Univerzitet u Beogradu
Matematički fakultet

Master Rad

Razvoj aplikacije za vizualizaciju algoritama za rekonstruisanje filogenetskih stabala

Autor:
Isidora Đurđević

Mentor:
doc. dr Jovana Kovačević

Članovi komsije:

doc. dr Jovana Kovačević
doc. dr Nina Radojičić Matić
Aleksandar Veljković

Beograd, 2021.

Sažetak

Teorija evolucije nam govori da je život na Zemlji nastao od zajedničkog pretka - vrste koja je menjala svoj DNK, prilagođavajući se sredini u kojoj se nalazila. Izmene u genetskom materijalu podrazumevaju razne mutacije i rekombinacije unutar DNK i dovode do stvaranja različitih vrsta. Uporednom analizom DNK sekvenci različitih organizama moguće je prikazati odnose između vrsta (ili drugih taksonomskih kategorija) u vidu stabla i tako uočiti kako su i kojom brzinom taksoni evoluirali u odnosu na zajedničkog pretka. Takvo stablo nazivamo filogenetskim stablom. Svaka grana u stablu predstavlja evolutivnu vezu, a njena dužina ukazuje na količinu proteklog vremena ili broj mutacija do trenutka razdvajanja taksona od svog pretka. Tačka grananja unutar stabla predstavlja zajedničkog pretka, dok su na krajevima grane oznake potomaka. Specifični cilj ovog rada je razvoj veb aplikacije koja vizualizuje algoritme za rekonstrukciju filogenetskih stabala. U aplikaciji će biti ilustrovana primena algoritama zasnovanih na matricama rastojanja i algoritama zasnovanih na karakteristikama organizama. Prikaz pojedinačnih koraka u rekonstrukciji stabla će olakšati razumevanje ovih algoritama, pa se aplikacija može koristiti kao pomoćno nastavno sredstvo.

Sadržaj

Sažetak **iii**

1 Uvod **1**

2 Biološke osnove **3**
2.1 Stablo . . . 3
2.1.1 Rastojanje između čvorova . . . 4
2.1.2 Matrica rastojanja . . . 4
2.1.3 Aditivne matrice . . . 5
2.1.4 Ultrametrične matrice . . . 6

3 Algoritmi za rekonstrkuciju filogenetskih stabala **9**
3.1 Aditivna filogenija . . . 9
Primer primene algoritma aditivne filogenije . . . 10
3.2 UPGMA . . . 14
Primer primene UPGMA algoritma . . . 14
3.3 Neighbor-joining algoritam . . . 17
Primer primene *NJ* algoritma . . . 20
3.4 Algoritam male parsimonije . . . 24
Primer primene algoritma male parsimonije . . . 26

4 Aplikacija PhyTReeV **33**
4.1 Klijent . . . 33
4.1.1 Implementacija klijenta . . . 36
4.2 Server . . . 39
4.2.1 API - Programski interfejs aplikacije . . . 39
4.2.2 Implementacija servera . . . 41
4.2.3 Korišćenje aplikacije . . . 43
4.2.4 Primer upotrebe . . . 43
Prvi primer upotrebe - učitavanje podataka iz datoteke . . . 43
Drugi primer upotrebe - unošenje podataka u tekstualno polje . 44
Treći primer upotrebe - unošenje višestrukog poravnanja u tekstualno polje . . . 46

5 Zaključak **49**

Bibliografija **51**

Spisak slika

1.2 Primer korenog i nekorenog stabla 2

2.1 Matrica rastojanja i odgovarajuće filogenetsko stablo 5
2.2 Prikaz (levo) aditivne i (desno) neaditivne matrice [6] 6
2.3 Prikaz ultrametričnog stabla nekoliko vrsta životinja [3] 6
2.4 Matrica rastojanja i odgovarajuće ultrametrično stablo 7

3.1 Primer matrice u kojoj minimalna vrednost ne odgovara susednim listovima 10
3.2 Prvi korak rekonstrukcije stabla 12
3.3 Drugi korak rekonstrukcije stabla 13
3.4 Treći korak rekonstrukcije stabla 13
3.5 Četvrti korak rekonstrukcije stabla 13
3.6 Poslednji korak rekonstrukcije stabla 14
3.7 Filogenetsko stablo dobijeno korišćenjem UPGMA algoritma 17
3.8 Prikaz rekonstrukcije filogenetskog stabla na osnovu rastojanja [6] . . . 19
3.9 Prvi korak rekonstrukcije *NJ* stabla 23
3.10 Drugi korak rekonstrukcije *NJ* stabla 23
3.11 Treći korak rekonstrukcije *NJ* stabla 23
3.12 Poslednji korak rekonstrukcije *NJ* stabla 24
3.13 Primer matrice karakteristika za utvrđivanje filogeneze kod različitih vrsta životinja 25
3.14 Prikaz topologije stabla za gore navedenu matricu karakteristika [6] . 27
3.15 Prikaz (plavog) podstabla T_v čvora v u okviru stabla T [6] 28
3.16 Inicijalizacija $s_k(v)$ za sve listove v [6] 28
3.17 Inicijalizacija $s_k(v)$ unutrašnjih čvorova na nivou 2 [6] 30
3.18 Inicijalizacija $s_k(v)$ unutrašnjih čvorova na nivou 3 [6] 30
3.19 Inicijalizacija korenog čvora [6] 31

