SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br.4190

Rekonstrukcija filogenetskog stabla metodom maksimalne uštede uz razgranaj-ograniči optimizaciju

Ana Marija Selak

Zagreb, svibanj 2015.

Zahvaljujem Ani Bulović na pomoći pri izradi ovog rada.

Zahvaljujem Mili Šikiću na svemu.

Sadržaj

1.Uvod

Život na zemlji počeo je prije otprilike 3.8 milijardi godina. Svi organizmi koji su se do danas razvili, pa tako i oni izumrli, razvili su se iz prve žive stanice. Genetički materijal prve žive stanice prenosio se na njezine potomke. Potomci su se razvijali i udaljavali od svojih predaka zbog raznih mutacija i rekombinacija koje su se događale. Upravo ove mehanizme kojima su se od prve žive stanice razvili svi ostali organizmi opisuje teorija evolucije. Kako bi se prikazali evolucijski odnosi među organizmima uvodi se grana znanosti, filogenija. Filogenija odnose među organizmima prikazuje i objašnjava filogenetskim stablom.

Filogenetsko stablo prikazuje evolucijske odnose između različitih vrsta za koje se pretpostavlja da imaju zajedničke pretke. Sastoji se od unutarnjih i vanjskih čvorova, pri čemu se svaki čvor smatra taksonomskom zajednicom, s tim da se unutrašnja čvorišta smatraju hipotetskom taksonomskom zajednicom ukoliko se odgovarajuće vrste ne mogu promatrati. Osnovna podjela filogenetskih stabala jest na one sa korijenom i na one bez korijena. Filogenetska stabla bez korijena predstavljaju odnose među taksonomskim zajednicama koje se nalaze na listovima stabla i ne zahtijevaju poznavanje glavnog pretka svih ostalih taksonomskih zajednica u stablu. Ne stvaraju se pretpostavke o precima određenih čvorova i povezanost više čvorova s jednim zajedničkim označava njihovu sličnost, a ne nužno podrijetlo od zajedničkog pretka. Filogenetska stabla s korijenom, s druge strane, prikazuju smjer evolucijskog procesa određujući odnos između pretka i potomaka, odnosno, imaju pretpostavljen smjer vremena.

Za rekonstrukciju filogenetskog stabla razvijen je čitav niz metoda.

Za većinu metoda ovaj problem je NP-težak problem[4] i zahtijeva puno vremena i memorije za izračun zbog čega se često koriste heurističke metode koje smanjuju broj mogućih filogenetskih stabala.

U ovom radu implementirana je izgradnja filogenetskog stabla korištenjem metode maksimalne uštede (engl.*maximum parsimony*), koja po principu Occamove oštrice zahtjeva da rekonstruirano stablo sadrži minimalan broj mutacija potreban za objašnjenje podataka. Metoda maksimalne uštede spada u metode koje koriste matricu obilježja. Matrica obilježja svakoj taksonomskoj

zajednici, za svako obilježje, pridružuje odgovarajuću vrijednost. Metoda maksimalne uštede također spada u metode koje za izgradnju filogenetskog stabla koriste optimalan kriterij.

Budući da ova metoda spada u klasu NP teških problema, kako bi se omogućio izračun za netrivijalan broj vrsta, korištena je razgranaj-ograniči (engl.*branch and bound*) optimizacija. Udaljenosti među vrstama rekonstruirane su pomoću višestrukog poravnanja (engl.*multiple sequence alignment*) među nukleotidnim i proteinskim sljedovima.

U drugom poglavlju ukratko je opisana druga skupina metoda, metode koje se temelje na matrici udaljenosti. U trećem i četvrtom poglavlju dan je detaljan opis metode maksimalne uštede i optimizacije razgranaj-ograniči. U petom poglavlju opisano je implementirano programsko rješenje.

2.Metode temeljene na matrici udaljenosti

Za izgradnju filogenetskog stabla postoje dvije kategorije metoda. Prva kategorija metoda jesu metode koje koriste matricu udaljenosti. Matrica udaljenosti izgrađuje se na temelju poravnanja između svake dvije taksonomske zajednice (engl.*pairwise alignment*). Svaka udaljenost u matrici predstavlja opaženu količinu promjene između dvije taksonomkse zajednice s obzirom na njihovog najbližeg zajedničkog pretka. Neke od metoda koje spadaju u tu kategoriju su *Neighbour Joining, UPGMA i Fitch Margoliash algoritam*[1][2][3]. Za skup sekvenci, za koje se filogenetsko stablo izgrađuje, stvara se matrica udaljenosti koja sadrži broj razlika u nukleotidima između svake dvije sekvence. Iz ove matrice gradi se filogenetsko stablo koje spaja čvorove stabla, gdje čvorovi stabla predstavljaju sekvence. Čvorovi se spajaju s unutarnjim čvorovima tako da se spoje one sekvence između kojih je udaljenost najmanja. Prednost ovih algoritama je njihova brzina i manja memorijska složenost u odnosu na druge metode[8]. Glavni nedostaci algoritama, koji koriste matricu udaljenosti, jesu ti što se za izračun udaljenosti uvijek koriste samo dvije sekvence kao i činjenica ne korištenja informacije koje su skrivene u raspodjeli nukleotida već oslanjanje samo na udaljenosti među biološkim sljedovima.

3.Metode temeljene na matrici znakova

Druga kategorija metoda su metode koje se temelje na matrici znakova. Najpoznatiji algoritmi koji spadaju u kategoriju metoda temeljenih na matrici znakova jesu algoritam maksimalne uštede i algoritam najveće vjerojatnosti (engl.*maximum likelihood*)[3].

Ulazni podaci za metode koje se temelje na matrici obilježja nizovi su vrijednosti koji predstavljaju obilježja svake taksonomske zajednice. Jedan redak matrice predstavlja jednu taksonomsku zajednicu, dok su stupci vrijednosti obilježja. Obilježje nije precizno definirano, a obično označava svojstvo na temelju kojeg se vrste promatraju. Obilježja mogu biti morfološka, molekularna, genetska, fizička ili ponašajna. Obilježja se dijele u diskretna stanja na temelju kojih se vrste klasificiraju. Diskretna stanja mogu na primjer biti za obilježje boje očiju,zelena, plava i smeđa boja, za obilježje načina prehrane, mesojedi, biljojedi, svejedi, za obilježje ponašanja, agresivno, miroljubivo, indiferentno ponašanje.

Za molekularne podatke, kakvi će se koristiti u ovom radu, matrica znakova je zapravo matrica višestrukog poravnanja sljedova, čiji je primjer prikazan tablicom 3.1.

Stanja svakog obilježja u matrici predstavljena su nukleotidima (A, T, C, G), kod DNA sljedova, ili simbolima aminokiselina, kod proteinskih sljedova. Kod poravnatih DNA sekvenci obilježje je pozicija u sekvenci, dok je stanje obilježja jedan od 4 nukleotida, a može biti i nepoznato ukoliko na toj poziciji nismo odredili nukleotid.

Tablica 3.1 Primjer matrice obilježja

	1	**2**	**3**	**4**
S1	A	T	C	T
S2	-	T	G	T
S3	A	C	G	T
S4	A	G	A	T

Svaki redak u tablici 3.1 predstavlja sekvencu, dok stupci predstavljaju nukleotide sekvenci na određenim pozicijama. Pozicije su obilježja, a stanja tih obilježja su

nukleotidi ili '-'. Stanja obilježja još se nazivaju znakovima i mogu nositi različite težine, dok neslaganje određenih znakova također može rezultirati različitim cijenama. Naime, može se zadati matrica koja definira cijene između svaka dva znaka i te cijene ne moraju biti jednake. Primjer takve matrice, za nukleotidne sljedove, dan je tablicom 3.2.

Tablica 3.2 Primjer različitih cijena

	A	**T**	**C**	**G**	**-**
A	0	1	1	2	1
T	1	0	2	3	1
C	1	2	0	1	1
G	2	3	1	0	1
-	1	1	1	1	0

Različite težine koriste se kada je zapaženo da su određene promjene manje vjerojatne i rjeđe od nekih drugih promjena pa je shodno tome i broj evolucijskih koraka između takvih promjena veći. Nakon što je matrica znakova definirana kreće se u izradu filogenetskog stabla koja se temelji na istoj matrici.

