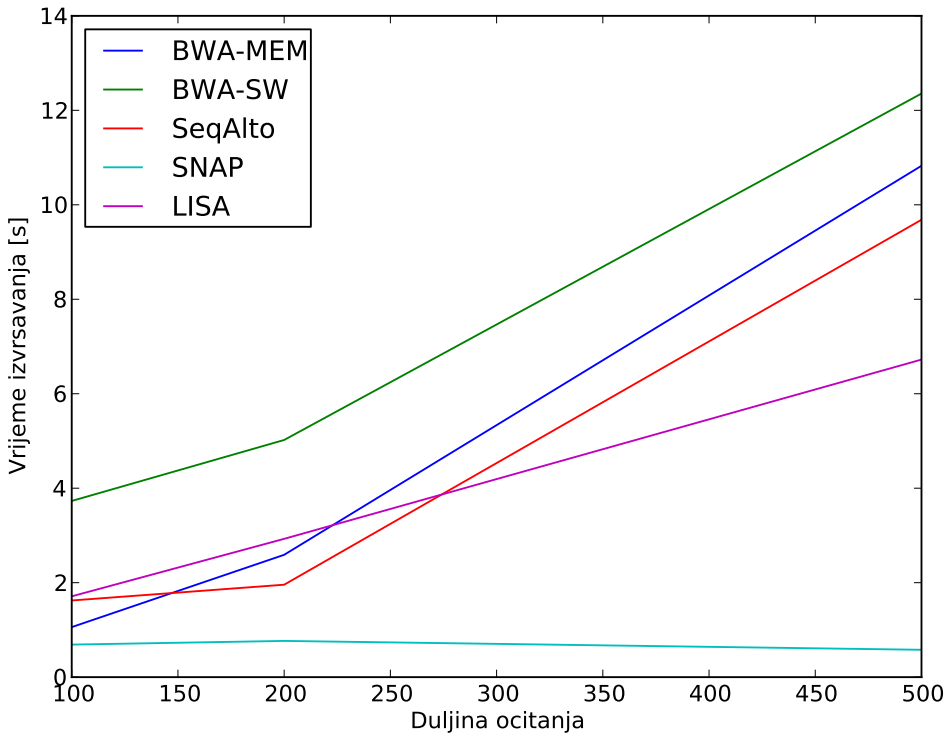
Isplativost distribucije mapiranja proizlazi iz fine granularnosti originalnog problema - višestruko se riješava identična zadaća mapiranja fragmenta na referentni genom, što svakoj paralelizaciji na broj računala znatno manji od broja fragmenata osigurava linearno ubrzanje vremena izvršavanja.

4. Rezultati

4.1. Pregled

U ovoj sekciji analiziramo performanse LISA-e. Analiza se provodi poravnanjem nad poznatim vrstama bakterija - kuga, salmonela, streptokok i E. Coli. Motivacija iza preciznog poravnanja očitanja u genome bakterije je jasna - efikasno i točno određivanje primjerice mutacija omogućuje brži dizajn lijekova za bakterije koje su stekle otpornost na postojeće lijekove.
Dajemo usporedbu LISA-e sa nekoliko modernih mappera. Uspoređujemo točnost i brzinu sa BWA-SW, BWA-MEM[7], SNAP i SeqAlto mapperom. Testna očitanja su generirana alatom za simuliranje wgsim koji je široko prihvaćen u bioinformatičkim krugovima i korišten u većini usporedbi. wgsim je korišten sa predodređenim parametrima za generiranje greške (greška se unosi u 2% nukleotida). Svaki test je napravljen na uzorku od 100000 slučajno generiranih očitanja. Svi testovi su provedni na računalu koje je opremljeno sa 192Gb RAM-a i Intel(R) Xeon(R) procesorom s 12 jezgre koje rade na 2.40 GHz.