4.1 Prikaz korisničkog interfejsa 34
4.2 Prikaz *UPGMA* algoritma 35
4.3 Prikaz *NJ* algoritma 35
4.4 Prikaz *AdditivePhylogeny* algoritma 36
4.5 Prikaz *SmallParsimony* algoritma 36
4.6 Prikaz čvorova u polarnom koordinatnom sistemu 37
4.7 Prikaz čvorova u Dekartovom koordinatnom sistemu sa horizontalnom orijentacijom 37
4.8 Prikaz vertikalne orijentacije stabla 37
4.9 Prikaz odabrane datoteke 44
4.10 Prikaz učitane datoteke 44
4.11 Prikaz korisničkog interfejsa nakon izvršavanja *UPGMA* algoritma . . 45
4.12 Prikaz početnog stabla 45

4.13 Prikaz korisničkog interfejsa nakon izvršavanja *Small Parsimony* algoritma 46
4.14 Prikaz unetog višestrugog poravnanja u tekstualno polje 47
4.15 Prikaz korisničkog interfejsa nakon izvršavanja *Neighbor-Joining* algoritma 47

Spisak tabela

4.1 Prikaz API deklaracije 40
4.2 Prikaz parametara zahteva 41

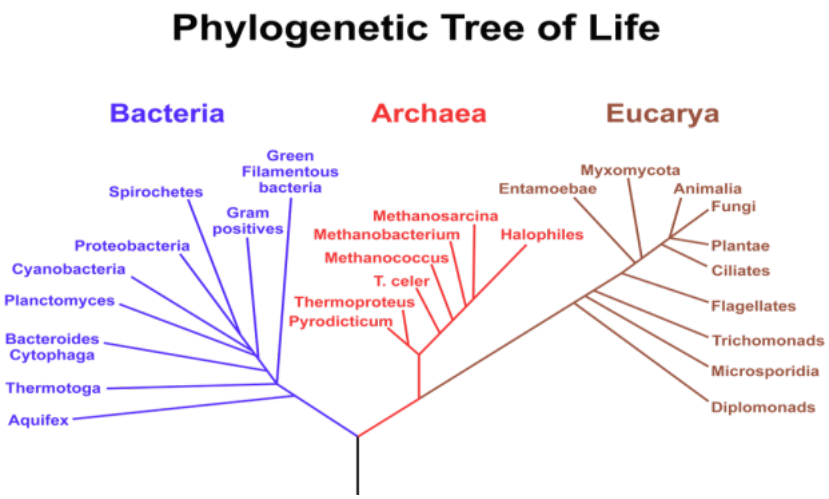
Mami i tati...

Glava 1

Uvod

SARS se prvi put pojavio 2003. godine i ostao zapamćen kao virus koji se za kratak vremenski period proširio po celom svetu. Naime, virusu je bilo potrebno svega sedam dana da pređe preko Tihog okeana do obale Severne Amerike polazeći iz Hong Konga. Poređenja radi, bolesti kao što je kuga je bilo potrebno četiri godine kako bi prešla iz Istanbula do Kijeva, dok je HIV-u bilo potrebno dve decenije da obiđe ceo svet. Kada je utvrđeno da je uzročnik nove bolesti virus, posmatrajući ga pod mikroskopom uočeno je da pripada porodici korona virusa. Korona virus je već bio poznat naučnicima i lekarima, ali se do tog trenutka širio i prenosio samo u životinjskom svetu, odnosno nije bilo zabeleženo da je u nekom delu sveta čovek zaražen ovim tipom virusa.

Do danas su istraživači sekvencirali dosta sojeva korona virusa izolovanih iz pacijenata iz različitih zemalja, ali i dalje je ostalo pitanje kako je korona virus prešao sa životinje na čoveka? Kako se virus širio i sa koijh se osoba na koje prenosio? Ovi problemi spadaju u probleme filogenetske analize i rešavaju se rekonstrukcijom takozvanih filogenetskih stabala. Filogenetska ili evolutivna stabla predstavljaju evolutivnu vezu između taksona kroz vreme.