Metoda maksimalne uštede, kao i metoda najveće vjerojatnosti, uz neke metode temeljene na matrici udaljenosti, također spadaju i u metode izgradnje filogenetskog stabla koje se temelje na određenom optimalnom kriteriju. Svaka od metoda definira funkciju koja za zadano stablo vraća cijenu tog stabla. Na temelju cijene koju vrati funkcija, različite se topologije filogenetskih stabala mogu uspoređivati i rangirati. Loša strana ovakvog pristupa jest što potraga za optimalnom topologijom dugo traje. Naime, metode temeljene na optimalnom kriteriju računaju cijenu svakog stabla s obzirom na zadani kriterij i traže ono stablo koje vrati najmanju cijenu. Kako broj mogućih topologija filogenetskog stabla eksponencijalno raste, već za 20 vrsta postaje veći od broja atoma u svemiru i problem pronalaska optimalnog stabla postaje nerješiv u realnom vremenu[4]. Ovakav problem implicira veliku vremensku i memorijsku složenost i broj sekvenci za koje se filogenetsko stablo može izraditi u realnom vremenu postaje ograničen. Unatoč ograničenosti i kompleksnosti rješenja, ovakav je pristup matematički precizan. Metode koje koriste optimalni kriterij garantiraju da će izgrađeno stablo biti optimalno s obzirom na kriterij. Osporavanje ovakvih

metoda može se temeljiti jedino na osporavanju optimalnog kriterija koji te metode koriste.[3]

4.Metoda maksimalne uštede

Metoda maksimalne uštede spada u metode koje koriste matricu znakova kao i u skupinu metoda koje se temelje na optimalnom kriteriju. Temelji se na principu Occamove oštrice koji je osmislio William of Occham oko 1320. godine. Princip Occamove oštrice može se opisati jednom rečenicom "Ukoliko postoji više teorija koje predviđaju isto, odaberite onu najjednostavniju.". Primjena ovog principa na izgradnju filogenetskog stabla odnosi se na pretpostavku da je ono filogenetsko stablo koje ima najmanji broj mutacija između vrsta ujedno i ispravno. Iako evolucija vrsta nije inherentno takva da se može pretpostaviti najmanji broj mutacija između vrsta, odnosno evolucija ne slijedi uvijek princip maksimalne uštede, razna znanstvena istraživanja podupiru ovaj princip i metodu[9]. Evolucija u kojoj se između vrsta događaju manje promjene poželjnija je i vjerojatnija od one u kojoj su promjene veće i kompliciranije. Metoda maksimalne uštede uvijek odabire ono stablo koje najbolje objašnjava evoluciju između vrsta, odnosno ono stablo koje minimizira broj karakteristika koje se ne mogu objasniti zajedničkim precima između vrsta. Odabrano filogenetsko stablo objašnjava evoluciju zadanih podataka s najmanjom količinom evolucijskih promjena između vrsta. Osnovna ideja ove metode prvi put je predstavljena 1970.godine od strane James S. Farrisa i 1971.godine od strane Waltera M. Fitcha[7].

U ovom radu metoda maksimalne uštede izvodila se nad skupom poravnatih DNA molekula. Svaka pozicija unutar sekvence predstavlja znak koji može biti jedan od četiri postojeća nukleotida ili praznina. Ukoliko je na nekoj poziciji u poravnanju umjesto nukleotida bio znak '-', nukleotid za tu poziciju u višestrukom poravnanju nije određen. Usporedba nukleotida s prazninom rezultira cijenom 0 dok usporedba između dva različita nukleotida rezultira cijenom 1 (neslaganje). Ipak, kako su neke mutacije biološki vjerojatnije od drugih, stanja znakova, u ovom slučaju nukleotidi ili znak '-', mogu nositi različite težine. Možemo reći da ukoliko na pripadajućim pozicijama imamo kod jedne vrste nukleotid A, a kod druge nukleotid T, broj evolucijskih koraka između te dvije vrste veći je nego da smo kod

jedne vrste imali na primjer nukleotid C, a kod druge nukleotid G. Razlog tome je što su neke mutacije rjeđe od drugih, kao npr. mutacija A-C, A-T, G-C, G-T i obrnute pa je onda i težina između takva 2 prijelaza veća od drugih težina. U ovom radu korištena je pretpostavka da su sve mutacije između nukleotida jednako vjerojatne što rezultira cijenom 1 prilikom neslaganja bilo koja dva nukleotida na odgovarajućim pozicijama.

Osnovna ideja metode maksimalne uštede može se podijeliti u dva osnovna koraka[3]:

1. Određivanje minimalne cijene topologije stabla koja predstavlja količinu promjene između znakova na nivou cijelog stabla.
2. Pretraživanje svih mogućih topologija stabla s ciljem pronalaženja one topologije stabla koja minimizira ukupnu cijenu stabla.

4.1.Određivanje cijene stabla

Prvi korak metode maksimalne uštede je određivanje cijene stabla. Ovaj je korak jednostavan, te nije vremenski ni memorijski zahtjevan.
Za n taksonomskih zajednica, binarno stablo bez korijena sadrži n vanjskih čvorova, (listova) koji predstavljaju upravo zadane taksonomske zajednice za koje se filogenetsko stablo gradi. Unutrašnjih čvorova ima $n-2$, a broj grana jednak je $2*n-3$. Svaka grana spaja po dva čvora.
Uzmimo da τ predstavlja određenu topologiju nekog stabla iz skupa svih mogućih stabala. Duljina stabla zadana je formulom :

$$L(\tau) = \sum_{j=1}^{N} l_j \tag{4.1}$$

gdje N predstavlja maksimalnu duljinu sekvence (taksonomske zajednice) od zadanih sekvenci, a l_j predstavlja količinu promjene u stanjima znakova za poziciju j u sekvencama. Za binarna stabla formula za l_j glasi:

$$l_j = \sum_{k=1}^{2N-3} c_{a(k),b(k)} \tag{4.2}$$

gdje $a(k)$ i $b(k)$ predstavljaju nukleotide koji se nalaze u čvorovima na krajevima grane k na poziciji j u sekvencama a i b. $C_{a(k),b(k)}$ je broj koji odgovara cijeni promjene između stanja $a(k)$ i $b(k)$. Kao što je navedeno, ta cijena predstavlja težinu određene promjene i može biti različita ovisno o stanjima $a(k)$i b(k). Ukoliko su stanja $a(k)$ i $b(k)$ jednaki nukleotidi tada $C_{a(k),b(k)}$ iznosi 0 za tu granu na poziciji j.

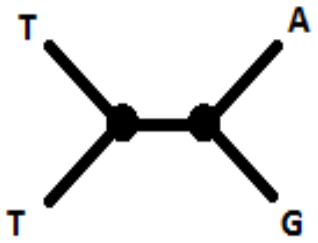
Ideja određivanja l_j, cijene određene topologije za zadanu poziciju j, jest ta da nakon što imamo zadanu topologiju pronalazimo onakav sastav unutrašnjih čvorova koji će dati minimalnu cijenu cijelog stabla koju s tom topologijom možemo postići. Određivanje minimalne cijene topologije ima više različitih pristupa, a neki od njih su takozvani brute-force pristup, Fitch algoritam i dinamičko programiranje.

Brute-force algoritam ispituje cijene svih mogućih stabala za zadanu topologiju. Za unutrašnje čvorove ispituju se sva moguća stanja znakova i svaka kombinacija čini jedno od mogućih stabala zadane topologije. Broj stabala za zadanu topologiju raste eksponencijalno s brojem taksonomskih zajednica, a iznosi r^{T-2} gdje je r broj mogućih stanja znakova ($r = 4$ ukoliko su znakovi nukleotidi, $r = 20$ ukoliko su znakovi aminokiseline), a T broj taksonomskih zajednica. Ova formula lako je objašnjiva. Naime, broj unutrašnjih čvorova za T taksonomskih zajednica jednak je $T - 2$, a kako je broj mogućih stanja za svaku poziciju jednak r onda je i broj različitih kombinacija jednak upravo navedenoj formuli. Broj ukupnih mogućih stabala za sve pozicije jednak je umnošku broja pozicija i broja stabala za svaku poziciju. U ovom radu korišten je upravo ovaj pristup.

Promotrimo sljedeći primjer:

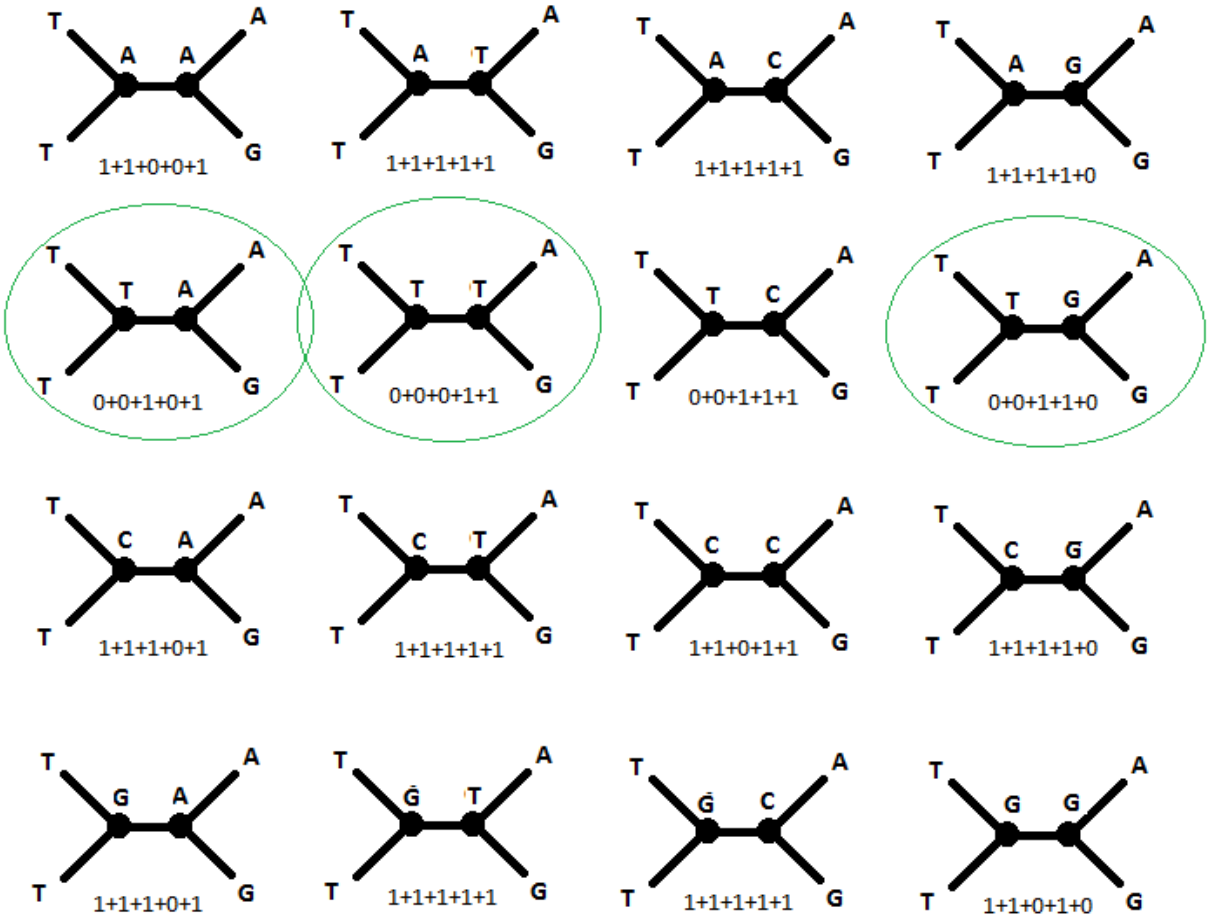
```
              j
W     ...AC**T**TGTCT...
X     ...AC**T**TGTCT...
Y     ...TG**A**AGCCT...
Z     ...GC**G**GGCCT...
```

Pretpostavimo da tražimo minimalnu cijenu stabla za topologiju ((W,Y),(X,Z)) koja je prikazana Slikom 4.1.



