SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1818

**Implementacija progresivne metode poravnanja više sekvenci**

Viktor Pijanec

Zagreb, lipanj 2025.

Sadržaj

[Uvod 1](#_Toc200922534)

[1. Teorijska osnova 2](#_Toc200922535)

[1.1. Višestruko poravnanje sekvenci (MSA) 2](#_Toc200922536)

[1.2. Filogenetska stabla 2](#_Toc200922537)

[1.3. Algoritamski pristupi u MSA 2](#_Toc200922538)

[1.3.1. Egzaktne motode 2](#_Toc200922539)

[1.3.2. Heurističe metode 3](#_Toc200922540)

[1.4. Progresivna metoda poravnanja 3](#_Toc200922541)

[2. Metode i implementacija 4](#_Toc200922542)

[2.1. Opis arhitekture implementiranog algoritma 4](#_Toc200922543)

[2.2. Programske biblioteke i alati 4](#_Toc200922544)

[2.3. Detalji o algoritmu i ključnim funkcijama 5](#_Toc200922545)

[3. Korišteni algoritmi 6](#_Toc200922546)

[3.1. Needleman-Wunsch algoritam 6](#_Toc200922547)

[3.2. UPGMA algoritam 7](#_Toc200922548)

[3.3. Neighbour-Joining algoritam 8](#_Toc200922549)

[4. Eksperimentalna evaluacija 9](#_Toc200922550)

[4.1. Skup podataka BaliBASE 9](#_Toc200922551)

[4.2. Postavke eksperimenta 10](#_Toc200922552)

[4.3. Metode evaluacije 11](#_Toc200922553)

[4.4. Skripta za evaluaciju 11](#_Toc200922554)

[4.5. Primjeri rezultata 13](#_Toc200922555)

[4.6. Analiza rezultata 15](#_Toc200922556)

[5. Rasprava 20](#_Toc200922557)

[5.1. Kvaliteta dobivenog poravnanja 20](#_Toc200922558)

[5.2. Usporedba s ClustalW-om 20](#_Toc200922559)

[5.3. Ograničenja implementirane metode 20](#_Toc200922560)

[5.4. Moguća poboljšanja 21](#_Toc200922561)

[5.5. Primjenjivost u bioinformatici 21](#_Toc200922562)

# Uvod

U današnje doba genomike, analiza bioloških sekvenci postala je temelj razumijevanja evolucijskih odnosa i strukturnih karakteristika živih sustava. Eksponencijalni porast dostupnih DNK, RNK i proteinskih podataka potaknuo je razvoj sofisticiranih računalnih metoda koje omogućuju usporedbu i interpretaciju ovih sekvenci. Višestruko poravnanje sekvenci (Multiple Sequence Alignment - MSA) predstavlja ključnu metodu u bioinformatici koja omogućava otkrivanje obrazaca te rekonstruiranje evolucijskih veza među organizmima.

Implementacija progresivne metode poravnanja nudi učinkovito rješenje za probleme skalabilnosti i računalne zahtjevnosti kod MSA. Ovaj pristup gradi višestruko poravnanje postupnim uparivanjem najsličnijih sekvenci, čime se postiže kompromis između brzine izvršavanja i preciznosti rezultata. Iako je progresivna metoda osjetljiva na inicijalni redoslijed poravnanja, njena jednostavnost primjene i dokazana učinkovitost čine je popularnim izborom u brojnim bioinformatičkim alatima, poput ClustalW.

Cilj ovog rada jest implementacija progresivnog algoritma za višestruko poravnanje sekvenci korištenjem programskog jezika C++ u Linux okruženju. Uspoređujući rezultate dobivene implementacijom s rezultatima alata ClustalW, nastojat će se identificirati prednosti i ograničenja progresivne metode, a evaluacija će biti provedena na javno dostupnom skupu podataka BaliBASE, koji predstavlja izazovan test za preciznost i robusnost algoritma.

Rad je strukturiran tako da prvo daje pregled teoretskih osnova višestrukog poravnanja sekvenci i algoritamskih principa progresivnog pristupa. Nakon toga slijedi detaljan opis implementacije, uključujući arhitekturu koda te korištene biblioteke. Završni dio rada posvećen je analizi rezultata, usporedbi s ClustalW-om te raspravi o mogućim poboljšanjima i smjerovima za daljnji razvoj.