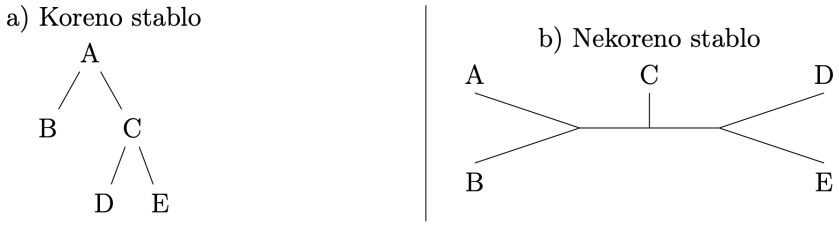
Slika 1.1: Prikaz filogenetskog stabla živog sveta na Zemlji[1]

Vrste u grafu su predstavljene čvorovima, a veze između njih granama grafa. Listovi takvog stabla predstavljaju današnje vrste, dok unutrašnji čvorovi predstavljaju

[1]Preuzeto sa https://commons.wikimedia.org/wiki/File:Phylogenetic_tree_scientific_names.svg

izumrle vrste. Svaki čvor ovakvog stabla pripada odgovarajućoj taksonomskoj kategoriji, s tim da unutrašnji čvorovi odgovaraju hipotetičkoj taksonomskoj kategoriji ukoliko vrsta ne postoji, nikad nije ni postojala ili je izumrla. Koren filogenetskog stabla predstavlja najstarijeg pretka, prvu živu vrstu od koje su potekle ostale vrste kao rezultat mutacija i kombinacija genetskog materijala. Između listova i korena nalaze se čvorovi, tako da prolaskom od korena do listova možemo ispratiti kako su se organizmi kroz vreme razvijali i menjali. Na slici 1.1 je prikazano evolutivno stablo živog sveta na Zemlji [1].

Treba naglasiti da filogenetsko stablo ne mora da ima koren u slučaju kada se ne zahteva poznavanje evolutivnog poretka taksona predstavljenih unutrašnjim čvorovima stabla. U tom slučaju bi povezanost više čvorova sa jednim zajedničkim čvorom označavalo njihovu sličnost, a ne nužno evolutivni poredak. S druge strane, postojanje korena u filogenetskom stablu ukazuje na smer evolutivnog procesa kao i odnos između pretka i potomka kroz vreme. Primer korenog i nekorenog stabla je prikazan na slici 1.2.



Slika 1.2: Primer korenog i nekorenog stabla

Na osnovu genetskih sekvenci mogu se rekonstruisati filogenetska stabla, a na osnovu njegove analize može se odgovoriti na pitanje kako je došlo do prelaska virusa sa životinje na čoveka.

U prvom poglavlju date su biološke osnove neophodne za razumevanje motivacije i cilja istraživanja. Najpre su opisana stabla, njihova struktura i kako se koriste u slučaju filogeneze. Potom se govori o matricama rastojanja, njihovoj strukturi i značaju. Na samom kraju će biti predstavljena određena svojstva nekih matrica rastojanja koja umnogome mogu da olakšaju proces rekonstrukcije filogenetskog stabla.

U drugom poglavlju govori se o različitim metodama filogenetske analize koje uključuju algoritme za rekonstrukciju filogenetskih stabala zasnovane na rastojanju (*UPGMA*, *Neighbour-Joining* i aditivna filogenija) i algoritme zasnovane na karakteristikama organizama (mala parsimonija).

U trećem poglavlju se govori o implementaciji same aplikacije, korišćenim tehnologijama i načinima upotrebe. Poslednje poglavlje opisuje postignute rezultate i moguća unapređenja.

Glava 2

Biološke osnove

U ovoj sekciji biće predstavljena filogenetska stabla i pojmovi neophodni za njihovo uvođenje. Poseban akcenat se stavlja na matrice rastojanja, njihov značaj, kao i na bitna svojstva koja one mogu imati.

2.1 Stablo

Stablo je povezani aciklički graf za koji se može pokazati da važi:

- *Svako stablo sa bar dva čvora sadrži bar dva lista*
- *Svako stablo sa n čvorova sadrži tačno n - 1 grana*
- *Za proizvoljna dva čvora postoji tačno jedna putanja koja ih povezuje*

Kod filogenetskih stabala, čvorovi će predstavljati različite vrste. Vrste su predstavljenje svojim genomskim sekvencama ili karakteristikama. Imajući u vidu da se pokazalo da je teško konstruisati višestruko poravnanje za cele genome, o čemu će biti više reči u nastavku, najčešće se uzima deo genoma koji je specifičan za datu vrstu. U slučaju korona virusa, da bi naučnici razumeli kako je virus prešao sa životinje na čoveka bilo je potrebno sekvencirati genome korona virusa koji napadaju različite vrste. Sekvenciranjem ovih genoma i njihovim međusobnim poređenjem mogli bi se pronaći izolati virusa koji napadaju određenu životinjsku vrstu, a koji su na genetskom nivou najsličniji izolatima pronađenih kod čoveka. Time bi se osnovano mogla doneti pretpostavka da je virus prenet na čoveka sa baš te životinjske vrste.