Slika 4.1 Topologija stabla

Također pretpostavimo da smo minimalnu duljinu za prvih j-1 znakova već izračunali i da sada računamo minimalnu duljinu za znak j. S obzirom da imamo 4 sekvence u stablu, broj mogućih kombinacija unutrašnjih čvorova za ovu topologiju je $4^{4-2} = 16$. Moguće kombinacije prikazane su na slici 4.2.



Slika 4.2 Različite kombinacije unutrašnjih čvorova

Uz pretpostavku da su težine svih stanja znakova i razlika u stanjima znakova 1, cijena svakog od stabala navedena je ispod slike. Vidimo da je minimalna cijena ovakve topologije jednaka 2 i da je postignuta za 3 kombinacije.

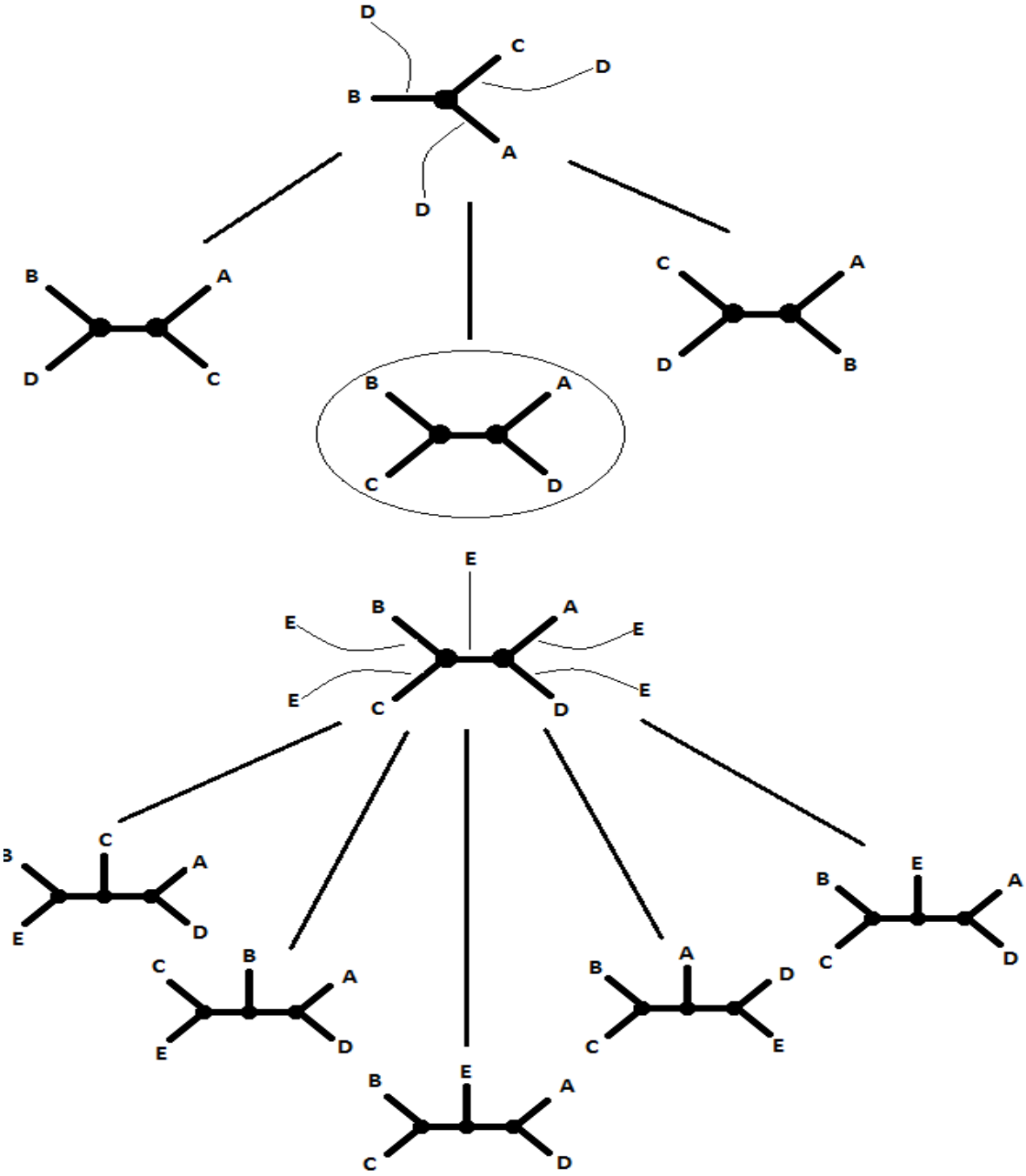
Opisana metoda primjenjuje se za svaki znak (poziciju) u sekvenci i ukupna cijena stabla za neku topologiju jednaka je zbroju cijena stabala pojedinih pozicija.

S obzirom da prethodno opisani postupak zahtjeva ispitivanje velikog skupa stabala, često se koriste alternativni pristupi za izračunavanje minimalne cijene određene topologije stabla, kao što su navedeni Fitch algoritam i dinamičko programiranje.
Fitch algoritam pogodan je za stabla u kojima sva stanja znakova nose jednaku težinu, dok je dinamičko programiranje prilagođeno stablima kod kojih stanja znakova imaju različite težine[3].

4.2 Pretraživanje svih topologija stabla

Prvi korak metode maksimalne uštede jest generiranje svih mogućih topologija stabla. Ovaj korak zahtjeva jako puno vremena i memorije i upravo je zbog njega metoda maksimalne uštede ograničena brojem sekvenci za koje je moguće generirati optimalno filogenetsko stablo.
Jedan od načina pretrage topologija stabala jest generiranje svih mogućih topologija počevši od stabla koje u sebi sadrži tri sekvence. Nove sekvence dodaju se rekurzivno jedna po jedna na svaku od mogućih grana iz prethodnog koraka. Nakon što se doda zadnja sekvenca sve su moguće topologije stvorene i slijedi traženje minimalne cijene stabla svake od topologija.
Primjer generiranja svih topologija prikazan je za 5 sekvenci na slici 4.3.
U prikazanom primjeru početne 3 sekvence predstavljene su slovima A, B i C. Za 3 sekvence postoji samo jedna moguća topologija. Četvrta sekvenca je D i ona se dodaje na broj mogućih mjesta koji je jednak broju grana prethodnog stabla, a to su 3 mjesta. Postoje 3 različite topologije za 4 sekvence jer je broj topologija prethodne razine jednak 1, a broj grana prethodne razine jednak 3. Peta sekvenca je sekvenca E i onda se dodaje na svaku od grana za svaku prethodno stvorenu

topologiju. Kako je broj topologija stabla sa 4 sekvence jednak 3, a broj grana stabla sa 4 sekvence jednak 5 proizlazi da će ukupan broj topologija stabla koje sadrži 5 sekvenci biti 15. Na slici 4.3 prikazano je dodavanje sekvence E samo na jednu od prethodnih topologija (zaokružena). Kada bi dodali i šestu sekvencu broj topologija bio bi 105 (15 topologija iz prethodne razine pomnoženo sa brojem grana stabla od 5 sekvenci).