Ovim radom nastoji se produbiti razumijevanje progresivnih metoda u kontekstu višestrukog poravnanja sekvenci te pokazati njihovu primjenjivost u praktičnim istraživačkim scenarijima, čime se doprinosi unapređenju metodologije u obradi i analizi bioloških podataka.

# Teorijska osnova

## Višestruko poravnanje sekvenci (MSA)

Višestruko poravnanje sekvenci predstavlja postupak usklađivanja triju ili više bioloških sekvenci (DNA, RNA ili proteina) radi prepoznavanja sličnosti među njima. Identificirane sličnosti mogu ukazivati na evolucijske odnose ili strukturne karakteristike proteina. MSA je ključna metoda u bioinformatici jer omogućuje analizu regija u sekvencama, što je od velike važnosti za filogenetska istraživanja, prepoznavanje genetskih funkcija i razumijevanje molekularne evolucije.

Primjena MSA obuhvaća širok spektar bioloških istraživanja, uključujući:

Filogenetsku analizu, gdje poravnanje sekvenci ukazuje na izgled evolucijskih stabala.

Predviđanje strukture proteina, jer neke regije proteina mogu ukazivati na to kako protein izgleda u 3D prostoru.

Annotaciju genoma, gdje se analizom homolognih sekvenci mogu identificirati sačuvani geni u različitim vrstama.

## Filogenetska stabla

Filogenetska stabla predstavljaju grafički prikaz evolucijskih odnosa među sekvencama, genima ili vrstama. Ona modeliraju pretpostavljene zajedničke pretke te pokazuju koliko su pojedine sekvence srodne na temelju sličnosti u njihovim biološkim svojstvima, najčešće sekvencama aminokiselina ili nukleotida.

U kontekstu višestrukog poravnanja sekvenci, filogenetska stabla koriste se kao vodilni mehanizam za određivanje redoslijeda poravnanja. Filogenetsko tablo definira kojim redoslijedom se sekvence ili već poravnane grupe trebaju spajati. Takvo stablo omogućuje hijerarhijsko (progresivno) poravnanje, gdje se najprije spajaju najsličnije sekvence, a potom sve udaljenije.

Postoje različite metode za konstrukciju filogenetskih stabala no dvije se najčešće koriste u izradi MSA algoritma, UPGMA i Neighbour-Joining algoritam. Ispravna konstrukcija stabla od ključne je važnosti za točnost konačnog višestrukog poravnanja, jer pogreške u ranijim fazama poravnanja mogu negativno utjecati na sve kasnije korake.

## Algoritamski pristupi u MSA

Postoji nekoliko računalnih pristupa za višestruko poravnanje sekvenci, pri čemu se metode općenito dijele na ekzaktne i heurističke algoritme.

### Egzaktne motode

Ekzaktni algoritmi, poput dinamičkog programiranja korištenog u Needleman-Wunsch i Smith-Waterman metodama, osiguravaju optimalno poravnanje sekvenci. Međutim, zbog eksponencijalne složenosti, njihova primjena na velike podatkovne skupove, kakvima baratamo kod MSA, njihova uporaba postaje nepraktična.

### Heurističe metode

S obzirom na računalne zahtjeve ekzaktnih metoda, razvijene su heurističke strategije koje omogućuju približno, ali učinkovito poravnanje sekvenci. Među njima se ističe progresivna metoda poravnanja, koja se temelji na postupnom građenju MSA počevši od najbliže srodnih sekvenci.

## Progresivna metoda poravnanja

Progresivne metode za višestruko poravnanje sekvenci slijede hijerarhijski pristup, gdje se poravnanje konstruira korak po korak. Ova metoda obično uključuje sljedeće faze:

Izračun matrice sličnosti (ili udaljenosti) između sekvenci pomoću algoritama za poravnanje dviju sekvenci.

Izgradnja filogenetskog stabla, često korištenjem UPGMA ili Neighbor-Joining algoritma.

Iterativno dodavanje sekvenci prema filogenetskom stablu, gdje se prvo poravnaju najsličnije sekvence, a zatim postupno dodaju ostale.