S obzirom na to da kod viralnih genoma često dolazi do mutacija, odnosno insercija i delecija, od ukupno šest gena virusa SARS za filogenetsku analizu korišćen je gen koji najmanje podleže mutacijama. Ovaj gen kodira takozvani *Spike* protein koji je zadužen za vezivanje virusa za ćelije domaćina.

Grane u filogenetskom stablu predstavljaju kvantitativnu vrednost razlike između čvorova koje povezuju. Te razlike se mogu ogledati u broju izmena nad biološkim sekvencama ili nekim drugim karakteristikama organizama što je obično srazmerno vremenu proteklom od nastanka vrste na jednom kraju do nastanka vrste na drugom kraju grane stabla. Ova vrednost predstavlja evolutivnu udaljenost između vrsta na krajevima grane.

U evolutivnom stablu, za čvorve koji se nalaze na većoj međusobnoj udaljenosti kažemo da su u *daljem* srodstvu, a za čvorove na manjoj razdaljini da su u bližem srodstvu. Na taj način, koren stabla će biti označen početnom vrstom, unutrašnji čvorovi hipotetičkim vrstama, a listovi današnjim vrstama.

2.1.1 Rastojanje između čvorova

Svako filogenetsko stablo τ definiše skup vrednosti koje predstavljaju rastojanja $d^{\tau}_{i,j}$ između čvorova i, $j \in X = \{1, 2, \ldots, n\}$. Ako $b(e)$ predstavlja dužinu grane e, onda se rastojanje između čvorova definiše kao:

$$d^{\tau}_{i,j} = \sum_{e \in \tau_{i,j}} b(e)$$

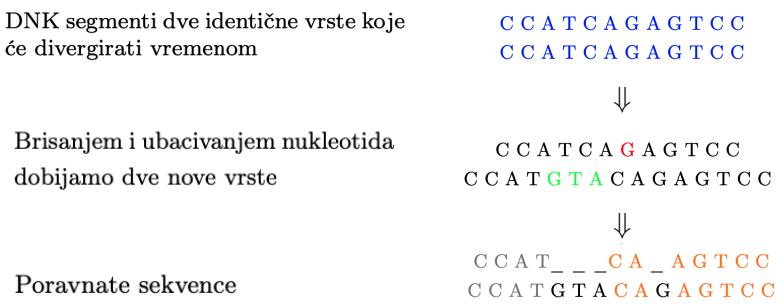
gde $\tau_{i,j}$ predstavlja skup grana na putanji od čvora i do čvora j u stablu τ.

U nastavku će biti uveden pojam matrice rastojanja koje predstavljaju osnovu za izgradnju jedne grupe algoritama za rekonstrukciju filogenetskih stabala.

2.1.2 Matrica rastojanja

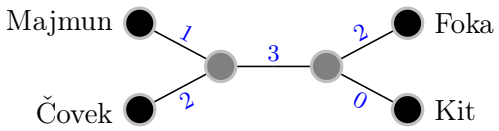
Matrica rastojanja je simetrična kvadratna matrica čije vrednosti predstavljaju rastojanja između čvorova. Ako svaku vrstu predstavljamo genomskom sekvencom, matrica rastojanja će predstavljati vrednost funkcije rastojanja između dve genomske sekvence. Da bismo dobili rastojanja između dva organizma potrebno je izvršiti poravnanje njihovih sekvenci i ustanoviti koliko se mutacija desilo između dve vrste. Ukoliko su u pitanju srodne vrste, broj mutacija u sekvencama će biti mali. Nasuprot tome, poređenjem genomskih sekvenci udaljenijih vrsta dobili bismo veliki broj mutacija i potencijalno određeni broj dobro konzerviranih regiona koji su zajednički za obe vrste. Brojanjem različitih simbola u poravnatim sekvencama dobijamo rastojanje u matrici. Ukoliko se isti simbol nalazi na istoj poziciji u sekvencama onda se to smatra poklapanjem (engl. *match*), inače se nedostatak simbola i različiti simboli na istoj poziciji tretiraju kao nepoklapanje (engl. *missmatch*).

Pretpostavimo da imamo dve sekvence čiji je nekodirajući deo DNK jako sličan. Ukoliko bi bio obrisan prvi deo jedne sekvence, kao rezultat bi ta sekvenca bila pomerena u levo za onoliko nukleotida koliko ih je obrisano. Ukoliko se brisanje ne uzme u obzir, kao i pomeranje sekvence, onda bi pri naivnom poređenju nukleotida na istim pozicijama dve naizgled jako slične sekvence mogli da dobijemo neispravne rezultate. Iz tog razloga, za potrebe poređenja bioloških sekvenci nastale su specifične metode za računanje poravnanja. Za poređenje više od dve genomske sekvence koriste se metode višestrukog poravnanja. Detalji ovih metoda neće biti detaljno obrađeni u ovom radu. U nastavku je prikazan primer poravnanja 2.1.2, kao i mutacija koje su se dešavale kroz vreme između dve sekvence. Kao rezultat u poslednjem koraku, broj različitih nukleotida na odgovarajućim pozicijama u sekvencama će predstavljati rastojanje između ove dve sekvence.



