Paralelno aproksimativno pretraživanje uzoraka u tekstu

Paralelizam i konkurentnost - grupa 15

Eno Peršić
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu
Zagreb, Hrvatska
eno.persic@fer.hr

Karla Pišonić
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu
Zagreb, Hrvatska
karla.pisonic@fer.hr

Antonio Sabljić
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu
Zagreb, Hrvatska
antonio.sabljic@fer.hr

Domagoj Sviličić
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu
Zagreb, Hrvatska
domagoj.svilicic@fer.hr

Hrvoje Tkalčević
Fakultet elektrotehnike i računarstva
Sveučilište u Zagrebu
Zagreb, Hrvatska
hrvoje.tkalcevic@fer.hr

***Sažetak*—Ovaj dokument predstavlja dio projektne dokumentacije u sklopu projektnog zadatka u okviru kolegija Paralelizam i konkurentnost na Fakultetu elektrotehnike i računarstva u Zagrebu.**

IME PROJEKTA: Usporedba vremena izvođenja algoritama za aproksimativno pretraživanje uzoraka u tekstu s obzirom parametar tolerancije

***Index Terms*—paralelizam, bioinformatika, sekvenciranje genoma**

I. Uvod

Aproksimativno pretraživanje uzoraka u tekstu odnosi se na proces pronalaženja uzorka ili podniza unutar teksta uz određenu dopuštenu razinu pogrešaka ili razlika. Taj postupak koristi se kada je potrebno pronaći uzorak ili slične uzorke unutar teksta, čak i kada nema potpune podudarnosti s uzorkom. Primjena aproksimativnog pretraživanja uzoraka je ključna u mnogim područjima, uključujući bioinformatiku (sekvenciranje genoma, pretraživanje sličnih sekvenci), pretraživanje teksta (u pretraživačima, obradi prirodnog jezika, analizi podataka), kompresiji podataka (prepoznavanje i eliminacija ponavljajućih uzoraka), obradi slika i mnogim drugim područjima. Ovaj projekt obrađuje problematiku traženja genske sekvence unutar većih referentih nizova koji predstavljaju očitanja uređaja za sekvenciranje. Referentni niz i tražena sekvenca koju je potrebno pronaći kao podniz unutar spomenutog referentnog niza predstavljeni su kao niz nukleotidnih baza A, T, C i G. Nekoliko je razloga zbog čega je primjerenije koristiti aproksimativno prepoznavanje uzoraka u nizovima nukleotida umjesto određivanja potpuno točnih sekvenci:

- **Pogreške u sekvenciranju**: procesi sekvenciranja genoma nisu savršeni i mogu dovesti do pogrešaka. To može uključivati zamjene, umetanja ili brisanja nukleotida. Aproksimativno prepoznavanje uzoraka može pomoći u identifikaciji sekvenci koje su blizu traženog uzorka, unatoč tim pogreškama.
- **Genetske varijacije**: genetski materijal može varirati između različitih jedinki iste vrste. Na primjer, ljudski genomi se razlikuju jedan od drugog za oko 0,1%. Aproksimativno prepoznavanje uzoraka može pomoći u identifikaciji sekvenci koje su slične traženom uzorku, ali imaju male varijacije.
- **Evolucija i mutacije**: tijekom vremena, genetski materijal može evoluirati i mutirati što može dovesti do promjena u sekvencama nukleotida. To je osobito izraženo kod genoma virusa i bakterija koji se uostalom i najčešće proučavaju unutar područja bioinformatike. Aproksimativno prepoznavanje uzoraka može pomoći u identifikaciji sekvenci koje su evoluirale ili mutirale od traženog uzorka.
- **Funkcionalne varijacije**: u nekim slučajevima, različite sekvence nukleotida mogu imati slične funkcije. Na primjer, različite sekvence mogu kodirati isti protein. Aproksimativno prepoznavanje uzoraka može pomoći u identifikaciji tih funkcionalno sličnih sekvenci.

