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 ciljnog 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

Ciljna 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 ciljna 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 ciljnu 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 ciljne sekvence

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

Tablica 4. Lokalna raspršena tablica za drugi segment ciljne sekvence

**2.3 Pretraga lokalnih podudaranja**

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

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

Tablica 5. Pretraga lokalnih podudaranja na prvom bloku

Isti postupak ponavlja se za drugi blok ciljne 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 6. Pretraga lokalnih podudaranja na drugom bloku

Nema nijednog lokalnog podudaranja duljeg od k = 3. Lokalna faza bila je neuspješna i algoritam prelazi na globalnu fazu pretrage.

**2.4 Izgradnja globalne raspršene strukture**

Globalna pretraga se koristi za sve nepokrivene pozicije (ili nakon što lokalna faza nije uspjela). Neuspjeh lokalne faze definira se kao nedovoljan udio pronađenih podudaranja u segmentu u odnosu na veličinu segmenta. 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 7. 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“ |

Tablica 8. Pretraga globalnih podudaranja

U petom bloku ciljne sekvence pronađeno je podudaranje duljine 3 znaka s referentnim nizom na njegovoj prvoj poziciji. Izlaz iz algoritma je ciljna sekvenca zapisana u opisnom obliku u odnosu na referentnu sekvencu. Svi literali zapisuju se neizmijenjeno, a nizovi za koje su pronađena podudaranja zapisani su delta-kodiranim parovima (pozicija, duljina). Konkretno, za naš primjer, izlaz je sljedeći:

C T G G

(1, 3)

A A T C G

Dekodiranje se obavlja pomoću izlaza algoritma i priložene referentne sekvence za pronalazak niza. Rezultat dekodiranja jednak je izvornom obliku ciljne sekvence.

**3 Rezultati testiranja**

Za testiranje naše implementacije algoritma odabrali smo ljudski genom hg18 kao referentni i hg19 kao ciljni genom. Testirali smo na Ubuntu 64-bitnoj Linux distribuciji s 16GB RAM-a i 6 procesorskih jezgri.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sekvenca | Veličina ZIP datoteke [B] | Veličine raspakirane datoteke [B] | Veličina ZIP datoteke [MB] | Veličina raspakirane datoteke [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 |

Tablica 9. Veličina sekvenci iz referentnog genoma hg18

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sekvenca | Veličina ZIP datoteke [B] | Veličine raspakirane datoteke [B] | Veličina ZIP datoteke [MB] | Veličina raspakirane datoteke [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 |

Tablica 9. Veličina sekvenci iz ciljnog genoma hg19 prije provedbe algoritma

Nakon provedbe algoritma, zabilježili smo performanse poput brzine kodiranja i dekodiranja te veličine izlaznih datoteka. Crvenom bojom označene su sekvence za koje je bila potrebna globalna pretraga, a zelenom bojom označene su one za koje je lokalna pretraga bila dovoljna i uspješna.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Target | Brzina kodiranja [s] | Brzina dekodiranja [s] | Veličina ZIP izlazne datoteke [MB] | Brzina kodiranja [s]  -originalni alg. | Brzina dekodiranja [s]  -originalni alg. | Veličina ZIP izlazne datoteke [MB]  -originalni alg. |
| chr14 | 44,42 | 2,52 | 0,529637337 | 18,846277527 | 1,582689554 | 0,303045 |
| chr15 | 138,68 | 2,37 | 0,342815399 | 69,162941054 | 2,114874293 | 0,312417 |
| chr16 | 65,82 | 2,18 | 0,343686104 | 73,065707469 | 1,866240828 | 0,308821 |
| chr17 | 200,56 | 2,52 | 0,38060379 | 215,703893079 | 1,997895369 | 0,347495 |
| chr18 | 71,45 | 2,40 | 0,28150177 | 68,455126327 | 1,295469763 | 0,255516 |
| chr19 | 102,13 | 1,82 | 0,245615959 | 19,495665838 | 1,556920419 | 0,220589 |
| chr20 | 59,34 | 1,85 | 0,321821213 | 20,173453412 | 1,479047523 | 0,230305 |
| chr21 | 81,46 | 1,52 | 0,343641281 | 12,47287405 | 1,118065203 | 0,322901 |
| chr22 | 114,10 | 1,39 | 0,160811424 | 24,487266952 | 1,203080067 | 0,144554 |
| chrY | 13,65 | 1,69 | 0,153185844 | 9,669978135 | 1,355052895 | 0,076241 |

Tablica 10. Performanse algoritma

Iz priloženih rezultata vidljivo je da je brzina kodiranja/pretrage bila u prosjeku brža za zeleno označene sekvence. Brzina, osim potrebe za globalnom pretragom, ovisi i o veličini sekvence, ali iz usporedbe pretrage nad većim chr14 i manjim chr15 sekvencama, može se zaključiti da je globalna pretraga bitan čimbenik u brzini.

Izlazna datoteka je prethodno opisana verzija ciljne sekvence komprimirane na literale i delta-kodirane zapise (položaj, duljina). Usporedili smo veličine izlaznih datoteka naše implementacije algoritma s originalnom implementacijom. Rezultati naše kompresije u prosjeku odstupaju za +28,5%, što znači da su u prosjeku naše izlazne datoteke 1,285 puta veće, što je dosta približno.

Vrijeme kodiranja u prosjeku nam odstupa od vremena kodiranja originalne implementacije za +180,187%, što znači da nam je u prosjeku izvođenje algoritma kodiranja 2,80187 puta duže.

Vrijeme dekodiranja u prosjeku nam odstupa od vremena dekodiranja originalne implementacije za +31,99%, što znači da nam je u prosjeku izvođenje algoritma dekodiranja 1,3199 puta duže.

|  |  |  |
| --- | --- | --- |
| Sekvenca | Veličina ZIP datoteke [B] | Veličina ZIP datoteke [MB] |
| Test1 | 210 | 0,205078 |
| Test2 | 775 | 0,7568359 |
| Test3 | 8439 | 8,2412109 |
| Test4 | 10191 | 9,952148 |
| Test5 | 16713 | 16,321289 |
| Test6 | 31629 | 30,887695 |
| Test7 | 88761 | 86,68066 |
| Test8 | 136794 | 133,58789 |
| Test9 | 282331 | 275,71387 |
| Test10 | 553925 | 540,94238 |

**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

**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*