Progresivne metode, poput one korištene u alatu ClustalW, nude kompromis između preciznosti i računalne učinkovitosti. Međutim, osjetljive su na pogreške u početnim koracima poravnanja, jer se kasnije faze oslanjaju na inicijalne odluke.

# Metode i implementacija

## Opis arhitekture implementiranog algoritma

Implementacija progresivnog algoritma za višestruko poravnanje sekvenci (MSA) temelji se na heurističkom pristupu postupnog grupiranja sekvenci prema stupnju sličnosti. Algoritam slijedi standardni radni tijek progresivne metode, koji uključuje tri ključne faze:

Izračun matrice sličnosti – Parno poravnanje svih sekvenci i izračun sličnosti.

Generiranje vodilnog stabla – Korištenjem jednostavne verzije UPGMA ili Neighbour-Joining algoritma.

Progresivno dodavanje sekvenci – Iterativno proširivanje poravnanja prema uputama stabla.

Glavna programska struktura podijeljena je na sljedeće komponente:

Parser podataka – Čita ulazne sekvence iz FASTA formata.

Modul za poravnanje dvije sekvence – Implementira algoritam Needleman-Wunsch za inicijalna parna poravnanja.

Modul za izgradnju vodilnog stabla – Generira stablo temeljem matrice sličnosti.

Modul za spajanje više sekvenci – Implementira algoritam sličan Needleman-Wunsch-u koji spaja dvije grupe sekvenci.

Glavni modul aplikacije – Koordinira izvršavanje svih koraka algoritma.

## Programske biblioteke i alati

Implementacija algoritma realizirana je u C++, koristeći sljedeće standardne biblioteke i alate:

STL (Standard Template Library) – Za rad s nizovima, setovima i mapama.

Bioparser – Omogućuje parsiranje ulaza iz FASTA formata.

ClustalW – Za usporedbu dobivenih poravnanja s referentnim rezultatima.

Izbor programskog jezika C++ omogućava optimalnu računalnu učinkovitost, dok primjena naprednih podatkovnih struktura osigurava brzo izvođenje ključnih koraka algoritma.

## Detalji o algoritmu i ključnim funkcijama

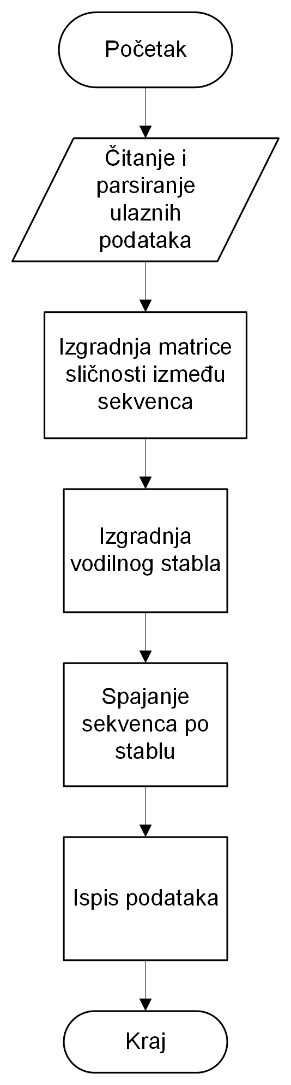
Ključne funkcije unutar implementacije uključuju:

Align(): Izračunava udaljenosti između dvije sekvence – koristi se za izgradnju matrice međusobnih udaljenosti.

BuildTree(): Kreira vodilno stablo na temelju sličnosti sekvenci koje govori kojim redoslijedom spajamo sekvence.

BuildByTree(): Primjenjuje progresivnu metodu poravnanja na sekvence prema poretku iz stabla.

Dijagram toka koda može se vidjeti na slici (Slika 2.1).



Slika 2.1 Dijagram toka algoritma

# Korišteni algoritmi

## Needleman-Wunsch algoritam

Nedleman-Wunsch algoritam ključni je alat za globalno poravnanje dviju sekvenci korištenjem dinamičkog programiranja. Algoritam izgrađuje dvodimenzijalnu tablicu gdje svaka ćelija u tablici sadrži optimalni rezultat poravnanja do te pozicije.