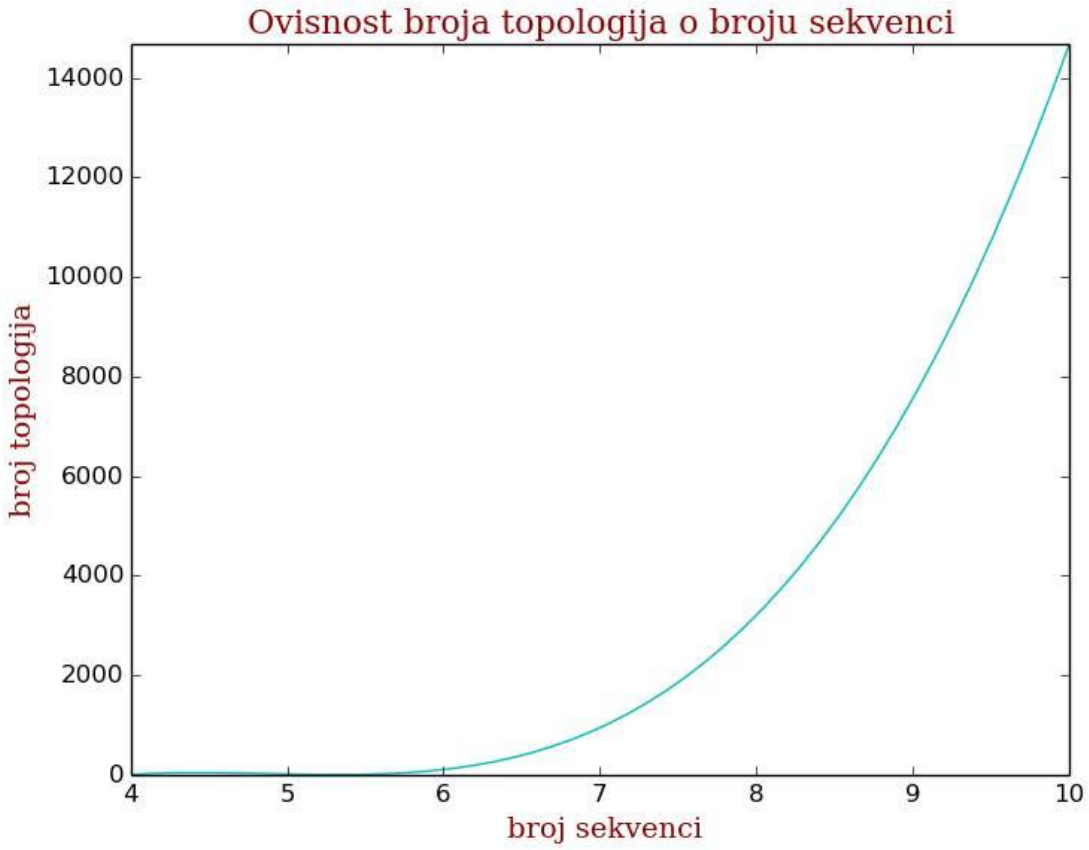
Slika 4.3 Prikaz generiranja svih topologija

Iz opisanog primjera jasno je da vrijedi sljedeća relacija između broja topologija i broja sekvenci za koje se stablo izgrađuje. Relacija je opisana sljedećom formulom:

$$B(t) = \prod_{i=3}^{t}(2i - 5) \tag{4.3}$$

gdje je $B(t)$ broj mogućih topologija, a t je broj sekvenci.

Iz formule je jasno da broj mogućih topologija jako brzo raste i za 20 sekvenci prelazi $2 * 10^{20}$. Ovisnost broja mogućih topologija o broju sekvenci prikazana je na slici 4.4.



Slika 4.4 Ovisnost broja topologija o broju sekvenci

Kako je ovakav pristup poprilično zahtjevan i pretražuje sva moguća stanja često se koristi algoritam optimizacije razgranaj-ograniči (engl.*branch and bound*). Ovakav pristup iskoristiv je za do otprilike 20 sekvenci, ovisno o podacima.

Osnovna ideja algoritma razgranaj-ograniči jest evaluacija svih mogućih topologija uz odbacivanje onih koje sigurno neće dovesti do optimalnog stabla[3].

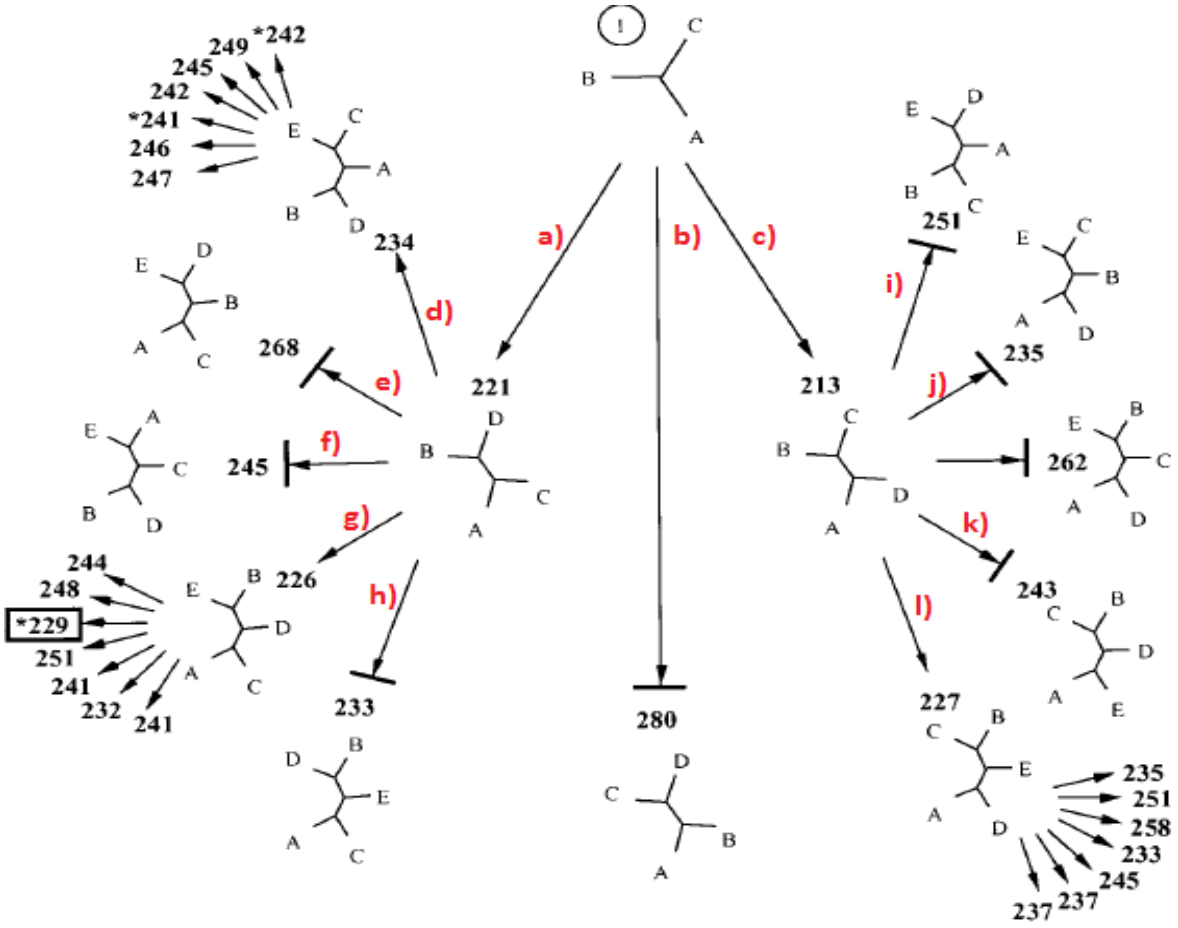
Temelji se na dva osnovna koraka:

1. određivanje gornje granice cijene stabla
2. izgradnja svih topologija uz odbacivanje onih čija je cijena veća od trenutne gornje granice

Kreće se kao i u prethodno opisanom algoritmu. Na početku imamo 3 sekvence i jednu topologiju. Dodaju se ostale sekvence sve dok se ne izgradi jedna topologija koja sadrži sve sekvence. Nakon što je prva takva topologija izgrađena, cijena tog stabla postavlja se kao gornja granica. Algoritam gradi ostale topologije ali sada za svaku topologiju stabla koja ne sadrži još uvijek ukupan broj sekvenci prvo računa cijenu. Nakon što se izračuna minimalna cijena zadane topologije ista se uspoređuje sa trenutnom gornjom granicom. Ako je ta cijena veća za put koji bi se nastavio iz te topologije zaključuje se da nije moguće doći do optimalnog rješenja i taj put se odbacuje. Sa svakom sljedećom topologijom koja se uspije izgraditi tako da sadrži sve sekvence provodi se usporedba trenutne gornje granice i minimalne cijene stabla te topologije. Ukoliko je cijena stabla manja od gornje granice, gornja granica se osvježava i postavlja na novu vrijednost. Algoritam se nastavlja dok se svi mogući putovi u stablu ne evaluiraju. Optimalna topologija jest ona topologija čija je cijena stabla jednaka zadnjoj gornjoj granici koja predstavlja minimalnu moguću cijenu.

Algoritam nije ovisan o optimalnom kriteriju maksimalne uštede i može se koristiti i za druge optimalne kriterije kao i za slične probleme pretraživanja stanja čiji cilj nije nužno izgradnja filogenetskog stabla. Ono što je ključno za ovaj algoritam jest da funkcija koja evaluira neko stablo (optimalni kriterij) ne bude padajuća, odnosno da se s povećanjem broja sekvenci u stablu cijena stabla sigurno poveća ili ostane ista. Kada navedeno ne bi bilo ispunjeno algoritam ne bi mogao zaključiti da neki put sigurno ne vodi optimalnom rješenju jer bi postojala mogućnost da se prilikom dodavanja nove sekvence cijena stabla smanji. Optimalni kriterij maksimalne uštede zadovoljava navedeno.

Primjer ovog algoritma dan je na slici 4.5. U zadanom primjeru broj sekvenci je 6. Kreće se od stabla koje sadrži 3 sekvence (A, B i C) i iz te topologije grade se 3 topologije sa 4 sekvence (a, b i c). Prvo se izgrađuju sva stabla iz topologije stabla *a* dodavanjem sekvence D. Nakon toga put se nastavlja izgradnjom stabala iz topologije *d*. Kako algoritam tim putem nastavlja dobivamo prvo puno stablo čija je cijena 242 i to se postavlja za gornju granicu. Nakon što su izgrađena sva puna stabla iz topologije *d* gornja granica postaje 241. Iz topologije *a* odbacuju se putovi koji idu preko *e* i *f* topologija jer je duljina tih stabala, koja sadrže 5 sekvenci, veća od trenutne gornje granice 241. Algoritam napreduje po istom principu kroz cijeli prostor stanja i na kraju pretraživanja gornja granica je 229 i to je cijena optimalnog filogenetskog stabla za zadani skup sekvenci.



