SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**SCCG ALGORITAM**

Dora Bilić Pavlinović, Marija Kompar

Zagreb, lipanj 2025.

**1 Opis algoritma**

Razvoj tehnologija sljedeće generacije sekvenciranja (Next Generation Sequencing, NGS) doveo je do eksponencijalnog rasta količine genomskih podataka, s desetljećima pohranjene informacije u obliku višestrukih terabajta FASTA ili FASTQ datoteka. Dok je cijena i brzina sekvenciranja ubrzano rasla, prijenos, pohrana i obrada ovih ogromnih nizova postali su ključno usko grlo u genetici i biomedicinskim istraživanjima [1]. Tradicionalni alati za kompresiju (gzip, bzip2, LZMA) ne iskorištavaju specifične karakteristike bioloških nizova – mala abeceda, visoki stupanj ponavljanja i evolucijske varijacije – zbog čega postižu znatno slabije omjere kompresije na genomskim podacima nego na općenitim datotekama [2]. Generički algoritmi tretiraju genom kao proizvoljan niz znakova, ne uzimajući u obzir da različiti uzorci jedne vrste (ili iste osobe) dijele >99 % identičnih baza. Referencijska kompresija koristi poznatu referentnu sekvencu za opisivanje ciljne sekvence samo kroz razlike: pozicije i duljine točnih podudarnosti te "literale" za varijante. Time se postiže dramatično smanjenje veličine izlaznih datoteka, često na nekoliko desetaka megabajta za ljudski genom od ~3 GB.

Jedan od prvih znanstvenih prijedloga upućenih na problem referencijske kompresije potječe iz 2012. godine, kad su Chern i suradnici predložili mapiranje ciljnih baza na referencu, a zatim entropijsko kodiranje dobivenog niza razlika. U međuvremenu, Deorowicz i suradnici razvili su GDC 2, koji učinkovitije komprimira velike zbirke genomskih skupova (>1000 ljudskih genoma) s omjerom do 9500× te brzinom obrade >200 MB/s. Ovi radovi postavili su temelj za specijalizirane pristupe koji iskorištavaju ponavljanja i sličnosti među genomima.

ERGC (Efficient Referential Genome Compression) predstavlja rani praktični alat iz 2015. koji koristi greedy alignment za pronalaženje maksimalnih točnih podudarnosti (maximal exact matches, MEMs) između ciljne i referentne sekvence. Svaki MEM se zapisuje kao par (pozicija, duljina), a ne-podudarni dijelovi spremanju se kao literali ili pomoću edit-skripti [1]. Eksperimenti na stvarnim genomskim skupovima pokazali su da ERGC često nadmašuje tadašnje metode u omjeru kompresije i brzini, posebno kada sekvence sadrže mješovite velike i male znakove.

HiRGC (High-speed and High-ratio Referential Genome Compression), objavljen 2017., uvodi 2-bitno kodiranje i naprednu raspršenu tabličnu pretragu temeljenu na tzv. „*greedy-matching“* pristupu. Umjesto gradnje globalne strukture za cijelu referencu, HiRGC koristi optimizirane raspršene strukture koje omogućuju vrlo brzo nalaženje MEM-ova, uz brzine 2,9 puta brže od konkurencije. Algoritam je stabilan i robustan na različite reference te je dokazan na osam ljudskih genoma i dodatnih 100 iz projekta 1000 genoma [1].

SCCG (Strategically Combined Compression Genome) iz 2019. unosi dinamičko prebacivanje između dvije faze pretraživanja: lokalno i globalno podudaranje. Kod lokalnog podudaranja, ulazne sekvence se dijele na fiksne segmente, svaki s vlastitom raspršenom tablicom, što omogućava brzo uočavane kratkih, bliskih podudarnosti. S druge strane, u slučaju da lokalno podudaranje ne dostigne prag duljine ili je nedovoljno isplativo zbog udaljenosti od prethodne podudarnosti, koristi se globalno podudaranje pomoću globalne raspršene strukture nad cijelom referencom za pronalaženje najduljeg mogućeg MEM-a. Pri svakoj poziciji algoritam procjenjuje duljinu i relativnu udaljenost podudarnosti prije prelaska na globalnu fazu, čime se optimizira omjer kompresije bez značajnog gubitka brzine [1]. Izlaz se kodira kroz apsolutne i delta-kodirane parove (pozicija, duljina), a literali se uključuju za sve nepodudarne dijelove. Time SCCG postiže bolje omjere od ERGC i HiRGC na ljudskim genomima (~3 GB) uz vrijeme kompresije od ~18 min.