Algoritam se sastoji od tri faze:

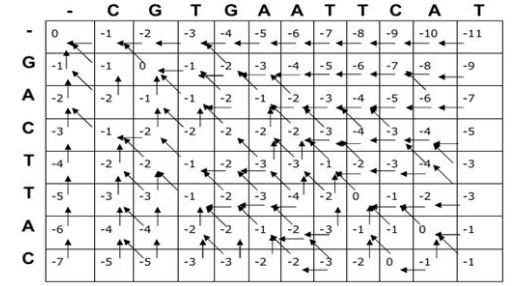
Inicijalizacija matrice – postavljanje početnih vrijednosti u prvi redak i stupac matrice

Popunjavanje matrice – za svaku ćeliju računa se maksimalna/minimalna (ovisi računa li se minimalna udaljenost ili maksimalna sličnosti) vrijednost iz tri moguća smjera:

* Dijagonalno – podudaranje/nepodudaranje elemenata na indeksima i i j u prvoj i drugoj sekvenci
* Vertikalno – praznina u prvoj sekvenci
* Horizontalno – praznina u drugoj sekvenci

Vraćanje – praćenje optimalnog puta od donjeg desnog kuta tablice do gornjeg lijevog kuta

Primjer poravnanja dviju sekvenci korištenjem Needleman-Wunsch algoritma vidimo na slici (Slika 3.1)



Slika 3.1 Poravnanje dviju sekvenci koristeći Needleman-Wunsch algoritam

Needleman-Wunsch algoritam se često koristi u bioinformatici te se u MSA algoritmima koristi za dobivanje inicijalno poravnanja između svaka dva para sekvenci. Nakon toga se ti rezultati koriste za izradu binarnog stabla koje će opisivati redoslijed spajanja sekvenci.

## UPGMA algoritam

UPGMA algoritam (Unweighted Pair Group Method with Arithmetic Mean) predstavlja hijerarhijsku metodu koji se koristi za konstrukciju binarnog stabla temeljenog na matrici udaljenosti ili sličnosti između sekvenci. Algoritam iterativno spaja najbliže sekvence dok ne formira potpuno binarno stablo.

Ključni koraci algoritma uključuju:

Inicijalizaciju – svaka sekvenca se tretira kao zaseban klaster s početnom veličinom 1

Pronalaženje minimalne udaljenosti/maksimalne sličnosti – za pronalazak dva najsličnija klastera koja sljedeća spajamo

Spajanje dvaju najsličnijih klastera u jedan

Ažuriranje matrice udaljenosti pomoću formule:

gdje su i veličine tih klastera.

Prikaz vodilnog stabla kojeg moj algoritam dobije kao rezultat prikazan je na slici (Slika 3.2)

Slika na kojoj se prikazuje tekst, Font, snimka zaslona, knjiga

Sadržaj generiran uz AI možda nije točan.

Slika 3.2 Prikaz dobivenog stabla

## Neighbour-Joining algoritam

Neighbour-Joining (NJ) algoritam je heuristička metoda za konstrukciju filogenetskih stabala iz matrice udaljenosti. Za razliku od UPGMA algoritma, NJ ne pretpostavlja konstantnu brzinu evolucije što ga čini pogodnijim za realne biološke podatke gdje brzina mutacija može varirati među granama stabla.

Osnovna ideja NJ algoritma je pronalazak para klastera koji, kada se spoje, minimiziraju ukupnu duljinu stabla. Nakon što se identificira takav par, formira se novi klaster, a matrica udaljenosti se ažurira na temelju prosječne udaljenosti između tog novog klastera i preostalih sekvenci. Proces se ponavlja dok ne ostanu samo dva klastera, koji se tada spajaju u korijen stabla.

Jedna od prednosti NJ algoritma je njegova fleksibilnost i mogućnost konstrukcije nelinearnih stabala, čime se često dobivaju precizniji rezultati kod divergentnih sekvenci. U mojoj implementaciji, Neighbour-Joining algoritam koristi se kao alternativa UPGMA algoritmu za izgradnju vodilnog stabla, no kako sam se ja fokusirao na UPGMA algoritam i na temelju njega radio ostatak programa SP-score i TC-score vrijednosti kod korištenja NJ algoritma su nešto manje u usporedbi s UPGMA algoritmom kad se provede analiza na podacima iz BaliBASE skupa podataka.