4.2. E. Coli



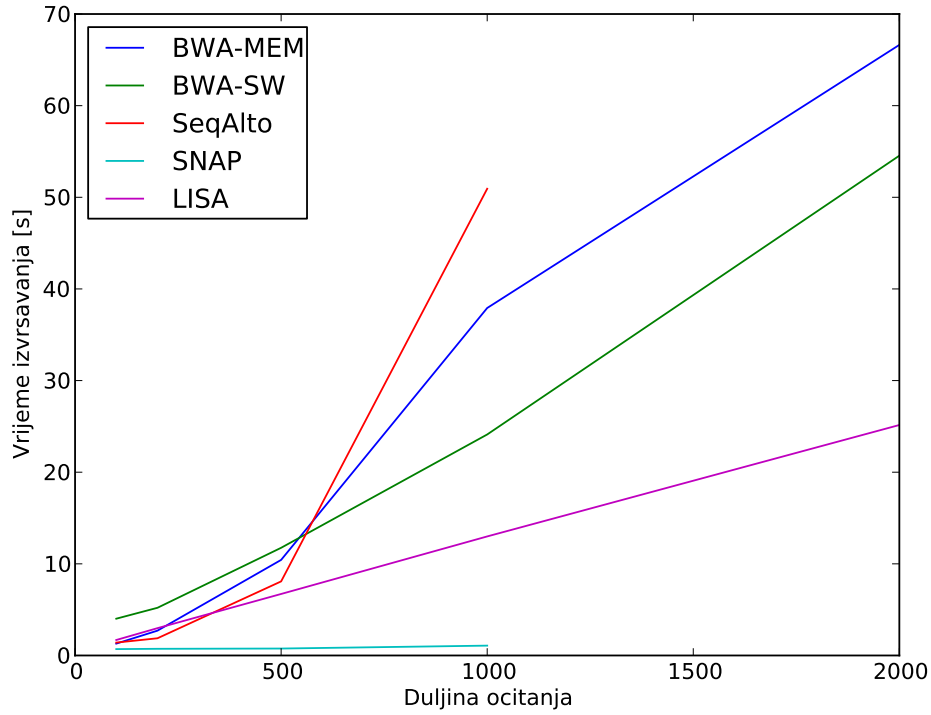
Slika 4.1: Usporedba performansi mappera

Genom bakterije *E. Coli* sadrži otprilike 5M parova nukleotida. Proveli smo testiranja poravnanja očitanja duljine 100, 200 i 500. Natjecanje vremenskih performansi je jasno pobijedio SNAP (4.1). To ne iznenađuje jer je on posebno dizajniran za kraće duljine očitanja. Kasnije ćemo vidjeti da prestaje biti praktičan kada duljine očitanja rastu. LISA performansama nadmašuje preostale mappere. U tablici 4.1 je vidljivo da su na testiranju svi mapperi bili iznimno precizni.

Duljine	100	200	500
BWA-MEM	99.60	99.64	99.65
BWA-SW	99.55	99.64	99.65
SeqAlto	99.10	99.63	99.64
SNAP	99.61	97.37	99.56
LISA	99.54	99.63	99.62

Tablica 4.1: Postotak točno poravnatih očitanja

4.3. Salmonella Enterica



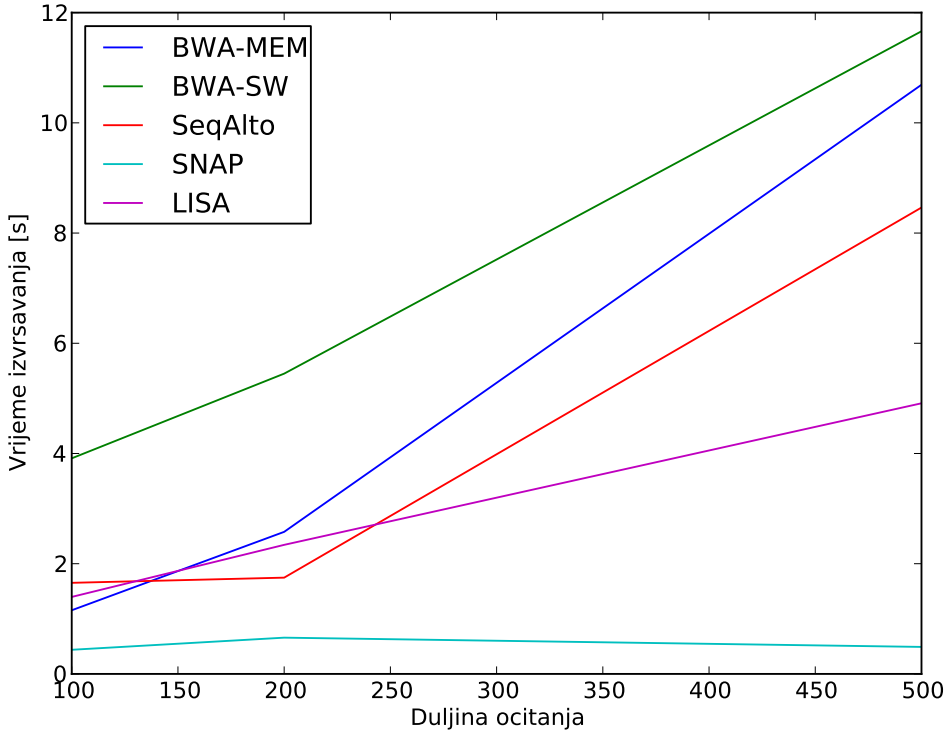
Slika 4.2: Usporedba performansi mappera