Primer 2.1.2: Prikaz poravnanja između dve sekvence.

	Majmun	*Covek*	*Foka*	*Kit*
Majmun	0	3	6	4
Covek	3	0	7	5
Foka	6	7	0	2
Kit	4	5	2	0



Slika 2.1: Matrica rastojanja i odgovarajuće filogenetsko stablo

Definicija 1 *Matricom rastojanja D nazivamo matricu čiji svaki element u oznaci $D_{i,j}$ predstavlja broj različitih simbola na istim pozicijama u i-toj i j-toj sekvenci višestrukog poravnanja.*

Matrica rastojanja će odgovarati filogenetskom stablu ukoliko su rastojanja između listova jednaka odgovarajućim rastojanjima u matrici rastojanja. Na slici 2.1 je prikazan primer matrice rastojanja i njenog odgovarajućeg filogenetskog stabla.

Određivanje matrice rastojanja na osnovu stabla je trivijalno, dok za obrnut problem to nije slučaj. Ukoliko matrica rastojanja ispunjava određene uslove o kojima će biti reči u potpoglavljima 2.1.4 i 2.1.3, onda se može generisati jedinstveno stablo, ali u realnim primenama to neće biti slučaj i neće biti moguće konstruisati stablo koje odgovara matrici. Iz tog razloga, pronalaženje stabla na osnovu matrice rastojanja se svrstava u NP klasu problema pa se uglavnom pribegava rešenjima koje se zasnivaju na heuristikama kako bi se smanjio broj međustabala koja mogu da se generišu tokom izvršavanja algoritma za rekonstrukciju finalnog stabla.

Na koji god način da se dobije stablo, bitno je naglasiti da se njegova struktura, to jest topologija ne menja - redosled čvorova, veze i rastojanja između njih ostaju ista.

Iz gore navedenog razloga osnovna svojstva koja mora da zadovoljava matrica da bi se na osnovu nje efikasno rekonstruisalo stablo jesu aditivnost ili ultrametričnost. U suprotnom je konstrukcija stabla vremenski i prostorno veoma zahtevna ili nemoguća.

2.1.3 Aditivne matrice

Kao što je već pomenuto, na osnovu jedne matrice rastojanja može se konstruisati više stabala različite topologije, zbog čega se uvodi pojam *prostog* stabla. Prosto stablo je stablo koje ne sadrži čvorove stepena dva. Problem rekonstrukcije filogenetskog stabla na osnovu matrice rastojanja svodi se na pronalaženje prostog filogenetskog stabla koje odgovara matrici rastojanja. U nastavku sledi definicija uslova koju matrica mora da ispunjava kako bi bilo moguće efikasno konstruisati filogenetsko stablo na osnovu nje. Na slici 2.2 je dat primer aditivne i neaditivne matrice.

Definicija 2 *Ukoliko za bilo koja četiri proizvoljna indeksa u matrici i, j, k i l važi $D_{i,j} + D_{k,l} <= D_{i,k} + D_{j,l} = D_{i,l} + D_{j,k}$ onda se ta matrica naziva aditivnom matricom.*

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0

	i	*j*	*k*	*l*
i	0	3	4	3
j	3	0	4	5
k	4	4	0	2
l	3	5	2	0

Slika 2.2: Prikaz (levo) aditivne i (desno) neaditivne matrice [6]

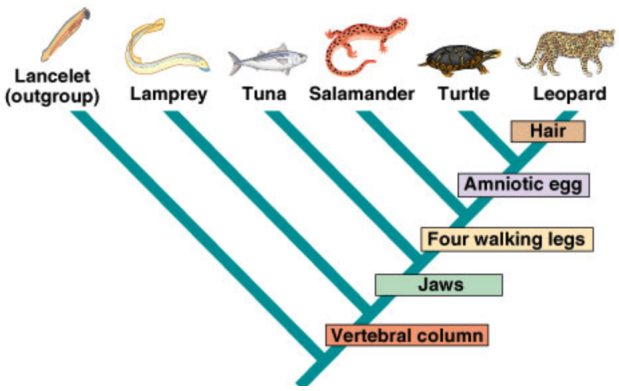
Ukoliko imamo matricu rastojanja koja je pritom i aditivna, tada možemo izgraditi jedinstveno filogenetsko stablo [2]. Gorepomenuti uslov koji matrica mora da zadovoljava da bi bila aditivna se naziva *uslovom četiri tačke*.

