

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

SCCG algoritam

Martin Bakač, Marta Bonacin

Voditeljica: *Mirjana Domazet-Lošo*

Zagreb, svibanj 2023.

SADRŽAJ

1. Uvod	1
2. SCCG algoritam	2
2.1. Opis algoritma	2
2.1.1. Faza lokalnog podudaranja	2
2.1.2. Faza globalnog podudaranja	4
2.2. Implementacija	4
2.2.1. Kompresija	4
2.2.2. Dekompresija	7
3. Rezultati	8
4. Zaključak	9
5. Literatura	10
6. Sažetak	11

1. Uvod

Genom predstavlja potpuni skup genetskih instrukcija u nekom organizmu [2]. Dakle genom obuhvaća sve gene nekog organizma. Gen je osnovna jedinica nasljeđivanja. Sastoji se od određenog broja nukleotidnih parova [1] pri čemu se svaki nukleotid sastoji od nukleinskih baza (adenin A, citozin C, gvanin G, timin T). Genomi su veliki skupovi podataka npr. ljudski genom se sastoji od 6.4 milijardi slova (baze A, C, G, T). Stoga ukoliko bi se radilo s takvim podacima u sklopu nekakvog istraživanja veliki problem bi bio kako pohraniti toliku količinu informacija. Podatke bi trebalo nekako sažeti te tako pohraniti, odnosno trebala bi se napraviti kompresija podataka. U ovome radu obradit će se jedan algoritam sažimanja sastavljen specifično za sažimanje genoma. Uz opis algoritma bit će prezentirana i implementacija u C++ programskom jeziku.

2. SCCG algoritam

SCCG je algoritam koji se koristi za kompresiju genoma [4]. Već postoje razni alati za kompresiju podataka, neki od njih su: 7-zip, WinZip, WinRAR, PeaZip [3], no oni ne iskorištavaju specifičnosti genoma kako bi postigli bolju kompresiju. Neke karakteristike genoma su mala veličina abecede i količina ponavljanja u sekvencama. Kada bi se iskoristile ove specifičnosti postigla bi se bolja kompresija podataka. SCCG je dakle algoritam koji obavlja korake pretprocesiranja kako bi napravio što bolju kompresiju. U nastavku objašnjavamo sam algoritam te njegovu implementaciju.

2.1. Opis algoritma

Sam rad algoritma može se podijeliti u dva dijela; faza lokalnog podudaranja (*local matching phase*) i faza globalnog podudaranja (*global matching phase*). Njima pretходи pretprocesiranja kojim se ulazni podaci podešavaju za rad algoritma.

Algoritam na ulazu prima sekvencu ciljnog genom T (*target genom sequence*) i sekvencu referentnog genoma R (*reference genome sequence*). U rad algoritma se kreće s pretpostavkom kako su ove dvije sekvence međusobno slične. Također prije početka sva mala slova prebacuju se u velika, te se pamte pozicije malih slova ciljnog genoma T .

2.1.1. Faza lokalnog podudaranja

Kako je prethodno pretpostavljeno da je sličnost T i R velika, algoritam započinje traženjem lokalnih podudaranja. Ulazni podaci T i R se na početku faze lokalnog podudaranja dijele u segmente $(t_1, t_2, \dots, t_n$ i $r_1, r_2, \dots, r_m)$. Svi segmenti osim posljednjeg su duljine L , t_n i r_m mogu biti duljine L ili manje.

Efikasan način na koji algoritam traži sličnosti među parovima su hash tablice (vrijeme pretrage je $O(1)$). Dakle za trenutni par (t_i, r_i) napravi se hash tablica H temeljem segmenta r_i . Parsiramo segment r_i na dijelove veličine k , takve k -dugačke

stringove zovemo k-mer-ovi. Hash funkcija prima k-mer i računa njegov hashcode. Na taj način svaka stavka u hash tablici H je povezana s barem jednim k-mer-om. Konkretno u tablicu H na poziciju hashcode trenutnog k-mera segmenta r_i zapisuje se početna pozicija tog k-mera, p_1 . Zatim se konstruira idući k-mer koji je pomaknut od prethodnog za jednu poziciju u r_i . Taj postupak se ponavlja dok se ne prođe ukupno $L - k + 1$ k-mera u r_i . Postoji mogućnost da su dva ili više k-mera unutar r_i ista, odnosno da na izlazu hash funkcije daju isti hashcode. Tada se na poziciji tog hashcoda u tablici H stvara lista pozicija p .