Genom *S. Enterica*-e sastoji se od otprilike 5M parova nukoleotida. Proveli smo testiranja poravnanja očitanja duljina 100, 200, 500, 1000 i 2000. Na slici 4.2 je vidljivo da SNAP ima najbolje performanse za očitanja do duljine 1000. Za duljinu 2000 je imao poteškoća sa izvođenje pa nemamo rezultat. Zbog preglednosti grafa izostavili smo rezultat SeqAlto-a na očitanjima duljine 2000 (eksperiment je trajao 6m12s). LISA nadmašuje sve mappere osim SNAP-a na malim duljinama. Tablica 4.2 ističe iznimnu točnost svih mappera.

Duljine	100	200	500	1000	2000
BWA-MEM	99.57	99.70	99.83	99.97	99.99
BWA-SW	99.53	99.69	99.82	99.97	99.99
SeqAlto	99.06	99.69	99.83	99.97	99.99
SNAP	99.13	97.02	99.47	97.95	-
LISA	99.54	99.67	99.81	99.95	99.91

Tablica 4.2: Postotak točno poravnatih očitanja

4.4. Streptococcus Pneumoniae



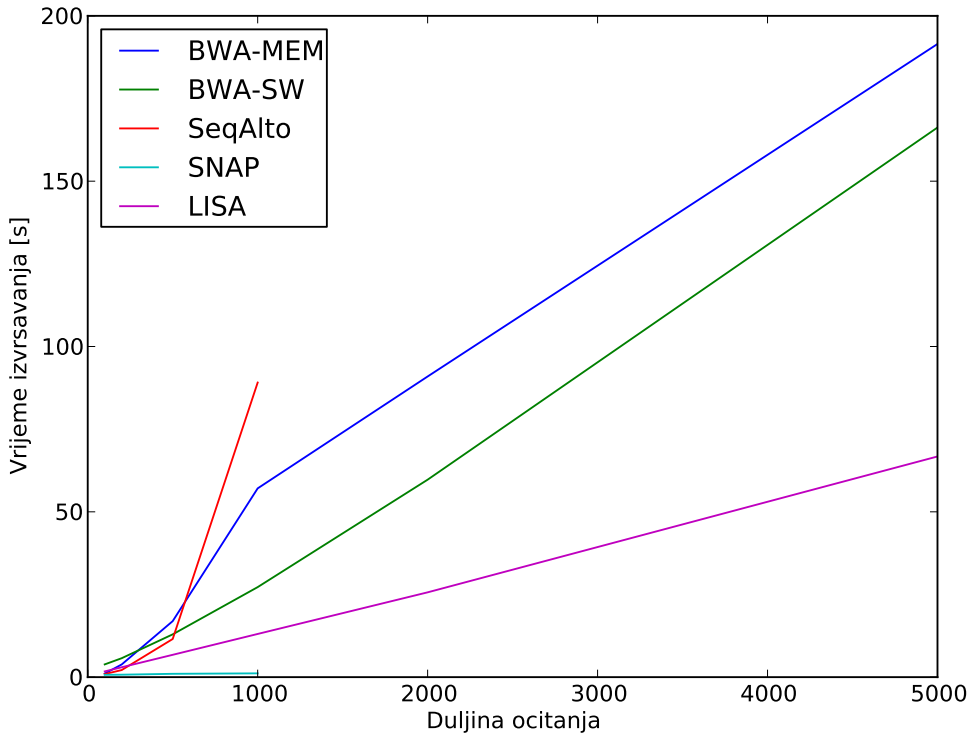
Slika 4.3: Usporedba performansi mappera

Genom *S. Pneumoniae* sadrži otprilike 3M parova nukleotida. Proveli smo testiranja poravnanja očitanja duljina 100, 200 i 500. Na slici 4.3 vidljivo je kako na ovim kraćim duljinama SNAP ima najbolje performanse. To je očekivan rezultat budući da je on pažljivo dizajniran za kratke duljine očitanja. LISA je bolja od preostalih mappera. Tablica 4.3 pokazuje vrhunsku preciznost svih mappera. Valja istaknuti kako SNAP ima nešto manju preciznost u usporedbi s ostalim mapperima.