Jedan od središnjih pojmova u kontekstu aproksimativnog pretraživanja uzorka u tekstu predstavlja parametar tolerancije koji se definira kao dopušteni broj grešaka ili razlika koji se uzima u obzir prilikom usporedbe traženog uzorka s podacima u tekstu. Ova tolerancija omogućuje algoritmima koji rade s podacima sklonim pogreškama da pronađu uzorak ili podniz unutar teksta, čak i ako nije potpuno identičan traženom uzorku. Ovaj projekt bazira se na proučavanju ovisnosti vremena izvođenja različitih algoritama za aproksimativno

pretraživanje uzoraka u tekstu o iznosu parametra tolerancije. U osnovi, veći broj dopuštenih pogrešaka očito će povećati vrijeme potrebno za pretragu jer algoritam mora provjeriti više mogućih podudaranja. Međutim, točan utjecaj može se razlikovati ovisno o specifičnom algoritmu i načinu na koji se pogreške obrađuju. Korišteni algoritmi su Lavenshteinov algoritam, Rabin-Karp, naivni algoritam i Bitap algoritam. Međusobno se razlikuju po pomoćnim strukturama koje koriste te načinima kojima sužavaju prostor rješenja. Svi ovi algoritmi s većim ili manjim modifikacijama nalaze primjenu kako unutar područja bioinformatike tako i izvan njega. U ovome projektu osnovna im je namjena odabir podnizova iz referentnog niza koji se od traženog uzorka razlikuju za iznos definiran parametrom tolerancije. Takvi kandidati traženog uzorka potom se najčešće koriste kao grupa nad kojom se mogu učinkovitije vršiti bilo kakve daljnje analize. Cilj projekta je da prosljeđivanjem konstatnog traženog uzorka nad konstantnim skupom referentnih nizova i variranjem parametra tolerancije otkrijemo koji od korištenih algoritama nudi najveću robusnost, tj. najkonstantnije vrijeme izvođenja. Obzirom da parametar tolerancije nije broj koji je moguće jednostavno izračunati, nego se najčešće određuje empirijski ovisno o konkretnom problemu, ovaj projekt mogao bi ponuditi odgovor na pitanje koji od algoritama bi predstavljao najbolji izbor za korištenje prilikom postupka traženja tog parametra.

A. Ulazni podaci

Polazišna točka projekta je tekstualna datoteka u FASTA formatu koja predstavlja realna očitanja uređaja za sekvencioniranje genoma provedena nad uzorkom bakterije (Escherichia coli K-12 substr. MG1655 genome). U bioinformatici i biokemiji, FASTA format je tekstualni format za prikazivanje nukleotidnih sekvenci ili sekvenci aminokiselina, u kojima se nukleotidi ili aminokiseline prikazuju pomoću jednoslovnih kodova. Format omogućuje da imena sekvenci i komentari prethode sekvencama. Potekao je iz softverskog paketa FASTA i od tada je postao gotovo univerzalni standard u bioinformatici.