2.1.4 Ultrametrične matrice

Koreno filogenetsko stablo čiji je svaki list podjednako udaljen od korena se naziva *ultrametrično stablo*. Ultrametrična stabla se konstruišu na sledeći način:

1. Svakom čvoru se dodeli ceo broj koji predstavlja njegovu starost, odnosno vreme koje je proteklo od trenutka specijacije vrste koja se nalazi u tom čvoru na druge vrste
2. Listovima se dodeljuje starost jednaka nuli
3. Težina grana se određuje kao razlika starosti čvorova koje grana spaja

Na slici 2.3 je prikazan primer ultrametričnog stabla nad životinjskim vrstama.

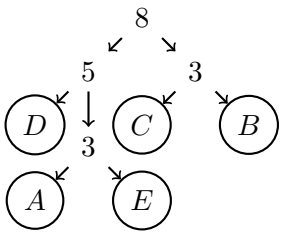


Slika 2.3: Prikaz ultrametričnog stabla nekoliko vrsta životinja [3]

Neka je matrica D matrica rastojanja dimenzije $N \times N$. Onda za ultrametrično stablo T koje je generisano na osnovu matrice rastojanja D važi:

1. Stablo T se sastoji iz N listova, po jedan za svaki red matrice D
2. Svaki unutrašnji čvor ima tačno dva potomka
3. Svaki unutrašnji čvor je obeležen brojem koji predstavlja starost čvora, gde vrednosti opadaju krećući se od korena ka listovima

	A	B	C	D	E
A	0	8	8	5	3
B		0	3	8	8
C			0	8	8
D				0	5
E					0



Slika 2.4: Matrica rastojanja i odgovarajuće ultrametrično stablo

Na slici 2.4 prikazan je primer matrice rastojanja i njenog odgovarajućeg ultrametričnog stabla.

Teorema 1 *Ukoliko za matricu M postoji ultrametrično stablo koje joj odgovara, onda je ono i jedinstveno.*

Gorepomenuta teorema nam govori da ukoliko je matrica rastojanja nastala na osnovu višestrukih poravnanja sekvenci genoma i ukoliko one predstavljaju razlike između organizama i na osnovu nje je moguće generisati ultrametrično stablo, onda to znači da je stablo jedinstveno i da se može tumačiti kao stablo evolucije vrsta, ali nam to ne govori mnogo o odabranoj metrici (sa svim parametrima poravnanja). Primera radi ukoliko bi se poravnavale sekvence gena koji se skoro ne menjaju između vrsta, matrica i stablo bi se konstruisali, ali nam to ne govori da je matrica nastala na pravi način i da nužno predstavlja pravi tok evolucije. Teorema o jedinstvenosti omogućava nam da imamo dobru procenu toka evolucije i vremena proteklog između razdvajanja vrsta.

Kada je na osnovu matrice rastojanja moguće generisati ultrametrično stablo? Može se pokazati da se ultrametrično stablo može rekonstruisati na osnovu matrice M koja zadovoljava sledeći uslov: Ukoliko za bilo koja tri reda i, j i k matrice M maksimalna vrednost od [*D(i, j)*, *D(i, k)*, *D(j, k)*] *nije* jedinstvena. Ovaj uslov se naziva *test relativne stope* (engl. *relative rate test*). Ukoliko matrica ispunjava test relativne stope onda je ona ultrametrična [4].

Ultrametrična stabla su od velikog značaja u teoriji evolucije, pre svega zbog svoje jednostavnosti. Iako ona predstavljaju stabla nastala nad matricama rastojanja koja moraju da ispunjavaju određene uslove, koja teško da će tako izgledati u realnom slučaju, moguće je konsturisati stablo na osnovu kojeg možemo doneti neke zaključke o evoluciji. Posmatrajući ultrametrično stablo od korena ka listovima, gde koren predstavlja prošlost, a listovi sadašnjost možemo videti koliko se evolutivnih promena desilo kroz vreme.

Ukoliko je matrica ujedno i ultrametrična matrica, onda se može generisati jedinstveno ultrametrično stablo u polinomijalnom vremenu koristeći neke od algoritama

koji će biti obrađeni u poglavlju 3.

Naveli smo uslove koje matrica rastojanja treba da ispunjava kako bi bilo moguće rekonstruisati stablo u polinomijalnom vremenu. U nastavku ćemo rezimirati sličnosti i razlike između ovakvih matrica rastojanja.

U tabeli 2.1.4 je prikazan odnos između aditivnih i ultrametričnih matrica rastojanja.