Duljine	100	200	500
BWA-MEM	99.10	99.38	99.55
BWA-SW	99.12	99.37	99.54
SeqAlto	98.57	99.31	99.52
SNAP	97.74	96.26	98.82
LISA	98.98	99.36	99.53

Tablica 4.3: Postotak točno poravnatih očitanja

4.5. Yersinia Pestis



Slika 4.4: Usporedba performansi mappera

Genom *Y. Pestis*, šire poznate kao kuga, sadrži otprilike 5M parova nukleotida. Proveli smo testiranja poravnanja očitanja duljina 100, 200, 500, 1000, 2000 i 5000. Slika 4.4 demonstrira efikasnost SNAP-a na manjim duljinama. SNAP je imao poteškoća s obrađivanjem očitanja duljina većih od 1000, iz tog razloga izostaju ti rezultati. Zbog preglednosti grafa na njemu nisu vidljiva vremena izvođenja SeqAlto-a za duljine 2000 (7m9s) i 5000 (19m27s). Za dugačka očitanja LISA se ponaša uvjerljivo najbolje, postižući ubrzanja 3-20x u usporedbi s ostalim mapperima. Točnosti svih mappera su vidljive u tablici 4.4. Svi mapperi su vrlo velike i točnosti zadovoljavajuće za praktičnu uporabu.

Duljine	100	200	500	1000	2000	5000
BWA-MEM	95.63	96.31	97.34	98.66	99.90	100.00
BWA-SW	95.60	96.31	97.33	98.58	99.79	99.98
SeqAlto	95.12	96.29	97.37	98.63	99.88	63.85
SNAP	96.64	93.34	96.57	96.22	-	-
LISA	95.53	96.21	97.21	98.54	99.72	99.81

Tablica 4.4: Postotak točno poravnatih očitanja

4.6. Dodatne informacije

Ovdje navodimo naredbe kojima smo pokretali svaki od mappera.

BWA-MEM	bwa mem -t 24 index ocitanja.fq > rezultat.sam
BWA-SW	bwa bwasw -t 24 index ocitanja.fq > rezultat.sam
SeqAlto	seqalto_basic align index -n 1 -f -p 24 ocitanja.fq >rezultat.sam
SNAP	Za duljine < 500: snap single index ocitanja.fq -t 24 Za duljinu 500: snap single index ocitanja.fq -t 24 -d 20 Za duljinu 1000: snap single index ocitanja.fq -t 24 -d 40 -h 100 -n 2000
LISA	client solve genom.fasta index ocitanja.fq /dev/null

5. Zaključak i daljnji rad

5.1. Osvrt

U ovom radu prezentirali smo LISA-u, novi alat za poravnanje očitanja na referentni genom. Proveli smo usporedbu s trenutno najpopularnijim alatima i pokazali da LISA već u svojoj ranoj fazi razvoja pokazuje jako dobre performanse, pogotovo za očitanja velike duljine. Duljine očitanja su u trendu porasta tako da predviđamo da će tehnike korištene biti od značaja u bližoj i daljoj budućnosti.

5.2. Daljnji rad

Najveći istraživački prioritet kojem su autori trenutno posvećeni je povećanje točnosti mapiranja očitanja malih duljina kao i onih s velikom greškom. Slutnja koju još valja potvrditi je da zadovoljavajuće točnosti možemo dobiti aproksimiranjem metrika kvalitete koje koriste SNAP ili SeqAlto. Te se metrike mogu aproksimirati s manjom računskom složenosti od izračuna pravih vrijednosti pa vjerujemo da neće biti znatnog gubitka vremenskih performansi.
Implementacijski prioritet je ispisivanje rezultata u standardiziranom SAM formatu jer se tada otvara mogućnost korištenja mnogih već razvijenih alata za analizu efikasnosti i rezultata.