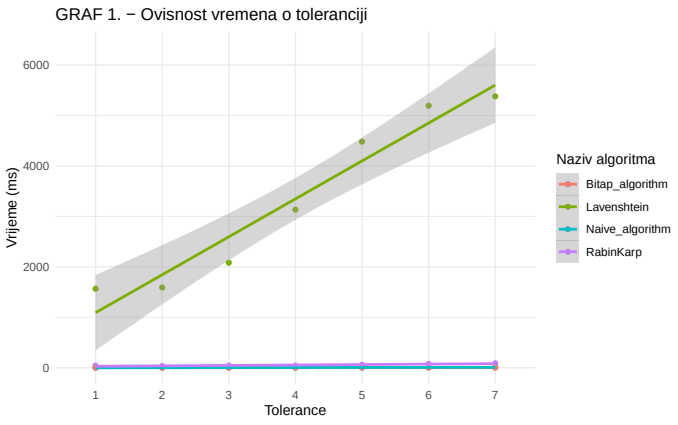
Jednostavnost FASTA formata olakšava manipulaciju i parsiranje sekvenci pomoću alata za obradu teksta i skriptnih jezika. Sekvencija započinje znakom veće od (“>”) praćenim opisom sekvence (sve u jednom retku). Linije koje odmah slijede liniju opisa su prikaz sekvence, s jednim slovom po aminokiselini ili nukleinskoj kiselini. Dotična datoteka dostupna je na sljedećem linku https://nanopore.s3.climb.ac.uk/MAP006-1_2D_pass.fasta i preuzeta je od istraživačke skupine Loman Labs sa Sveučilišta u Birminghamu (detaljnije na http://lab.loman.net/2015/09/24/first-sqk-map-006-experiment/). Svaka sekvenca očitanja potpuno je nezavisna od drugima u smislu da se ne mora nadovezivati na prethodnu i sljedeću unutar datoteke te da njen položaj u datoteci ne preslikava njen položaj u stvarnom genomu. To je posljedica činjenice da uređaji za sekvencioniranje nisu sposobni očitati cijeli genom nekog uzorka već samo sekvence ili dijelove tog genoma. Na temelju tih očitanja i njihovih međusobnih preklapanja (ako ih je provedno dovoljno veliki broj) moguće je određenim tehnikama rekonstruirati cijeli genom mikroorganizma. Cjelokupni genom bakterije Echerichije coli dostupan je na stranicama Oxford Nanopor Technologies (sa linka: https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/005/845/GCF_000005845.2_ASM584v2/GCF_000005845.2_ASM584v2_genomic.fna.gz).

B. Metodologija

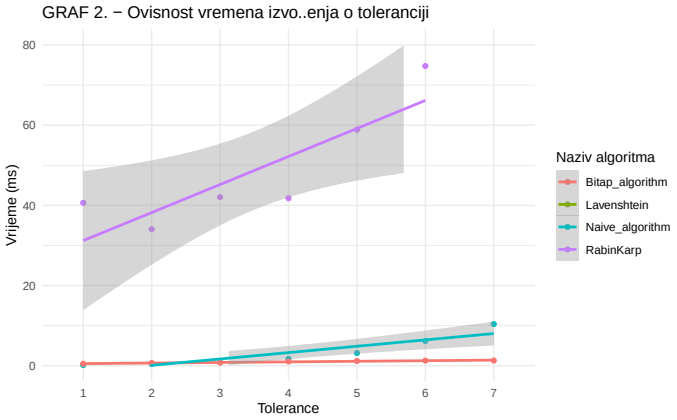
Tu napišite proces kreiranja datoteke, load balancinga i algoritme koje smo koristili. Koristili smo paralelizaciju s 8 dretvi jer smo se vodili činjenicom da testno računalo ima 8 logičnih jezgri pa se na svakoj jezgri može nesmetano izvoditi jedna dretva i to bi trebalo pružiti najbolje performanse izvođenja.