Slika 4.5 Primjer algoritma razgranaj-ograniči

Algoritam razgranaj-ograniči često dolazi u kombinaciji sa heuristikom. Uloga heuristike je izračunavanje manje početne gornje granice kako bi se neki putovi u

izgradnji stabla mogli u ranijim koracima odbaciti. Za heuristiku često se koristi algoritam najbližeg susjeda. Osim heuristike, postoje i druge razvijene metode koje omogućavaju ranije odbacivanje određenih putova.
Ovaj algoritam u većini slučajeva smanjuje prostor pretraživanja te se povećava broj sekvenci za koje je moguće izgraditi optimalno stablo u razumnom vremenu, ali to ne mora uvijek biti slučaj. Prostor stanja, kao i heuristika, mogu biti takvi da se jako mali broj putova može odbaciti, a ponekad se čak i cijeli prostor stanja mora pretražiti. Puno toga u izgradnji optimalnog filogenetskog stabla ovisi o podacima za koje se isto izgrađuje.

4.3 Prednosti i nedostaci

Metoda maksimalne uštede jedna je od popularnijih metoda za izgradnju filogenetskih stabala. Pokazano je da ova metoda daje najbolje rezultate za proteinske sljedove i jedne od najboljih rezultata za DNA sljedove[8]. Metoda je relativno jednostavna i puno brža od metoda koje se temelje na statističkim metodama kao što je metoda najveće izglednosti. Ne zahtjeva komplicirano računanje kao ni velik broj pretpostavki, a svejedno uspijeva u većini slučajeva dati zadovoljavajuće rezultate.

Ipak, postoje i uvjeti u kojima ova metoda podbacuje. Jedan od problema leži u tome što metoda maksimalne uštede polazi od pretpostavke da dvije vrste mogu dijeliti isto stanje određenog znaka (isti nukleotid na istoj poziciji) jedino ukoliko su one genetski povezane, što nije uvijek ispravno. Metoda maksimalne uštede nema svojstvo statističke konzistentnosti, odnosno nema garancije da će se vjerojatnost dobivanja ispravnog filogenetskog stabla povećavati sa dodavanjem podataka. Jedan od primjera uvjeta u kojim metoda podbacuje zove se potaknuto privlačenje (engl.*long branch attraction)*. Takva pojava naziva se još i Felsensteinova zona[6]. Potaknuto privlačenje označava pojavu pogreške prilikom određivanja evolucijske udaljenosti između vrsta kada se za dvije vrste koje su zapravo evolucijski udaljene izvodi zaključak da su one evolucijski bliske. Ova pogreška događa se kada dvije vrste koje su prošle kroz mnoge evolucijske promjene dođu do stanja u

kojime se čine sličnima, što direktno dovodi do pogrešnog zaključka da su te dvije vrste evolucijski bliske[5].

Druga loša strana jest već spomenuta činjenica da je izračunavanje optimalnog filogenetskog stabla ovom metodom NP-težak problem. NP-težak problem implicira da se za određenu količinu vrsta metoda neće završiti u razumnom vremenu kao i to da metoda ima veliku memorijsku složenost.

5.Implementacija

U ovom radu implementirana je metoda maksimalne uštede uz razgranaj-ograniči optimizaciju. Implementacija se temelji na prethodno opisanim idejama uz minimalne preinake.

Programsko rješenje implementirano je u programskom jeziku Java, a kao ulaz koristi se datoteka višestrukog poravnanja sljedova (engl.*multiple sequence alignment*), poput one što ju generira alat ClustalW. Osim poravnatih sekvenci, zadaje se i putanja do direktorija u koji se želi spremiti izlaz programa. Programsko rješenje rekonstruira optimalno filogenetsko stablo za zadani ulazni skup podataka. Kako može postojati više optimalnih topologija, izlaz predstavljaju sve moguće topologije. Topologije su zapisane u Newick formatu. Newick format predstavlja jedan od standardnih formata za zapis stabala i postoje brojni alati koji ovaj format prevode u grafički.

Osim optimalnih stabala programsko rješenje ispisuje na standardni izlaz broj informativnih pozicija i minimalnu cijenu optimalnog filogenetskog stabla.

Programsko rješenje testirano je na različitim skupovima sekvenci i uspoređeno s drugim suvremenim alatima koji rješavaju isti problem. Rezultati su u skladu s očekivanjima.

U nastavku su detaljno opisani koraci algoritma, memorijska i vremenska složenost i strukture podataka koje su korištene.

5.1 Pseudokod

```
begin

informativePositions = getInformativePositions();

for brSeq := 4 to sequences.size() step 1 do

   foreach informativePos do

            Node newTaxa = sequences.get(brSeq);

            foreach tree in posTrees(informativePos) do

                makeNewTrees(tree,newTaxa,newTrees,
                                        topology);
            done

            posTrees.put(informativePos, newTrees);

            if(brSeq == sequences.size() -1)
                 almostFullTrees.put(informativePos,
                                          posTrees);
            end

        done

   done

   foreach topology do

        i=0;
        foreach informativePos do

            if(i==0)

                 buildFullTrees(almostFullTrees
                           (informativePos),fullTrees);
                 minimumCost = getBoundary(fullTrees);
                 continue;

            end

            almostFullCost=countMinimumCost
                         (almostFullTree(informativePos));

            if(almostFullCost > minimumCost)
```

```
                continue;
            end

            buildFullTrees(almostFullTrees(informativePos)
                                ,fullTrees);

            currentCost = getBoundary(fullTrees);

            if(currentCost < minimumCost)
                minimumCost = currentCost;
            end

        end

    end
```

end

Na početku, ulazna datoteka se parsira i algoritam nastavlja izvođenje nad matricom n x m, gdje je n broj sljedova, a m duljina poravnatih sljedova. Svaki stupac predstavlja određenu poziciju u sekvencama i algoritam se izvršava upravo po svakoj poziciji.

Kako bi se postupak ubrzao u poravnanju se promatraju samo takozvani informativni stupci. Informativni stupci su oni za koje vrijede sljedeća dva uvjeta :

1. U tom stupcu postoje barem 2 tipa nukleotida
2. Svaki od tipova nukleotida, koji se pojavljuje u stupcu, zastupljen je u barem 2 slijeda u poravnanju.

Iznimka ova dva uvjeta jest situacija u kojoj u stupcu imamo zastupljeno više tipova nukleotida koji se pojavljuju samo jednom, ali su svi ostali elementi tog stupca neodređeni. Za ovakvu situaciju takav stupac također se uzima kao informativan. Jednostavnije rečeno, neinformativni su oni stupci u kojima su gotovo svi nukleotidi jednaki.
Prvi korak algoritma jest određivanje upravo opisanih informativnih stupaca.
Algoritam nastavlja od četvrte sekvence s pretpostavkom da je ukupan broj sekvenci sigurno veći ili jednak od 4. Kao što je u poglavlju 4.1 navedeno, za određivanje minimalne cijene određene topologije korišten je brute force pristup.

Za svaku topologiju grade se sva moguća stabla, odnosno sve moguće kombinacije unutrašnjih čvorova za svaku od informativnih pozicija. Za svaku sljedeću sekvencu, izuzev za zadnju sekvencu, grade se sva moguća stabla iz onih koje smo u prethodnom koraku izgradili. Nakon što su sva moguća stabla, za svaku topologiju i za sve sekvence, osim zadnje, izgrađena, algoritam se nastavlja koristeći optimizaciju razgranaj ograniči. Mapa *almostFullTrees* sadrži upravo ono što joj i samo ime govori, za svaku informativnu poziciju sadrži sva stabla kojima nedostaje točno jedna sekvenca do punog stabla.

U drugom dijelu algoritma ide se po svim topologijama stabala koja sadrže sve sekvence osim zadnje. Za prvu topologiju dodajemo i zadnju sekvencu i izgrađuju se prva potpuna stabla. Sada se potpuna stabla po svakoj poziciji spajaju u puna stabla u metodi *buildFullTrees*. Od upravo izgrađenih potpunih i punih stabala, koja se spremaju u listu *fullTrees*, određuje se ono stablo koje ima minimalnu cijenu u metodi *getBoundary*. Metoda vraća minimalnu cijenu koja postaje prva gornja granica koja će se dalje koristiti. Nakon što je prva granica određena, za svaku sljedeću topologiju koja sadrži sve sekvence, osim zadnje, provjerava se ukupna cijena stabla. Ukoliko je cijena stabla veća od trenutne gornje granice taj put izgradnje potpunih stabala se odbacuje. Ukoliko je cijena manja ili jednaka iz te topologije izgrađuju se potpuna stabla za svaku poziciju i spajaju u puna stabla koja se dodaju u listu *fullTrees*. Novo dodana stabla se pretražuju i određuje se ono s najmanjom cijenom. Ukoliko je ta cijena manja od trenutne gornje granice gornja granica se osvježava.

Nakon završetka algoritma u varijabli *minimumCost* pohranjena je minimalna cijena optimalne topologije. Prolaskom kroz listu *fullTrees* možemo izdvojiti sva stabla koja su optimalna, odnosno sva stabla kojima je cijena jednaka vrijednosti varijable *minimumCost*.