Aditivna rastojanja	**Ultrametrična rastojanja**
- Uslov četiri tačke važi za sve četvorke tačaka matrice	- Test relativne stope važi za sve trojke matrice
- Rastojanja odgovaraju jedinstvenom *nekorenom* stablu	- Rastojanja odgovaraju jedinstvenom *korenom* stablu i svi listovi su ekvidistantni od korena
- Dužine grana i topologija stabla je jedinstveno određena	- Dužine grana i topologija je jedinstveno određena
- Pohlepni algoritam koji pronalazi stablo je *Neighbor-Joining*	- Pohlepni algoritam koji pronalazi stablo je *UPGMA*

Tabela 2.1.4: Prikaz odnosa između ultrametričnih i aditivnih matrica rastojanja

Kao što je već bilo reči, matrica je aditivna ukoliko zadovoljava uslov četiri tačke. Stablo definiše *metriku stabla* u oznaci $T_{i,j}$ i predstavlja sve parove rastojanja između svih parova listova. Samim tim, svaka metrika stabla je aditivna. Ukoliko je matrica D aditivna to znači da postoji jedinstvena topologija stabla sa dužinama grana tako da važi $D_{i,j} = T_{i,j}$. Ovakvo stablo je moguće rekonstruisati u polinomijalnom vremenu. S druge strane, matrica je ultrametrična ukoliko ispunjava uslov tri tačke, odnosno test relativne stope. Sva ultrametrična stabla su korena stabla, ali metrika stabla ne mora biti ultrametrična. Ultrametrično stablo zadovoljava hipotezu molekularnog sata[1] [5]. Svaka ultrametrična matrica je ujedno i aditivna, dok obrnuto ne važi.

[1]Hipoteza molekularnog sata podrazumeva da su rastojanja od korena do listova ista, kao i da su dužine grana proporcionalne sa vremenom koje je proteklo od nastanka prve vrste do današnjih vrsta.

Glava 3

Algoritmi za rekonstrkuciju filogenetskih stabala

Postoji veliki broj algoritama i načina koji se koriste u svrhu rekonstruisanja filogenetskog stabla. Inicijalni izazov je svakako predstavljanje genetskog materijala određene vrste u formi u kojoj računar može da ga obrađuje, kao u slučaju virusa SARS, kod koga se sekvenca *Spike* proteina koristi za poređenje izolata virusa iz različitih domaćina. Sledeći izazov predstavlja pronalaženje algoritma koji u polinomijalnom vremenu može da rekonstruiše filogenetsko stablo.

U prethodnom poglavlju je navedeno da je moguće rekonstruisati jedinstveno filogenetsko stablo ukoliko je matrica rastojanja aditivna ili ultrametrična. S tim u vezi, u ovom poglavlju će biti reči o algoritmima za rekonstrukciju stabla na osnovu matrice rastojanja, *aditivna filogenija*, *UPGMA* i *Neighbor-Joining*. Na kraju će biti predstavljen algoritam koji za rekonstrukciju stabla koristi matricu karakteristika, *mala parsimonija*.

3.1 Aditivna filogenija

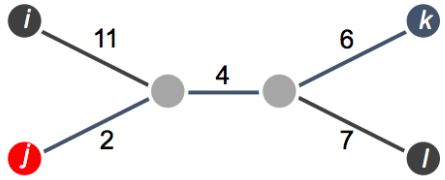
Osnovni pristupi rešavanju problema rekonstrukcije filogenetskog stabla (pristup preko susednih listova [6]) pretpostavljaju da minimalna nenula vrednost matrice rastojanja odgovara rastojanju između susednih listova. Slika 3.1 pokazuje primer gde ova pretpostavka ne važi. Zbog toga napredni pristupi koji se razmatraju u nastavku rada odbaciju ovu pretpostavku.

U drugom pristupu umesto smanjivanja problema uklanjanjem vrste i kolone u kojoj se nalazi minimalni element, uklanja se jedna spoljna grana. *Spoljna grana* predstavlja granu na čijem se jednom kraju nalazi list, dok ostale grane nazivamo *unutrašnjim granama*. Nakon uklanjanja i smanjivanja problema na problem manje dimenzije, list se vraća u rekonstruisano stablo računanjem dužine grane i povezivanjem sa odgovarajućim unutrašnjim čvorom pomoću spoljne grane. Dužina spoljne grane unapred je izračunata u trenutku njenog prvobitnog uklanjanja. Na taj način dobijeno stablo odgovara polaznom problemu. O dužinama spoljnjih grana govori naredna teorema.

Teorema 2 *Neka je D matrica rastojanja. Tada je dužina spoljne grane do lista i, LimbLength(i) jednaka minimalnoj vrednosti* $(D_{i,k} + D_{i,j} - D_{j,k})/2$ *po svim listovima j i k filogenetskog stabla kome pripada.*

S obzirom na to da nam je u trenutku rekonstrukcije filogenetskog stabla poznata samo matrica rastojanja, potrebno je ukloniti list kao i granu koja ga povezuje sa

	i	*j*	*k*	*l*
i	0	13	21	22
j	13	0	12	13
k	21	12	0	13
l	22	13	13	0



Slika 3.1: Primer matrice u kojoj minimalna vrednost ne odgovara susednim listovima

roditeljskim čvorom u matrici rastojanja.