C. Prikaz rezultata

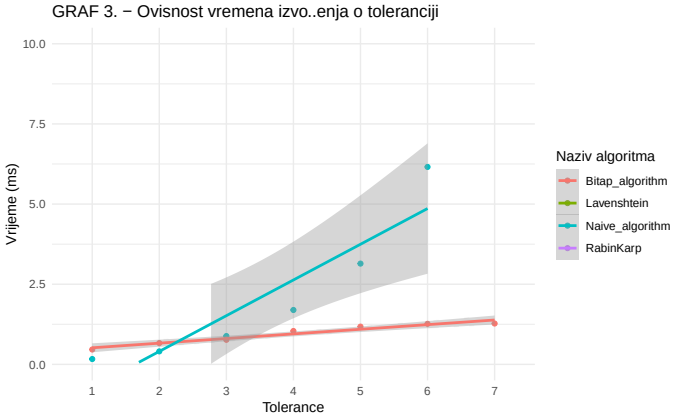
U nastavku su prikazani grafovi ovisnosti vremena izvođenja svakog od korištenih algoritama u odnosu na parametar tolerancije. Obzirom na veliku disipaciju vremena izvođenja između različitih algoritama, grafički prikaz je razdvojen u 3 grafa koji imaju različito limitirane vrijednosti na y-osi kako bi dovoljno dobro mogli pokazati međuodnose između algoritama. Možemo reći kako je GRAF2 uvećana verzija GRAFA1 , a GRAF3 uvećana verzija GRAFA2. Grafovi su nacrtani pomoću paketa ggplot2 unutar programskog jezika R. Vremena izvođenja za svaki algoritam upisana su .csv datoteku koja je učitana u radnu okolinu i uz pomoć brojnih vizualizacijskih alata koje nam R pruža stvorene su vizualizacije koje bi trebale ponuditi nedvosmislen odgovor na pitanje koji od algoritama bi predstavljao najbolji izbor za korištenje prilikom postupka traženja parametra tolerancije. Najprije je ostvaren točkasti prikaz ovisnosti vremena ovisnosti svakog algoritma o prosljeđenom parametru tolerancije, a potom su kroz te točke provučeni aproksimativni pravci (svaki pravac predstavlja jedan algoritam),a osjenčani dijelovi oko tih pravaca predstavljaju intervale pozdanosti.



Slika 1. GRAF1 - Ovisnost vremena izvođenja o parametru tolerancije



Slika 2. GRAF2 - Ovisnost vremena izvođenja o parametru tolerancije



Slika 3. GRAF3 - Ovisnost vremena izvođenja o parametru tolerancije

Odgovor na pitanje: "Koji od korištenih algoritama nudi najveću robusnost, tj. najkonstantnije vrijeme izvođenja prilikom variranja parametra tolerancije" lako ćemo dobiti proučavajući nagibe pravaca. Očito će algoritam sa najpoloženijim pravcem biti odgovor na naše pitanje. Na GRAFU1 jasno je vidljivo kako se pravac Lavenshteinovog algoritma jako brzo uspinje u odnosu na druge pravce. Metodom eliminacije njega odmah možemo izbaciti iz daljnjeg proučavanja. Kako situacija nad preostalim pravcima nije očito vidljiva promatrajući isključivo GRAF1, stvorili smo grafički prikaz GRAF2 na kojem smo postavili manji limit na y-osi. Sada je opet trivijalno uočiti da graf algoritma Rabin-Karp najbrže raste u odnosu na druge. Odnos između preostala dva pravca je također već vidljiv, a u GRAFU3 i dodatno je naglašeno kako pravac algoritma Bitap ima najblaži rast te je on pobjednik našeg istraživačkog pitanja. Testni primjeri napravljeni su na računalu sa modelom procesora: "AMD Ryzen 3 5425U with Radeon Graphics" i operacijskim sustavom Ubuntu 22.04. Eventualne nadogradnje ovog projekta uključivale bi i testiranje na više različitih platformi kako se bolje usporedili dobiveni rezultati.

LITERATURA

[1] Miller,J.R. et al. (2010) Assembly algorithms for next-generation sequencing data. Genomics, 95, 315–327.
[2] Ohlebusch,E. & Gog,S. (2010) Efficient algorithms for the all-pairs suffix-prefix problem and the all-pairs substring-prefix problem. Inf. Process. Lett., 110, 123–128.
[3] Gusfield,D. (1997) Algorithms on strings, trees, and sequences: computer science and computational biology Cambridge University Press.
[4] Butler,J. et al. (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. Genome Res., 18, 810–20.
[5] Simpson,J.T. et al. (2009) ABySS: a parallel assembler for short read sequence data. Genome Res., 19, 1117–23.