Memorijska složenost implementacije u najgorem slučaju jednaka je broju svih mogućih stabala za sve informativne pozicije. Najgori slučaj je onaj u kojemu optimizacija razgranaj-ograniči ne uspije odbaciti niti jednu topologiju već se moraju izgraditi sva moguća stabla

Uzmimo da n označava broj informativnih pozicija, a T broj sekvenci. $M(t)$ označava memorijsku složenost.

Vrijedi sljedeća formula :

$$M(t) = n * T! \tag{5.1}$$

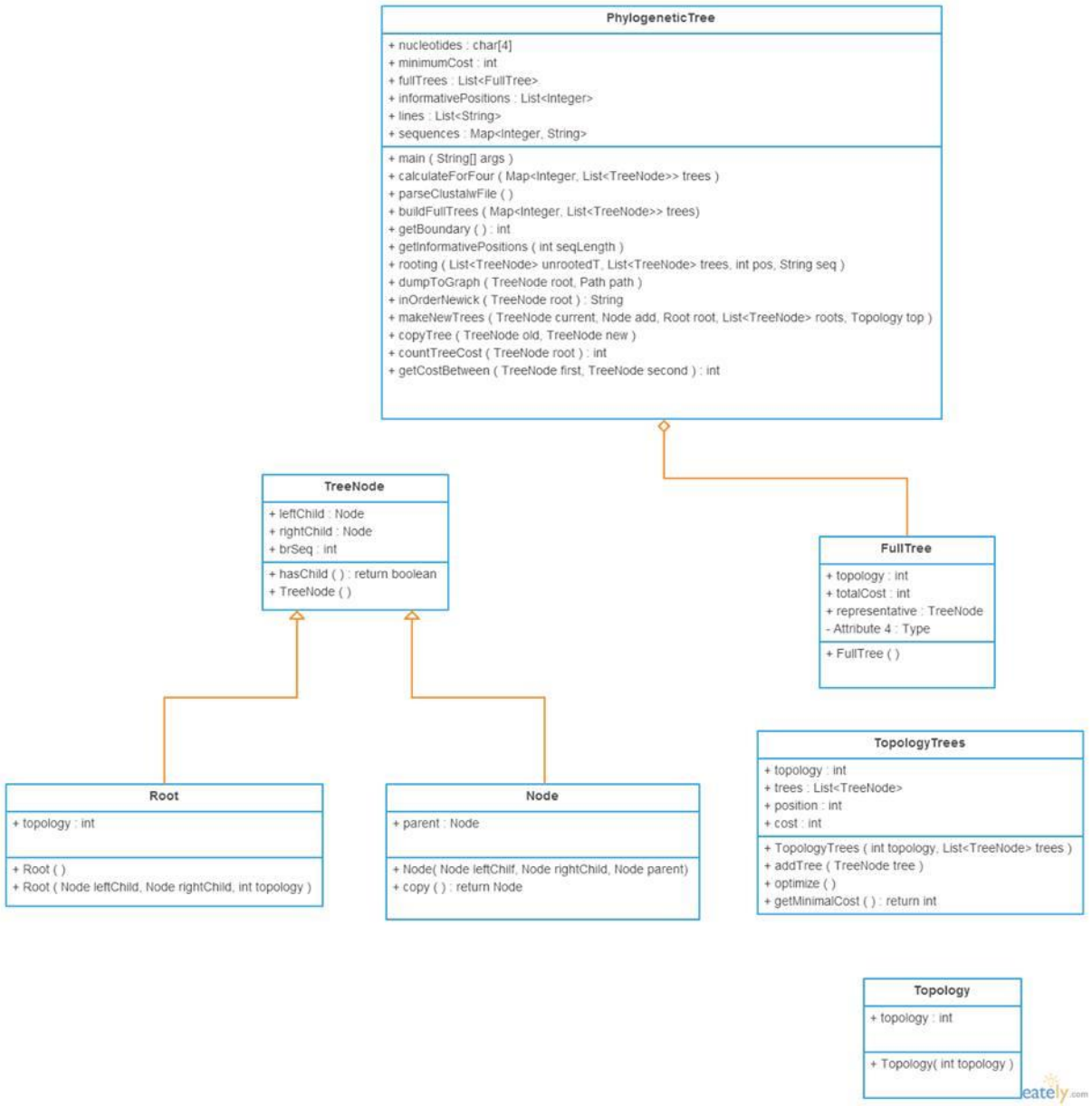
Također, **vremensku složenost** algoritma određujemo u najgorem slučaju. Uzmimo da $V(T)$ predstavlja vremensku složenost, T broj sekvenci, a n broj informativnih pozicija.

Vrijedi sljedeća formula :

$$V(T) = n * T^2 * T! \tag{5.2}$$

5.2 Strukture podataka programskog rješenja

U nastavku je priložen dijagram razreda koji detaljno opisuje glavne strukture podataka.



Slika 5. 1 Dijagram razreda

Razred *PhylogeneticTree* temeljni je razred algoritma. Kao jedna od glavnih struktura podataka ovog razreda izdvaja se lista *fullTrees* koja u sebi sadrži potpuna stabla koja su nastala tijekom dijela algoritma koji implementira razgranaj-ograniči optimizaciju. Druga glavna struktura podataka jest atribut *minimumCost* koji predstavlja cijenu optimalnog filogenetskog stabla koje je izgrađeno za zadani ulazni skup podataka.
Algoritam započinje u metodi *main*. Učitava se i parsira zadana datoteka i sekvence se spremaju u mapu *sequences*.
Neke od važnijih metoda ovog razreda su *makeNewTrees, buildFullTrees, getBoundary i countTreeCost.*
Metoda *makeNewTrees* stvara iz zadanog stabla sve moguće nove topologije sa svim kombinacijama unutrašnjih čvorova. Nova stabla sprema u listu *roots* koju prima kao ulazni argument.. Metoda *buildFullTrees* kao ulazni argument prima mapu koja kao ključ sadrži pozicije, a kao vrijednost listu svih stabala za tu poziciju. Metoda grupira sva stabla iste topologije, određuje ona optimalna za određenu topologiju i spaja sve pozicije u potpuno stablo. U ovoj metodi svakom stablu određuje se i ukupna cijena. Metoda *getBoundary* određuje minimalnu cijenu između cijena svih stabala koja se u tom trenutku nalaze u listi *fullTrees*.
Metoda *countTreeCost* računa cijenu zadanog stabla.

Razred *TreeNode* apstraktan je razred i služi kako bi objedinio razrede *Root* i *Node*. Sastoji se od lijevog i desnog djeteta svakog čvora (kod listova te vrijednosti su null) i metode *hasChild* koja određuje je li čvor list.

Razred *Root* predstavlja fiktivni korijen stabla. On nema nikakvo semantičko značenje i korišten je samo kako bi obilazak kroz stablo bio lakši. Kako se svako stablo može obići ako imamo korijen, u objekte tipa *Root* spremamo oznaku topologije cijelog stabla. Prilikom računanja cijene stabla korijen nema nikakav utjecaj. Direktna djeca korijena pokazuju jedno na drugo i uzima se cijena između njih umjesto cijene između korijena i njih.

Razred *Node* predstavlja čvor stabla, bilo unutrašnji, bilo listove. Ima atribut tipa *Node* koja predstavlja roditelja tog čvora, a kako nasljeđuje klasu *TreeNode* sadrži

i atribute lijevo i desno dijete. Sadrži metodu *copy* koja samo stvara novu instancu čvora sa jednakim podacima.

Razred *FullTree* predstavlja potpuno stablo. Sadrži oznaku topologije, ukupnu cijenu i atribut *representative*. Atribut *representative* predstavlja stablo pripadne topologije i minimalne cijene za neku od pozicija. Njegova uloga je prikazivanje stabla. Naime, svaki objekt tipa *TreeNode* ima atribut oznake broja sekvence preko kojeg dolazimo do imena sekvence kojoj pripada i tako možemo iscrtati stablo s imenima sekvenci. Kako potpuno stablo za sve pozicije mora imati istu topologiju dovoljna je jedna pozicija kao predstavnik cijelog stabla.

Razred *TopologyTrees* predstavlja sva stabla topologije određene atributom *topology* i pozicije određene atributom *position.* Također sadrži minimalnu cijenu stabla takve topologije za zadanu poziciju. Metoda *optimize* iz liste svih stabala izbacuje ona koja nisu optimalna.

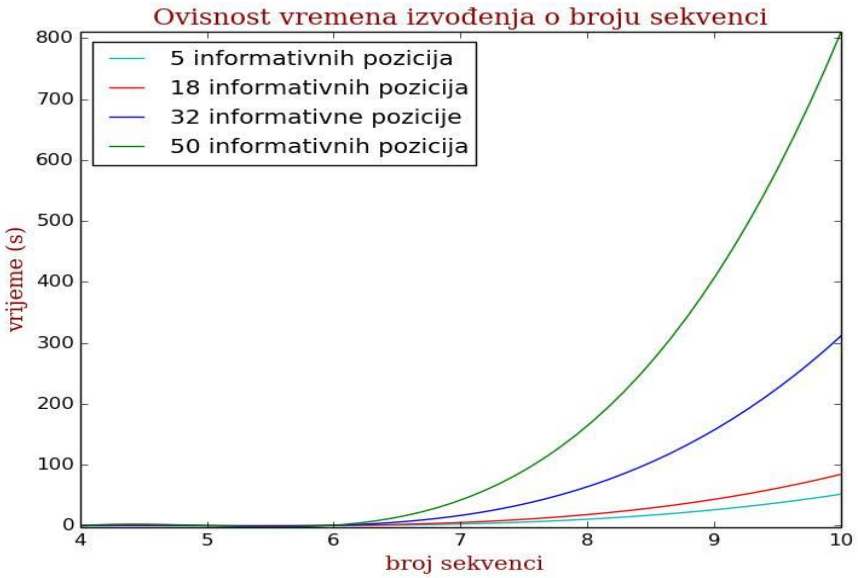
Razred *Topology* predstavlja topologiju koja je predstavljena brojem.