# Eksperimentalna evaluacija

## Skup podataka BaliBASE

Za evaluaciju implementiranog algoritma korišten je BaliBASE, referentni skup podataka namijenjen testiranju algoritama za višestruko poravnanje sekvenci. BaliBASE sadrži biološki relevantne proteinske sekvence podijeljene u više različitih grupa. BaliBASE također sadrži referentna poravnanja svih sekvenci napravljena na temelju njihovih 3D struktura s kojima se kasnije uspoređuju poravnanja programa koji ispitujemo. Ovaj skup podataka omogućava objektivnu procjenu preciznosti algoritama za MSA. Primjer jedne datoteke koja sadrži sekvence namijenjene za poravnanje može se vidjeti na slici (Slika 3.1), a primjer njezinog referentnog poravnanja na slici (Slika 3.2).

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.1 Primjer tfa dototeke iz BaliBASE-a

Slika na kojoj se prikazuje tekst, snimka zaslona, Font, broj

Sadržaj generiran uz AI možda nije točan.

Slika 4.2 Primjer referentnog poravnanja datoteke sa slike (Slika 3.1)

Prikaz broja datoteka po grupama u BaliBASE-u prikazan je u grafikonu (Grafikon 4.1), a prikaz ukupnog broja sekvenci i prosječne dužine sekvence u pojedinoj grupi u tablici (Tablica 4.1)

Grafikon 4.1 Prikaz broja datoteka po grupama u Bali-BASE-u

Tablica 4.1 Broj sekvenci i prosječna duljina sekvence po grupama

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RV11 | RV12 | RV20 | RV30 | RV40 | RV50 |
| Ukupan broj sekvenci | 522 | 792 | 3738 | 3790 | 1354 | 885 |
| Prosječna duljina sekvence | 275.19 | 345.38 | 314.95 | 320.86 | 465.50 | 422.12 |

## Postavke eksperimenta

Eksperimentalna evaluacija obuhvaćala je sljedeće korake:

1. Priprema ulaznih podataka – Odabir testnih skupova sekvenci iz BaliBASE-a.

2. Izvršavanje algoritma – Pokretanje implementirane metode na testnim sekvencama.

3. Usporedba s ClustalW-om – Analiza rezultata u odnosu na rezultate dobivene ClustalW.

4. Analiza performansi – Mjerenje vremena izvođenja i računalnih zahtjeva algoritma.

## Metode evaluacije

Za procjenu kvalitete poravnanja korišteni su sljedeći metrički pokazatelji:

* SP-score (Sum-of-Pairs Score) – Mjeri konzistentnost poravnanja u odnosu na referentne sekvence.
* TC-score (Column Score) – Procjenjuje točnost poravnanja na razini pojedinih stupaca u odnosnu na referentne sekvence.
* Vrijeme izvođenja – Mjerenje efikasnosti algoritma u usporedbi s ClustalW-om.

Rezultati su dobiveni programom bali\_score koji dolazi s BaliBASE skupom podataka.

## Skripta za evaluaciju

U nastavku, Kod 4.1, prikazana je skripta koju sam koristio za izračun prosječne vrijednosti SP-score-a i TC-score-a dobivenog mojom implementacijom, korištenjem UPGMA algoritma i korištenjem Neighbour-Joining algoritma, te vrijednosti dobivene alatom ClustalW za sve datoteke u pojedinoj grubi BaliBASE-a.

#!/usr/bin/bash

sp\_score\_uk=0

tc\_score\_uk=0

sp\_score\_nj\_uk=0

tc\_score\_nj\_uk=0

sp\_score\_clustal\_uk=0

tc\_score\_clustal\_uk=0

brojac\_uk=0

for tfa\_file in $1/\*.tfa; do

base\_name=$(basename "$tfa\_file" .tfa)

echo "$base\_name"

build/msa "$tfa\_file" > /dev/null

bb3\_release/bali\_score\_src/bali\_score "${1}/${base\_name}.msf" izgraden\_msa.txt > pom.txt

sp\_score=$(grep -oP 'SP score=\s\*\K\d+\.\d+' pom.txt)

tc\_score=$(grep -oP 'TC score=\s\*\K\d+\.\d+' pom.txt)

sp\_score\_uk=$(echo "$sp\_score\_uk + $sp\_score" | bc)