Jednom kada je formirana hash tablica H na temelju k-mera segmenta r_i uzima se prvi k-mer u t_i kojeg dalje oslovljavamo k-mer'. Za njega se računa hashcode' i provjerava postoji li stavka u tablici H na poziciji hashcode'. Ukoliko postoji znači da je prisutno podudaranje segmenata barem duljine k između t_i i r_i , tj. k-mer' je jednak barem jednom k-meru. Kako je u tablici H pohranjena informacija p_i (početak k-mera u segmentu r_i) od te pozicije započinje se računati duljina poklapanja. Ta duljina je barem k , ali može biti i veća. Povećavanje duljine zaustavlja se u trenutku kada se naiđe na prvi znak koji je različit.

U slučaju da je k-mer' jednak više k-mera (stavka hashcode' pokazuje na listu pozicija) potrebno je proći po svim tim k-merima i usporediti ih s k-mer' i tako pronaći duljinu najduljeg poklapajućeg segmenta. Ukoliko su dva takva segmenta iste duljine bira se onaj segment čija je početna pozicija bliža krajnjoj poziciji prvog prethodnog poklapanja T i R .

Jednom kada se na temelju k-mer i k-mer' pronašao najdulji poklapajući podniz pamti se njegova pozicija i duljina (informacije se zapisuju u posrednu datoteku (*intermediate file*)). Također se pamti prvi nepodudarani znak iza tog podniza. Postupak se ponavlja kontinuiranim pomakom za jednu poziciju u paru (t_i, r_i) . Ako bi se dogodio slučaj gdje k-mer' ne pronalazi nikakva poklapanja u hash tablici, sprema se prvi znak k-mer' u posrednu datoteku.

U posrednoj datoteci tako postepeno nastaje skup parova (p, l) koji su odvojeni znakovima koji se razlikuju između T i R (broj takvih znakova između navedenih parova je minimalno 1).

Također moguća je situacija gdje se nakon cijelog prolaska za par (t_i, r_i) ne nađe nikakvo poklapanje k-mer' s vrijednostima u tablici H . Tada je potrebno smanjiti duljinu k na k' . Ako ni tada nije pronađeno poklapanje, par (t_i, r_i) , odnosno cijeli ti segmenti se spremaju u posrednu datoteku. Ukoliko bi se ovakav slučaj ponavljao sadržaj posredne datoteke bio bi sve veći i veći te je to čini težom za sažimanje. U tom slučaju prestajemo s traženjem lokalnih podudaranja i započinjemo s traženjem

globalnih podudaranja.

2.1.2. Faza globalnog podudaranja

U fazi globalnog podudaranja nije potrebno segmentirati ulaze T i R . Naprotiv, segmentiranje nije poželjno jer tako povećavamo vjerojatnost poklapanja. Opcija traženja globalnog podudaranja uzima se u obzir tek kada su zadovoljeni određeni kriteriji. Dakle zbog svojih performansi ipak se preferira traženje lokalnih podudaranja.

Glavni kriterij po kojem se prelazi u fazu globalnog podudaranja je ako je broj spremljenih znakova prešao unaprijed definiranu veličinu T_1 i kada je broj segmenata s takvim brojem znakova prešao definiranu veličinu T_2 . Također treba napomenuti kako se među tim odstupajućim znakovima ne uzima u obzir znak N jer on po definiciji predstavlja mogućnost bilo koje baze A, C, G ili T. Drugi kriterij zbog kojeg se prelazi na fazu globalnog podudaranja je ako je veličina ulaznog genoma manja od veličine jednog segmenta.

Za fazu globalnog podudaranja uz prebacivanje malih slova u velika potrebno je eliminirati sva pojavljivanja znaka N i zapamtiti njihovu poziciju u posrednoj datoteci. Na temelju R tvori se hash tablica H_G . Računa se hashcode' za k-mer' u T i traži poklapanje u tablici H_G .