5.3 Vremenski i memorijski benchmark

Vremenski i memorijski benchmark ispitani su na računalu *Dell Inspiron N5110, i3 intel pentium* procesor.
Određeni su pokretanjem programa na različitim ulaznim podacima koji su sadržavali od 4 do 7 sekvenci. S obzirom na dobivene podatke, ovisno o tome što se računalo, graf je aproksimiran koristeći alat *Matplotlib* i metodu *polyfit* koja aproksimira krivulju s obzirom na zadane točke.
Kako veličina prostora stanja koje treba pretražiti, da bi se za ulazni skup sekvenci odredilo optimalno filogenetsko stablo, ne ovisi samo o broju sekvenci već i o broju informativnih pozicija za zadani skup sekvenci tako su i grafovi koji prikazuju zavisnost vremena/memorije o broju sekvenci prikazani za različitu količinu informativnih pozicija. Ovisnosti su mjerene za 5, 18, 32 i 50 informativnih pozicija.

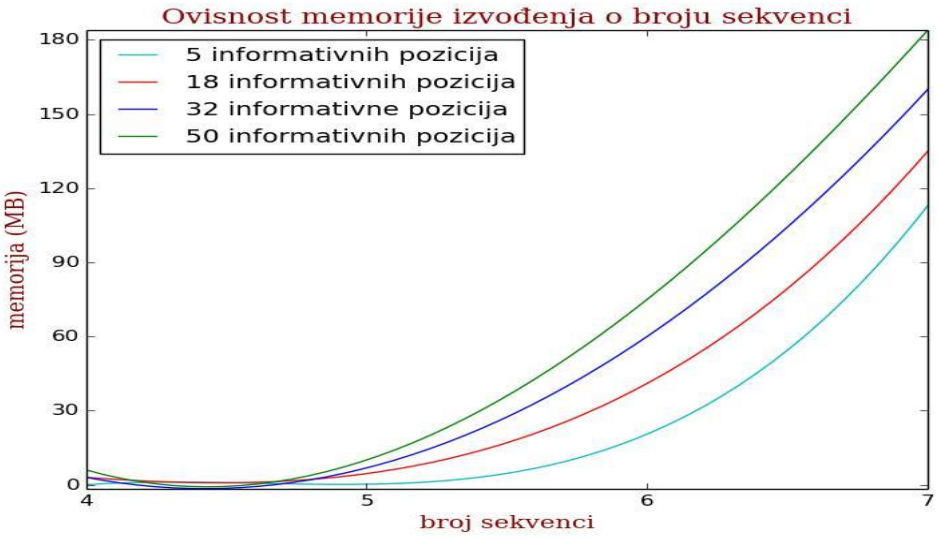
Vrijeme, kao i memorija, i broj informativnih pozicija u sekvencama su proporcionalni, porastom broja informativnih pozicija raste i vrijeme odnosno memorija izračuna.

Slika 5.2 prikazuje ovisnost vremena izvođenja programa o broju sekvenci za koje se računa filogenetsko stablo. Iz slike se jasno vidi da je ta ovisnost, neovisno o broju informativnih pozicija, eksponencijalna.



Slika 5.2 Ovisnost vremena izvođenja o broju sekvenci

Slika 5.3 prikazuje ovisnost zauzete memorije prilikom izvođenja programa o broju sekvenci za koje se računa filogenetsko stablo. Ovisnost je, kao i kod vremena, eksponencijalna.



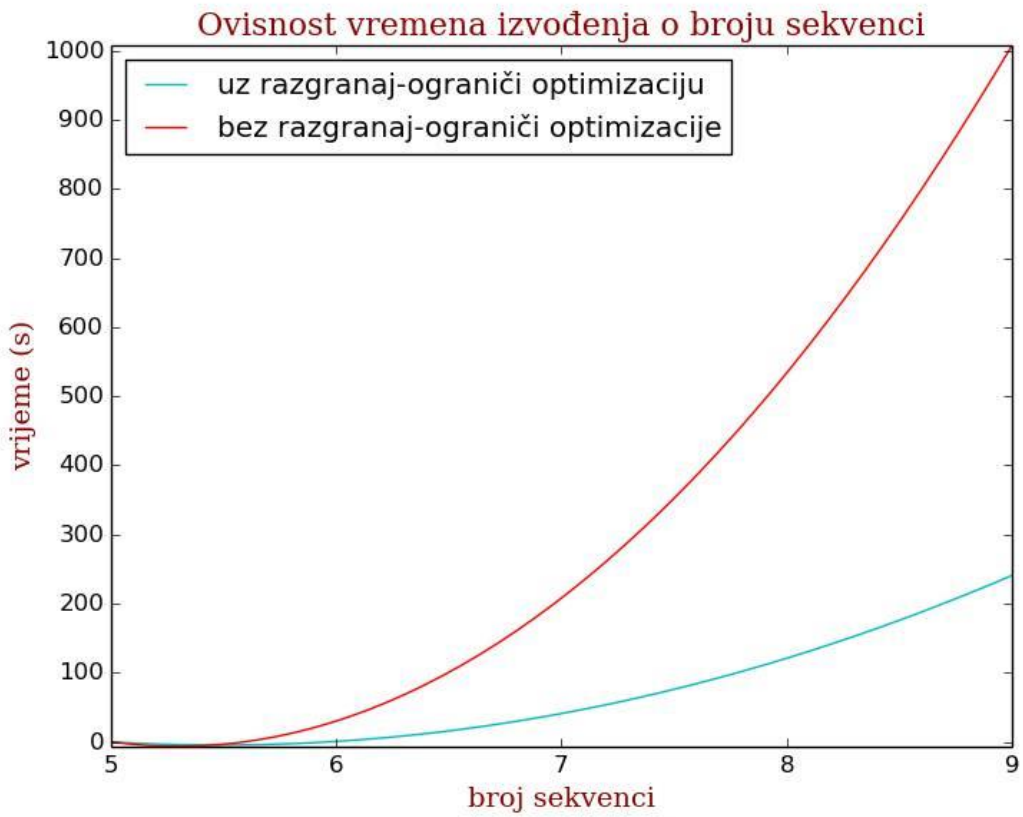
Slika 5.3 Ovisnost količine zauzete memorije o broju sekvenci

Vrijednosti prikazane na grafovima dobivene su uz korištenje optimizacijskog algoritma razgranaj-ograniči koji smanjuje prostor stanja.

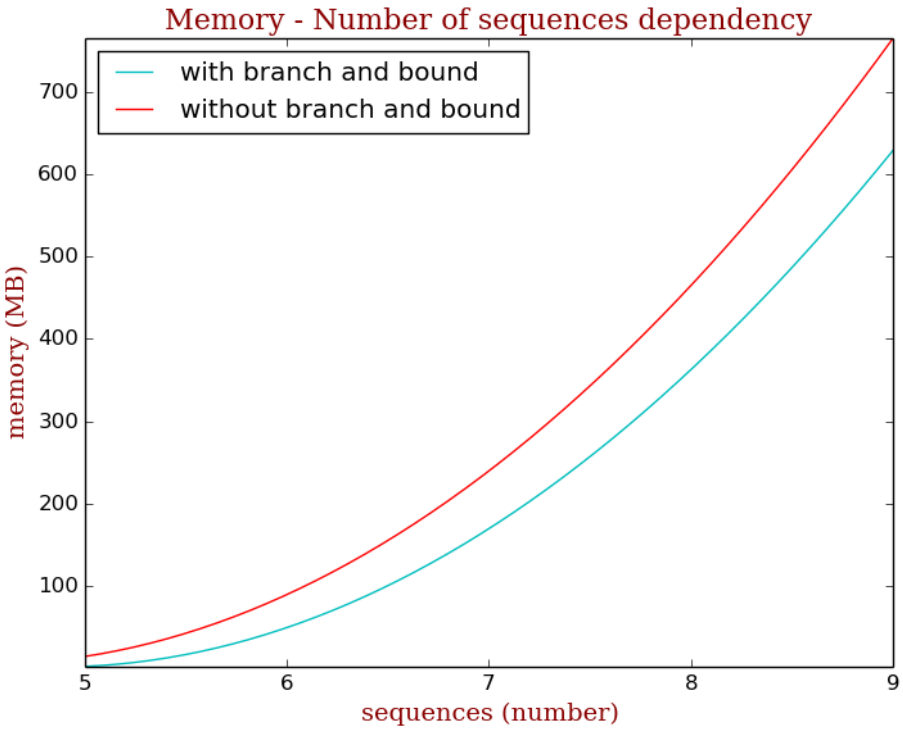
Na slikama 5.4 i 5.5 prikazane su ovisnosti vremena/memorije o broju sekvenci uz korištenje optimizacijskog algoritma i bez korištenja.

Graf je izgrađen na temelju vrijednosti koje su dobivene testiranjem programskog rješenja na podacima od 5, 6 i 7 sekvenci, dok je broj informativnih pozicija u svim primjerima bio 50.

Iz grafova vidimo da algoritam razgranaj-ograniči doista smanjuje broj stanja i samim time potrebno vrijeme i memoriju izračuna.