SCCG algoritam predstavlja značajan iskorak u dinamičkoj strategiji pretraživanja, kombinirajući brzinu lokalnih pretraga s dubinom globalnih, što rezultira superiornim kompresijskim omjerom i prihvatljivim vremenom izvršavanja. Ovaj uvod pruža teoretski i praktični okvir za daljnje proučavanje i implementaciju SCCG-a u seminarima i istraživačkim projektima.

**2 Koraci SCCG algoritma**

**2.1 Obrada (Preprocessing)**

U prvom koraku algoritma, uređuju se znakovi iz sekvence ciljanog i referentnoga genoma. Iz referentne sekvence uklanjaju se svi znakovi osim {A, C, G, T}, koji predstavljaju dušične baze u DNA (adenin, citozin, gvanin i timin). Također, u obje sekvence mala slova se pretvaraju u velika.

Referentna sekvenca (R): Rezultat:

A c r g T C C G a o T A A C A C G T C C G A T A A C

Ciljana sekvenca (T): Rezultat:

C t g G A C G A A T C G C T G G A C G A A T C G

**2.2 Izgradnja lokalne raspršene strukture (Local Matching Indeks)**

U drugom koraku, referentna i ciljana sekvenca dijele se u blokove fiksne duljine, s n brojem baza. Cilj je ubrzati pretragu podudarnosti tako da, umjesto skeniranja cijele referentne sekvence, brzo pronađemo kratke “lokalne” podudarnosti unutar uskog prozora. Kad se gledaju dijelovi ciljnih baza koji ostaju u okviru istog bloka, pretražuje se samo taj blok, što je znatno brže.

Na primjer, odaberemo li duljinu bloka od 6 baza, referentna sekvenca dijeli se u dva bloka/segmenta:

Blok 1: A C G T C C

Blok 2: G A T A A C

Sljedeći dio je gradnja lokalne raspršene tablice. Za svaki referentni blok gradimo zasebnu raspršenu-tablicu svih podnizova duljine k. Na primjer, za k = 3, podnizovi su:

Blok 1: „ACG“ [1], „CGT“ [2], „GTC“ [3] i „TCC“ [4]

Blok 2: „GAT“ [1], „ATA“ [2], „TAA“ [3] i „AAC“ [4]

Iz ovih podnizova grade se dvije lokalne raspršene tablice, koje sadrže podatak o podnizu i njegovoj poziciji u bloku.

|  |  |
| --- | --- |
| Podniz | Pozicija u bloku |
| „ACG“ | 1 |
| „CGT“ | 2 |
| „GTC“ | 3 |
| „TCC“ | 4 |

Tablica 1. Lokalna raspršena tablica za prvi blok referentne sekvence

|  |  |
| --- | --- |
| Podniz | Pozicija u bloku |
| „GAT“ | 1 |
| „ATA“ | 2 |
| „TAA“ | 3 |
| „AAC“ | 4 |

Tablica 2. Lokalna raspršena tablica za drugi blok referentne sekvence

Isti postupak primjenjuje se za ciljanu sekvencu.

Blok 1: C T G G A C

Blok 2: G A A T C G

|  |  |
| --- | --- |
| Podniz | Pozicija u bloku/segmentu |
| „CTG“ | 1 |
| „TGG“ | 2 |
| „GGA“ | 3 |
| „GAC“ | 4 |

Tablica 3. Lokalna raspršena tablica za prvi segment ciljane sekvence

|  |  |
| --- | --- |
| Podniz | Pozicija u bloku/segmentu |
| „GAA“ | 1 |
| „AAT“ | 2 |
| „ATC“ | 3 |
| „TCG“ | 4 |

Tablica 4. Lokalna raspršena tablica za drugi segment ciljane sekvence

**2.3 Pretraga lokalnih podudaranja**

Pretraga lokalnih podudaranja počinje s prvim blokom ciljane (T) i referentne (R) sekvence:

Blok 1 (T) = C T G G A C

Blok 1 (R) = A C G T C C nismo izlazile preko blokova ? eventalno smo kasnije spajale ali ne ovako

|  |  |  |  |
| --- | --- | --- | --- |
| i (u bloku 1) | podniz = T[i..i+2] | Postoji u tablici? | Akcija |
| 1 | „CTG“ | NE | nema podudaranja |
| 2 | „TGG“ | NE | nema podudaranja |
| 3 | „GGA“ | NE | nema podudaranja |
| 4 | „GAC“ | NE | nema podudaranja |
| 5 | **„ACG“** | **DA** | proširi |