Nakon cijelog postupka radi se kompresija posredne datoteke nekim od postojećih algoritama sažimanja. Na taj način kada je potrebno napraviti dekompresiju odnosno rekonstruirati ulazni T zapravo je potrebno napraviti samo dekompresiju posredne datoteke i na temelju nje i referentne sekvence R rekonstruirati ciljnu sekvencu T .

2.2. Implementacija

Vlastita implementacija algoritma napisana je u programskom jeziku C++. Program je testiran u Linux operacijskom sustavu. U nastavku je objašnjen rad algoritma na malom testnom primjeru.

2.2.1. Kompresija

Program za kompresiju na ulazu prima: ciljni genom, referentni genom i destinaciju za datoteku na izlazu kompresije. Ciljni genom pohranjen je u datoteci target.fa, a referentni genom u datoteci referent.fa.

>target.fa small target genom

```
taAAGGTTTATAttCCCAGGTAACaaACGAACCGGGGGNN  
CCCCCAAAAAATTTTTTNNACACATCATACTACATCANNA
```

>reference.fa small reference genom

```
TAAAGGTTTATATTCCCAGGTGGTACCGTAACGCAGCGAC  
CCCGGGATAGATGATCGACCTGCGATCATGCTACTAGCTA
```

Program koristi posrednu datoteku kako bi pohranio sve potrebne informacije za rekonstrukciju ciljnog genoma. U prvom redu pohranjeni su metapodaci iz datoteke ciljnog genoma. Drugi red pamti broj znakova po liniji ciljnog genom. Treći red pamti pozicije malih slova u ciljnom genomu. Naime kako bi algoritam efikasnije radio bolje je smanjiti veličinu abecede te se zato sva mala slova pretvaraju u velika. Pozicije malih slova se pamte kao parovi početne pozicije relativne na prethodnu i broja malih slova u nizu. Dva para su odvojena znakom ";". Dakle 0 2; 11 2 interpretirali bismo kao da prvi niz malih slova počinje na poziciji 0 u ciljnom genomu te ih je ukupno 2, a slijedeći niz malih slova počinje pomakom za 11 pozicija, odnosno na poziciji $11 + 2 = 13$ te sadrži također 2 mala slova. Četvrti red posredne datoteke ovisan je o tipu pretraživanja koje se koristi. Kod globalnog pretraživanja pamti pozicije znakova N i to na isti način kao i za mala slova jer je za potrebe pretrage potrebno izbaciti pojavljivanja znaka N iz ciljnog genoma. Kod lokalnog pretraživanja četvrti red posredne datoteke je prazan, odnosno algoritam provodi pretragu bez potrebe za brisanjem znaka N.

S obzirom da se u ovom primjeru radi o malom ciljnom i referentnom genomu algoritam bi proveo globalno pretraživanje jer je veličina cijelog dokumenta manja od veličine jednog segmenta (30000). No, kako bismo pobliže objasnili rad algoritma pretpostavimo da je veličina jednog segmenta 60.

Nakon pretprocesiranja započinje lokalno pretraživanje podudaranja. Ciljni i referentni genom se dijele na segmente definirane veličine. Za svaki segment se generira hash tablica. Interno je u pitanju C++ mapa čiji ključevi su hashcodovi k-mera, a vrijednosti liste pozicija. Prolaskom po k-merima trenutnog segmenta ciljnog genoma računaju se hashcodovi i provjerava postoje li u hash tablici tog segmenta. Ukoliko postoji prolazi se po listi kandidata poklapanja u referentnom genomu te se uspoređuju referentni i ciljni genom znak po znak. Sve dok postoje poklapanja povećava se varijabla duljine. Dolaskom do prvog odstupanja postupak se zaustavlja i ažurira se duljina najduljeg poklapanja ukoliko je trenutno poklapanje veće od prethodno najvećeg. Uko-

liko nije bilo poklapanja hashcoda k-mera ciljnog genoma i hashcodova u hash tablici prelazi se na idući k-mer.

Nakon usporedbe po svim segmentima u posrednu datoteku zapisuju se parovi (p, l) koji objašnjavaju poklapanje ciljnog i referentnog genoma u l znakova od pozicije p u referentnom genomu ili konkretni znakovi koji predstavljaju odstupanja ciljnog i referentnog genoma.