tc\_score\_uk=$(echo "$tc\_score\_uk + $tc\_score" | bc)

build/msa -n "$tfa\_file" > /dev/null

bb3\_release/bali\_score\_src/bali\_score "${1}/${base\_name}.msf" izgraden\_msa.txt > pom.txt

sp\_score=$(grep -oP 'SP score=\s\*\K\d+\.\d+' pom.txt)

tc\_score=$(grep -oP 'TC score=\s\*\K\d+\.\d+' pom.txt)

sp\_score\_nj\_uk=$(echo "$sp\_score\_nj\_uk + $sp\_score" | bc)

tc\_score\_nj\_uk=$(echo "$tc\_score\_nj\_uk + $tc\_score" | bc)

clustalw "$tfa\_file" > /dev/null

sed -i '1a\\//' ${1}/${base\_name}.aln

bb3\_release/bali\_score\_src/bali\_score ${1}/${base\_name}.msf ${1}/${base\_name}.aln > pom.txt

sp\_score=$(grep -oP 'SP score=\s\*\K\d+\.\d+' pom.txt)

tc\_score=$(grep -oP 'TC score=\s\*\K\d+\.\d+' pom.txt)

sp\_score\_clustal\_uk=$(echo "$sp\_score\_clustal\_uk + $sp\_score" | bc)

tc\_score\_clustal\_uk=$(echo "$tc\_score\_clustal\_uk + $tc\_score" | bc)

brojac\_uk=$((brojac\_uk + 1))

done

echo "uk: SP=$sp\_score\_uk/$brojac\_uk, TC=$tc\_score\_uk/$brojac\_uk"

echo "uk\_nj: SP=$sp\_score\_nj\_uk/$brojac\_uk, TC=$tc\_score\_nj\_uk/$brojac\_uk"

echo "uk\_clustal: SP=$sp\_score\_clustal\_uk/$brojac\_uk, TC=$tc\_score\_clustal\_uk/$brojac\_uk"

Kod 4.1 – skripta za evaluaciju programa na jednoj grupi iz BaliBASE-a

## Primjeri rezultata

U nastavku se nalazi prikaz rezultata na datoteci BB11013.tfa. Prikaz rezultata dobivenih mojom implementacijom nalazi se na slici (Slika 4.3), prikaz rezultata dobivenih mojom implementacijom, ali korištenjem Neighbour-Joining algoritma za izgradnju vodilnog stabla nalazi se na slici (Slika 4.4), prikaz rezultata dobivenih korištenjem alata clustalw na istoj datoteci dan je na slici (Slika 4.5) te prikaz referentnog poravnanja danog u samom BaliBASE skupu podataka dan je na slici (Slika 4.6).

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.3 Rezultati dobiveni mojom implementacijom

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.4 Rezultati dobiveni mojom implementacijom, korištenjem Neighbour-Joining algoritma

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.5 Rezultati dobiveni alatom ClustalW

Slika na kojoj se prikazuje tekst, snimka zaslona, Font

Sadržaj generiran uz AI možda nije točan.

Slika 4.6 Referentno poravnanje dano u BaliBASE-u

## Analiza rezultata

U nastavku su prikazani grafikoni rezultata dobiveni SP-score-ova i TC-score-ova na svakoj grupi podataka. Podaci o grupi RV11 prikazani su na grafikonu (Grafikon 4.2), podaci od grupi RV12 na grafikonu (Grafikon 4.3), grupi RV20 na grafikonu (Grafikon 4.4), grupi RV30 na grafikonu (Grafikon 4.5), grupi RV40 na grafikonu (Grafikon 4.6) te podaci o grupi RV50 na grafikonu (Grafikon 4.7).

Grafikon 4.2 SP-score i TC-score na RV11 skupu podataka

Grafikon 4.3 SP-score i TC-score na RV12 skupu podataka

Grafikon 4.4 SP-score i TC-score na RV20 skupu podataka

PROMJENII

Grafikon 4.5 SP-score i TC-score na RV30 skupu podataka

Grafikon 4.6 SP-score i TC-score na RV40 skupu podataka

Grafikon 4.7 SP-score i TC-score na RV50 skupu podataka

Grafikon 4.8 SP-score i TC-score na svim podacima

PROMJENIII

Stupci UPGMA i Neighbour-Joining su moje implementacije algoritma dobiveni korištenjem UPGMA i Neighbour-Joining algoritama za izgradnju vodilnog stabla a stupac ClustalW je stupac dobiven korištenjem alata ClustalW.