6. Dodatak

6.1. Najdulji rastući podniz[4]

Definicija: Za zadani niz od n brojeva a_i potrebno je naći najveći podskup indeksa tako da u odabranom podskupu za svako $i < j$ vrijedi $a_i < a_j$.
U ovom poglavlju dani su isječci koda koji jasno i jednostavno implementiraju algoritam koji rješava problem najduljeg rastućeg podniza. U svrhu sažetosti izostavljena je implementacija Fenwickovog stabla. To je stablo klasična struktura i njenu je implementaciju vrlo lako dobaviti. Samo napominjemo da operacije $Fenwick.get(x)$ dobavlja najveći element u intervalu $[1, x]$, dok $Fenwick.update(x, v)$ postavlja vrijednost na indeksu x na vrijednost v.

```
void reconstructLIS(vector<int>* result,
                    int last,
                    const vector<int>& reconstructionTable) {
  result->clear();
  result->reserve(reconstructionTable.size());
  for ( ; last != -1; last = reconstructionTable[last]) {
    result->push_back(last);
  }
  reverse(result->begin(), result->end());
}
```

```
void calcLongestIncreasingSubsequence(
    vector<int>* result,
    const vector<pair<int, int> >& elements) {
  int n = elements.size();

  vector<int> dpTable(n, 0);
  vector<int> reconstructionTable(n, -1);

  int maxSecond = -1;
  for (int i = 0; i < elements.size(); ++i) {
    maxSecond = max(maxSecond, elements[i].second);
  }

  FenwickMax fm(maxSecond);

  for (size_t i = 0; i < n; ++i) {
    std::pair<int, int> best = fm.get(elements[i].second-1);
    if (best.first) {
      dpTable[i] = best.first+1;
      reconstructionTable[i] = best.second;
    } else {
      dpTable[i] = 1;
    }
    fm.update(elements[i].second, make_pair(dpTable[i], i));
  }

  int last =
      max_element(dpTable.begin(), dpTable.end())-dpTable.begin();
  reconstructLIS(result, last, reconstructionTable);
}
```

LITERATURA

[1]

[2] The human genome project completion: Frequently asked questions. `http://www.genome.gov/11006943`, 2010. [dohvaćeno 22. travnja 2013.].

[3] Peter M Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and Experience*, 24(3):327–336, 1994.

[4] Michael L. Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29 – 35, 1975.

[5] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[6] Bruce Johnson. A bioinformatics-inspired adaptation to ukkonen's edit distance calculating algorithm. In *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ACM-SE 46, pages 46–50, New York, NY, USA, 2008. ACM.

[7] Heng Li and Richard Durbin. Fast and accurate long-read alignment with burrows–wheeler transform. *Bioinformatics*, 26(5):589–595, March 2010.

[8] Elaine R Mardis. Next-generation dna sequencing methods. *Annu. Rev. Genomics Hum. Genet.*, 9:387–402, 2008.

[9] John C. Mu, Hui Jiang, Amirhossein Kiani, Marghoob Mohiyuddin, Narges Bani Asadi, and Wing H. Wong. Fast and accurate read alignment for resequencing. *Bioinformatics*, 28(18):2366–2373, September 2012.

[10] Stephan Pabinger, Andreas Dander, Maria Fischer, Rene Snajder, Michael Sperk, Mirjana Efremova, Birgit Krabichler, Michael R. Speicher, Johannes Zschocke, and Zlatko Trajanoski. A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in Bioinformatics*, 2013.

[11] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[12] Mark Allen Weiss and Susan Hartman. *Data structures and problem solving using Java*, volume 204. Addison-Wesley Reading, 1998.

[13] Matei Zaharia, William J. Bolosky, Kristal Curtis, Armando Fox, David A. Patterson, Scott Shenker, Ion Stoica, Richard M. Karp, and Taylor Sittler. Faster and more accurate sequence alignment with snap. *CoRR*, abs/1111.5572, 2011.

LISA - Alat za poravnanje DNA očitanja

Sažetak

DNA je struktura koja kodira živi svijet. Bolje razumijevanje funkcija pojedinih dijelova može dovesti do pravovremene detekcije i liječenja mnogih bolesti. Strojevi za očitanje DNA iz dana u dan napreduju i proizvode velike količine sve duljih očitanja. Lociranje očitanja unutar referentnog genoma jedan je od temeljnih otvorenih problema u bioinformatici. U ovom radu predlažemo novi algoritam poravnanja DNA temeljen na brzom traženju najduljeg rastućeg podniza, posebno pogodnim za dulja očitanja. Napravljena je usporedba točnosti i brzine s trenutno najefikasnijim i najpopularnijim alatima.

Ključne riječi: poravnavanje DNA, bioinformatika, algoritmi, hash, LISA, najdulji rastući podniz

LISA - DNA reads alignment tool

Abstract

DNA is a structure which encodes all of the living world. Better understanding of it's particular section could lead to detection and curing of many diseases. DNA sequencing machines are getting better every day and producing big amounts of ever longer reads. Locating these reads inside a reference genome is a fundemental open problem in bioinformatics. This paper presents a novel DNA alignment algorithm, based on a fast longest increasing subsequence search. It is very feasible for long reads. Detailed analysis and comparison with state-of-the-art tools is given.

Keywords: DNA alignment, bioinformatics, algorithms, hash, LISA, longest increasing subsequence