Zapis u posrednoj datoteci nakon lokalnog pretraživanja poklapanja je slijedeći:

```
>target.fa small target genom
```

```
40
```

```
0 2;11 2;11 2;
```

```
0,20
```

```
AACAAACGAACCGGGGGNNCCCCCAAAAATTTTTTNNACACATCA  
TACTACATCANNA
```

Globalno pretraživanje poklapanja odvija se samo ako je zastavica *global* postavljena na *true*. Zastavica poprima vrijednost *true* ukoliko je duljina ciljnog genoma manja od veličine jednog segmenta ili ukoliko je broj pohranjenih znakova u segmentu veći od t_1 ($t_1 = 0.5$) te broj takvih zaredanih segmenata veći od t_2 ($t_2 = 4$).

Globalno pretraživanje poklapanja interno radi kao i lokalno pretraživanje poklapanja, ali pretragu čini nad cijelom ulaznom sekvencom umjesto nad segmentima.

Zapis u posrednoj datoteci nakon globalnog pretraživanja podudaranja je slijedeći:

```
>target.fa small target genom
```

```
40
```

```
0 2;11 2;11 2;
```

```
38 2;18 2;19 2;
```

```
0,20
```

```
AACAAACGAACCGGGGGCCCCCAAAAATTTTTTACACATCATACTACA  
TCAA
```

Na kraju postupka kompresije posredna datoteka se zapakira pomoću alata 7zip.

2.2.2. Dekompresija

Program za dekompresiju na ulazu prima: referentni genom, sažetu datoteku koju je program kompresije generirao i destinaciju za datoteku na izlazu dekompresije.

Sažetu datoteku se prvo raspakira i zatim učitava kao string. Također se pohranjuje referentni genom kako bi mogli u sljedećem koraku odraditi rekonstrukciju.

Funkcija za rekonstrukciju prima referentni genom i posrednu datoteku kako bi na izlazu generirala ciljni genom. Radi prolaz po podacima iz posredne datoteke. Ukoliko je u pitanju par (p, l) ciljni genom, odnosno string koji predstavlja ciljni genom gradi tako što preuzima l nukleotida iz referentnog genoma od pozicije p . U slučaju kada se radi o konkretnim nukleotidima u posrednoj datoteci jednostavno ih nadodaje na string ciljnog genoma. Jednom kada je ovaj prolaz završen rekonstrukcija poziva funkciju za podešavanje potrebnih znakova. Dakle ukoliko je originalni genom na ulazu u program kompresije imao mala slova ili na mjestu nukleotida vrijednost N, funkcija podešava te znakove.

Jednom kada je rekonstrukcija dovršena ciljni genom se pohranjuje u .fa datoteku pri tome pazeći na specifičnosti zapisa FASTA formata (za formatiranje se koristi dužina zapisana u posrednoj datoteci). Izlazna datoteka pohranjuje se u direktoriju resultsd unutar data direktorija. Dakle sadržaj .fa datoteke nakon dekompresije je sljedeći:

```
>target.fa small target genom  
taAAGGTTTATAttCCCAGGTAACaaACGAACCGGGGGNN  
CCCCCAAAAAATTTTTTNNACACATCATACTACATCANNA
```

3. Rezultati

4. Zaključak

Zaključak.

5. Literatura

- [1] gen. <https://www.enciklopedija.hr/natuknica.aspx?ID=21571>, . Last accessed date: 2023-05-13.
- [2] What is a genome? <https://www.yourgenome.org/facts/what-is-a-genome/>, . Last accessed date: 2023-05-13.
- [3] Top 8 compression software you can't miss in 2023. <https://videoconverter.wondershare.com/compress/top-compression-software.html>. Last accessed date: 2023-05-13.
- [4] Wei Shi, Jianhua Chen, Mao Luo, i Min Chen. High efficiency referential genome compression algorithm. *Bioinformatics*, 35(12):2058–2065, 2019.

6. Sažetak

U ovom radu opisan je rad algoritma za sažimanje genoma, SCCG. Objasnjeno je zašto se primjenjuje poseban algoritam, a ne neki od već postojećih alata za sažimanje. Opisana je implementacija algoritma u programskom jeziku C++. Na kraju su prezentirani rezultati same implementacije.