Tablica 5. Pretraga lokalnih podudaranja na prvom bloku

Za podniz iz ciljane sekvence „ACG“ pronađeno je podudaranje na referentnoj poziciji 1 i pretraga se proširuje:

|  |  |  |  |
| --- | --- | --- | --- |
| offset | T[5 + offset] | R[1 + offset] | Podudaranje? |
| 0 | A | A | DA |
| 1 | C | C | DA |
| 2 | G | G | DA |
| 3 | A | T | NE – zaustavi |

Tablica 6. Proširenje lokalne pretrage

Rezultat je lokalno podudaranje (p = 1, l = 3) koje pokriva pozicije iz T 5-7.

Isti postupak ponavlja se za drugi blok ciljane i referentne sekvence.

Blok 2 (T): G A A T C G

Blok 2 (R): G A T A A C

|  |  |  |  |
| --- | --- | --- | --- |
| i (u bloku 2) | podniz = T[i+6..i+8] | Postoji u tablici? | Akcija |
| 1 | „GAA“ | NE | nema podudaranja |
| 2 | „AAT“ | NE | nema podudaranja |
| 3 | „ATC“ | NE | nema podudaranja |
| 4 | „TCG“ | NE | nema podudaranja |

Tablica 7. Pretraga lokalnih podudaranja na drugom bloku

Nema nijednog lokalnog podudaranja duljeg od k = 3.

**2.4 Izgradnja globalne raspršene strukture**

Globalna pretraga se koristi za sve nepokrivene pozicije (ili nakon što lokalna faza nije uspjela), iste korake radimo na cijelom R, ali uz puno širi dohvat, što naravno troši više vremena, ali osigurava da nijedna podudarnost ne promakne. Iz svih mogućih podnizova u referentnoj sekvenci gradi se globalna tablica H s duljinom podniza k = 3.

R = A C G T C C G A T A A C

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| podniz | ACG | CGT | GTC | TCG | CCG | CGA | GAT | ATA | TAA | AAC |
| pozicija | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] |

Tablica 8. Globalna raspršena tablica za referentnu sekvencu

**2.5 Pretraga globalnih podudaranja**

Znakovi koji se ne poklapaju pohranjuju se „doslovno“ u izlazni tok podataka, bez kompresije. U SCCG algoritmu sve ne-poklapajuće baze (ili blokovi baza) ostaju kao “literali” pa ih moramo kasnije učitati točno onakve kakve jesu.

|  |  |  |  |
| --- | --- | --- | --- |
| index | T[index..index+2] | Nalazi li se u H? | Akcija |
| 1 | CTG | NE | literal=“C“, index = 2 |
| 2 | TGG | NE | literal=“T“, index = 3 |
| 3 | GGA | NE | literal=“G“, index = 4 |
| 4 | GAC | NE | Literal=“G“, index = 5 |
| 5 | **ACG** | **DA** | proširi |
| offset = 0: A-A |  |  |  |
| offset = 1: C-C |  |  |  |
| offset = 2: G-G |  |  |  |
| offset = 3: A-T; prekid; l = 3 |  |  |  |
| index = index + l + 1 = 9 |  |  |  |
| 9 | ATC | NE | literal=“A“, index = 10 |
| 10 | TCG | NE | literal=“T“, index = 11 |
| - | preostali (11-12) | - | literali=“C“, “G“ |

**Ref: hg18 tar hg19**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hg18** | **Zip tar file(bytes)** | **Unzip tar file** | **Zip MB** | **Unzip MB** |
| **Chr14** | 28944448 | 108495964 | 27,60358 | 103,4698 |
| **Chr15** | 26685221 | 102345701 | 25,44901 | 97,60447 |
| **Chr16** | 25643209 | 90603807 | 24,45527 | 86,40652 |
| **Chr17** | 25113607 | 80350244 | 23,9502 | 76,62796 |
| **Chr18** | 24547240 | 77639504 | 23,41007 | 74,0428 |
| **Chr19** | 17605180 | 65087892 | 16,78961 | 62,07265 |
| **Chr20** | 19496195 | 63684691 | 18,59302 | 60,73445 |
| **Chr21** | 11230132 | 47883217 | 10,70989 | 45,66499 |
| **Chr22** | 11303833 | 50685268 | 10,78018 | 48,33724 |
| **ChrY** | 8265731 | 58928420 | 7,882815 | 56,19852 |