Slika 5.4 Ovisnost vremena o broju sekvenci



Slika 5.5 Ovisnost količine zauzete memorije o broju sekvenci

5.4 Usporedba s Phylip alatom

Tablica 5.1 Usporedba s Phylip alatom

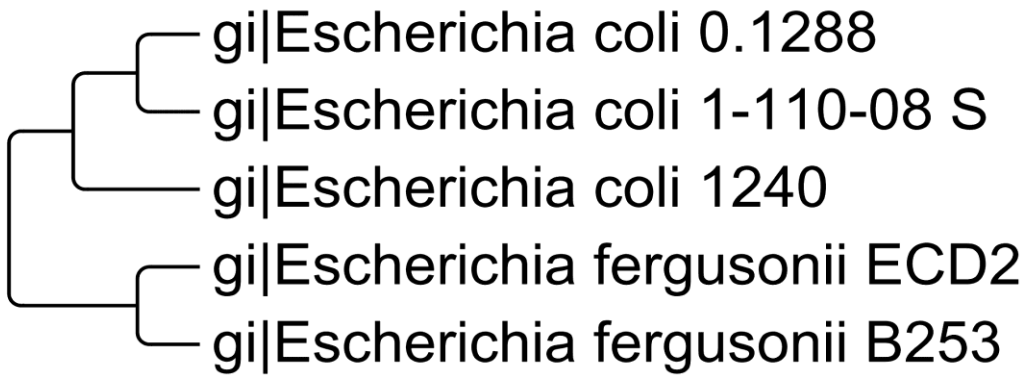
	Vrijeme (s)	Broj stabala	Cijena
Programsko rješenje	3.28	45	15
Phylip alat	1.5	45	21

Navedena tablica prikazuje usporedbu moje implementacije algoritma maksimalne uštede uz korištenje optimizacije razgranaj-ograniči i implementacije alata Phylip koji koristi istu metodu i istu optimizaciju. Usporedba je provedena na skupu podataka od 7 sekvenci.

I jedna i druga implementacija daju jednake optimalne topologije, ali cijena se razlikuje. To se dogodilo zbog različitog kriterija za određivanje informativnih pozicija. Vremenske i memorijske performanse alata Phylip nešto su bolje.

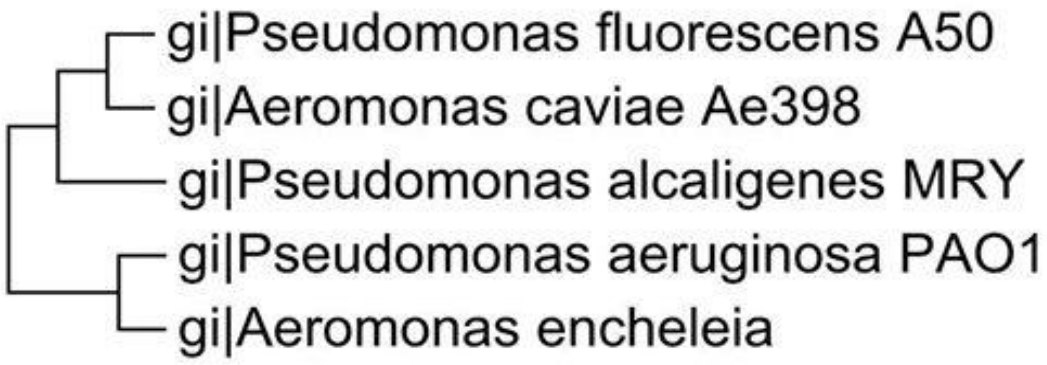
5.5 Primjer pokretanja na biološkim podacima

Na slici 5.6 prikazano je filogenetsko stablo dobiveno za zadane podatke metodom maksimalne uštede implementiranom u ovom radu. Isto stablo dobiveno je i Phylip alatom i predstavlja jedino optimalno stablo za zadane sekvence s obzirom na kriterij maksimalne uštede. Stablo je izgrađeno za rod Escherichia, vrste Escherichia Coli i Escherichia Fergusonii. Ovo je primjer sekvenci za koje metoda maksimalne uštede proizvodi biološki ispravno filogenetsko stablo. Zajedničkim čvorovima spojene su jednake vrste roda Escherichia koje doista i jesu evolucijski najbliže.



Slika 5.6 Filogenetsko stablo za vrste roda Escherichia

Na slici 5.7 prikazano je filogenetsko stablo dobiveno za rod Pseudomonas i Aeromonas i za njihove vrste Pseudomonas Fluorescens, Pseudomonas Alcaligenes, Pseudomonas Aeruginosa, Aeromonas Encheleia i Aeromonas Caviae.



Slika 5.7 Filogenetsko stablo za rodove Pseudomonas i Aeromonas

Isto stablo dobiveno je Phylip alatom i predstavlja jedino optimalno stablo s obzirom na kriterij maksimalne uštede. Ovo je primjer stabla koje nije biološki ispravno. Iz slike vidimo da su u zajedničke čvorove spojene vrste rodova Pseudomonas i Aeromonas. Očekivano filogenetsko stablo temeljeno na sistematici spojilo bi zajedno vrste roda Pseudomonas u jedan čvor i onda bi taj čvor bio spojen sa čvorom koji spaja sve vrste roda Aeromonas. Ovakvo stablo je optimalno za kriterij maksimalne uštede, ali pokazuje da iako se neke vrste manje razlikuju ne znači nužno da su evolucijski bliže.

Zaključak

Filogenija je grana znanosti koja proučava evolucijske odnose među vrstama koji se obično prikazuju filogenetskim stablom. Za rekonstrukciju filogenetskog stabla razvijen je čitav niz metoda, a u ovom radu opisana je i implementirana jedna od metoda koje se temelje na matrici obilježja i na optimalnom kriteriju, metoda maksimalne uštede.

Osnovni princip metode maksimalne uštede jest ideja da je ispravno ono stablo koje najjednostavnije objašnjava zadani skup podataka, odnosno ono stablo koje održava minimalan broj evolucijskih promjena. S obzirom da je riječ o NP-teškom problemu prilikom implementacije programskog rješenja korištena je optimizacija razgranaj-ograniči. Vremenska i memorijska složenost rješenja su faktorijelne jer ova metoda traži optimalno rješenje. Uz korištenje spomenute razgranaj-ograniči optimizacije broj stabala, koja se moraju izgraditi kako bi se odredilo optimalno stablo, znatno se smanjuje i omogućava korištenje rješenja na većem skupu podataka, 8-12 sekvenci. Broj sekvenci za koje rješenje daje izlaz u realnom vremenu također ovisi o sličnosti među sekvencama. Što je broj evolucijskih promjena među sekvencama manji, broj mogućih stabala pada pa se shodno tome rješenje može koristiti na većem skupu podataka.

Programsko rješenje za zadani skup sekvenci generira sva optimalna filogenetska stabla. Testirano je na različitim podacima koji su sadržavali višestruko poravnate sljedove dobivene sekvenciranjem različitih vrsti. Testiranjem je pokazano da implementacija programskog rješenja metode daje očekivane rezultate koji su matematički precizni s obzirom na zadani optimalni kriterij. Unatoč činjenici da je za neke sekvence koje spadaju u već spomenutu Felsensteionu zonu, dobiveno optimalno stablo evolucijski neispravno, metoda maksimalne uštede pokazala se kao metoda koja za većinu podataka daje biološki ispravna i optimalna filogenetska stabla.

Literatura

[1] Fitch, W.M., Margoliash, E. (1967) Construction of phylogenetic trees

[2] Mount, DM. (2004) Bioinformatics: Sequence and Genome Analysis

[3] Swofford, D.L., Sullivan J. (2001) Phylogeny inference based on parsimony and other methods using PAUP

[4] Day W.H.E. (1987) Computational complexity of inferring phylogenies from dissimilarity matrices: Bulletin of Mathematical Biology

[5] Siddall M.E., Whiting M.F. (1999) Long-Branch Abstractions

[6] Felsenstein J. (1987) Cases in which parsimony and compatibility methods will be positively misleading

[7] Fitch, W.M. (1971) Toward defining the course of evolution: minimum change for a specified tree

[8] Hall,B.G. (2004) Comparison of the Accuracies of Several Phylogenetic Methods Using Protein and DNA Sequences

[9] Jaynes, E.T. (2003) Probability theory: The logic of science. Cambridge. Cambridge University Press

Rekonstrukcija filogenetskog stabla metodom maksimalne uštede uz razgranaj-ograniči optimizaciju

Sažetak

Metoda maksimalne uštede, kao jedna od metoda za rekonstrukciju filogenetskog stabla, po principu Occamove oštrice zahtjeva da rekonstruirano stablo koristi minimalan broj mutacija potreban za objašnjenje podataka. Metoda maksimalne uštede spada u metode koje se temelje na matrici obilježja i na optimalnom kriteriju. Kako je ovaj problem NP-težak, prilikom implementacije metode korištena je razgranaj-ograniči optimizacija.

Programsko rješenje testirano je na nizu različitih sekvenci i dobiveni su rezultati u skladu s očekivanima.

Ključne riječi: metoda maksimalne uštede, matrica obilježja, razgranaj-ograniči

Phylogenetic tree reconstruction using maximum parsimony branch-and-bound algorithm

Abstract

Maximum parsimony is a character-based method, akin to Occam's razor principle, that infers a phylogenetic tree by minimizing the total number of evolutionary steps required to explain a given set of data. It relies on the use of optimality criteria. Due to the fact that this problem is an NP-hard problem, branch and bound optimization was used in software solution.

Software solution was tested on various sets of sequences and the results were as expected.

Keywords: maximum parsimony, character-based, branch and bound