Dobiveni rezultati ukazuju da moje implementacije nisu toliko dobre koliko alat ClustalW no nisu ni jako daleko od njega po točnosti dobivenih rezultata.

//Dodati za brzinu izvođenja rezultate

# Rasprava

## Kvaliteta dobivenog poravnanja

Dobiveni rezultati pokazuju da implementirani progresivni algoritam za višestruko poravnanje sekvenci postiže solidnu točnost u usporedbi s referentnim alatima poput ClustalW. Međutim, kvaliteta poravnanja značajno ovisi o inicijalnom izboru parnih poravnanja i izgradnji filogenetskog stabla. U slučajevima gdje su sekvence divergentne, greške u ranim fazama postupka mogu dovesti do propagacije netočnosti kroz cijelo poravnanje, što je poznati nedostatak progresivnih metoda.

U scenarijima gdje sekvence imaju visoku stupanj konzerviranosti, algoritam uspješno prepoznaje i očuva ključne strukturne elemente, potvrđujući njegovu učinkovitost u takvim slučajevima. Međutim, kod sekvenci s niskim stupnjem sličnosti, mogućnost generiranja biološki relevantnog poravnanja znatno je umanjena.

## Usporedba s ClustalW-om

Usporedba s ClustalW-om ukazuje na nekoliko ključnih prednosti i nedostataka implementiranog algoritma:

* Brzina izvođenja – kod manjih skupova sekvenci obje metode završavaju gotovo instantno, no kod najvećih skupova ClustalW završava znatno ranije od mojih implementacija MSA algoritma.
* Preciznost poravnanja – ClustalW u većini slučajevima postiže bolja poravnanja, zahvaljujući optimiziranim koracima pri izračunu težina filogenetskog stabla, no moje implementacije nisu puno lošije od njega.

## Ograničenja implementirane metode

Najznačajnije ograničenje progresivnog poravnanja je osjetljivost na početne odluke pri formiranju filogenetskog stabla. Budući da naknadne faze ovise o ranije poravnanim sekvencama, pogreške se mogu širiti kroz cijeli postupak, smanjujući biološku vjerodostojnost rezultata.

Dodatno, implementacija ne uključuje iterativne poboljšanja poravnanja, što je jedna od tehnika koju koriste moderni algoritmi za poboljšanje rezultata nakon inicijalnog poravnanja. Integracija takvih strategija mogla bi znatno unaprijediti preciznost bez značajnog povećanja računalnih zahtjeva.

## Moguća poboljšanja

Kako bi se poboljšala kvaliteta algoritma, nekoliko mogućih unaprjeđenja moglo bi se implementirati:

* Iterativna optimizacija poravnanja – Nakon inicijalnog poravnanja, može se provesti dodatna analiza kako bi se poboljšale nepravilnosti.
* Korištenje naprednijih metoda izračuna sličnosti - moglo bi točnije odrediti slične sekvence te tako pomoći u izgradnji točnijeg vodilnog stabla i kao posljedicu smanjiti propagaciju pogrešaka.
* Paralelizacija izvođenja – Implementacija paralelnih procesiranja mogla bi znatno ubrzati izvođenje algoritma na velikim skupovima podataka kad je paralelizacija moguća.
* Uključivanje statističkih modela: Integracija modela poput Hidden Markov Models (HMM) mogla bi poboljšati točnost poravnanja u varijabilnim regijama.

## Primjenjivost u bioinformatici

Unatoč određenim ograničenjima, implementirana metoda pokazuje potencijalnu primjenjivost u analizi bioloških podataka. Daljnji razvoj algoritma, uključujući integraciju naprednih heuristika i optimizacija poravnanja, mogao bi proširiti njegovu primjenu u bioinformatičkim istraživanjima.

Zaključak

Literatura

<https://vlab.amrita.edu/index.php?sub=3&brch=274&sim=1431&cnt=1>

Sažetak

Summary

Skraćenice

Privitak