**Zadnja dva stupca su samo preracunati**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hg19** | **Zip tar file(bytes)** | **Unzip tar file** | **Zip MB** | **Unzip MB** |
| **Chr14** | 28970334 | 109496538 | 27,62826 | 104,424 |
| **Chr15** | 26828094 | 104582027 | 25,58526 | 99,73719 |
| **Chr16** | 25667827 | 92161856 | 24,47875 | 87,8924 |
| **Chr17** | 25139792 | 82819122 | 23,97517 | 78,98247 |
| **Chr18** | 24574571 | 79638800 | 23,43614 | 75,94948 |
| **Chr19** | 17606811 | 60311570 | 16,79116 | 57,5176 |
| **Chr20** | 19513342 | 64286038 | 18,60937 | 61,30794 |
| **Chr21** | 11549785 | 49092500 | 11,01473 | 46,81826 |
| **Chr22** | 11327826 | 52330665 | 10,80306 | 49,90641 |
| **ChrY** | 8276338 | 60561044 | 7,892931 | 57,75551 |

**Zadnja dva stupca su samo preracunati**

**Preformanse ( crveno znaci da smo morali globalno- globalno prelazimo ako 4 segmenta zaredom imaju preko pola alela koji nisu matchani)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Brzina kodiranja**  **(nanosekunde)** | **Brzina dekodiranja**  **(nanosekunde)** | **Velicina final filea(bytes)** | **Velicina zippanog final filea?? Msm da ovo trebamo** | **Brzina kodiranja**  **(sekunde)** | **Brzina dekodiranja**  **(sekunde)** | **Velicina final filea(MB)** | **Velicina zippanog final filea(MB)** |
| **Chr14** | 112107628035 | 3984368305 | 1668101 | 555365 | 112,1076 | 3,984368 | 1,590825 | 0,529637 |
| **Chr15** | 238051801809 | 2252510565 | 942358 | 359468 | 238,0518 | 2,252511 | 0,898703 | 0,342815 |
| **Chr16** | 54206116040 | 2141435618 | 950737 | 360381 | 54,20612 | 2,141436 | 0,906693 | 0,343686 |
| **Chr17** | 231317784532 | 1948501305 | 1085513 | 399092 | 231,3178 | 1,948501 | 1,035226 | 0,380604 |
| **Chr18** | 70806188962 | 2221656260 | 748066 | 295176 | 70,80619 | 2,221656 | 0,713411 | 0,281502 |
| **Chr19** | 166728349634 | 1678167704 | 661237 | 257547 | 166,7283 | 1,678168 | 0,630605 | 0,245616 |
| **Chr20** | 76052094972 | 1488268841 | 933314 | 337454 | 76,05209 | 1,488269 | 0,890078 | 0,321821 |
| **Chr21** | 69430508960 | 972444741 | 1174707 | 360334 | 69,43051 | 0,972445 | 1,120288 | 0,343641 |
| **Chr22** | 83393693180 | 1151595146 | 434767 | 168623 | 83,39369 | 1,151595 | 0,414626 | 0,160811 |
| **ChrY** | 19754493958 | 1280390996 | 509870 | 160627 | 19,75449 | 1,280391 | 0,48625 | 0,153186 |

**Zadnja 4 stupca su samo preracunati**

**Kako runnamo :**

g++ -o SCCGC SCCGC.cpp

g++ -o SCCGD SCCGD.cpp

g++ -o testing testing.cpp

 ./SCCGC chr14\_tar.fa chr14\_ref.fa

 ./SCCGD chr14\_ref.fa

 ./testing chr14\_tar.fa

**O mom ubuntu:**

**16Gb RAM**

**Ubuntu ( 64-bit)**

**6 CPU jezgri**

**(okrivi svu sporost na to sto je virtualka hahaha)**

**Literatura**

[1] Wei Shi , Jianhua Chen , Mao Luo , Min Chen (2019), *High efficiency referential genome compression algorithm*, Bioinformatics, 35, 12

[2] Bobbie Chern, Idoia Ochoa, Alexandros Manolakos, Albert No, Kartik Venkat, Tsachy Weissman (2012), *Reference Based Genome Compression*