Najpre, pretpostavimo da je poznato *T(D)* i odaberimo proizvoljno jedan njegov list j. Trebalo bi odseći čvor j i granu za koju je zakačen tako što ćemo smanjiti njenu težinu za *LimbLength(j)* i tako je svesti na nulu. Pošto *T(D)* nije poznato, ovo odsecanje ćemo predstaviti unutar matrice rastojanja D tako što ćemo sve vrednosti u matrici u redu i koloni j (osim dijagonalnog) umanjiti za *LimbLength(j)*. Spoljne grane sa težinom nula zovemo *ogoljenim* (engl. *bald*) spoljnim granama, a matricu rastojanja dobijenu na ovaj način označićemo sa D_{bald}.

Sledeći korak je uklanjanje j-te vrste i kolone iz matrice D_{bald} čime se dobija „potkresana" matrica D_{trim} dimenzije *(n-1)*x*(n-1)*. Na taj način smo smanjili dimenziju polaznog problema za jedan. Rekurzivno možemo rekonstruisati stablo *T(D)* na sledeći način koristeći algoritam *aditivne filogenije* (engl. *AdditivePhylogeny*):

U nastavku biće prikazan primer primene aditivne filogenije korak po korak.

Primer primene algoritma aditivne filogenije

Pretpostavimo da imamo aditivnu matricu D:

$$\begin{bmatrix} & a & b & c & d \\ a & 0 & 13 & 21 & 22 \\ b & 13 & 0 & 12 & 13 \\ c & 21 & 12 & 0 & 13 \\ d & 22 & 13 & 13 & 0 \end{bmatrix}$$

Algoritam 1 AdditivePhylogeny

```
1: procedure AdditivePhylogeny(D, n)
2:     if n = 2 then return stablo T koje sadrži jednu granu dužine D_{1,2}
3:     limbLength ← LIMB(D, n)
4:     for j ← 1 do n - 1 do
5:         D_{j,n} ← D_{j,n} - limbLength
6:         D_{n,j} ← D_{j,n}
7:     (i, n, k) ← tri lista takvih da važi D_{i,k} = D_{i,n} + D_{n,k}
8:     x ← D_{i,n}
9:     Ukloni red n i kolonu n iz D
10:    T ← AdditivePhylogeny(D, n - 1)
11:    v ← Potencijalno novi čvor iz T na rastojanju x od i na putanji između i i k
12:    Dodaj list n nazad u stablo T kreiranjem spoljne grane limb (v, n) dužine
       limbLength
13:    return T
```

Biramo proizvoljno čvor d i za njega računamo *LimbLength(d)* u odnosu na sve čvorove i i j:

$$(D_{a,d} + D_{b,d} - D_{a,b})/2 = (22 + 13 - 13)/2 = 11$$
$$(D_{a,d} + D_{c,d} - D_{a,c})/2 = (22 + 13 - 21)/2 = 7$$
$$(D_{b,d} + D_{c,d} - D_{b,c})/2 = (13 + 13 - 12)/2 = 7$$

LimbLength(d) će biti najmanja vrednost po svakom čvoru i i j što je vrednost 7 i nju oduzimamo od svake grane koja vodi od ili ka čvoru d tako da sada dobijamo ogoljenu matricu u oznaci D_{bald}:

$$\begin{bmatrix} & a & b & c & d \\ a & 0 & 13 & 21 & 15 \\ b & 13 & 0 & 12 & 6 \\ c & 21 & 12 & 0 & 6 \\ d & 15 & 6 & 6 & 0 \end{bmatrix}$$

Nakon uklanjanja kolone i reda d dobijamo smanjenu matricu u oznaci D_{trim}:

$$\begin{bmatrix} & a & b & c & d \\ a & 0 & 13 & 21 & 15 \\ b & 13 & 0 & 12 & 6 \\ c & 21 & 12 & 0 & 6 \\ d & 15 & 6 & 6 & 0 \end{bmatrix}$$

Zatim ponovo računamo *LimbLength* za proizvoljan čvor. Biramo čvor c:

$$(D_{a,c} + D_{b,c} - D_{a,b})/2 = (21 + 12 - 13)/2 = 10$$
$$(D_{a,c} + D_{d,c} - D_{a,d})/2 = (21 + 13 - 22)/2 = 6$$
$$(D_{b,c} + D_{d,c} - D_{b,d})/2 = (12 + 13 - 13)/2 = 6$$

Slika 3.2: Prvi korak rekonstrukcije stabla

$LimbLength(c) = 6$

Oduzimamo $LimbLength(c)$ za sve grane koje vode od i do čvora c i dobijamo ponovo ogoljenu matricu D_{bald}:

$$\begin{bmatrix} & a & b & c & d \\ a & 0 & 13 & 15 & 15 \\ b & 13 & 0 & 6 & 6 \\ c & 15 & 6 & 0 & 6 \\ d & 15 & 6 & 6 & 0 \end{bmatrix}$$

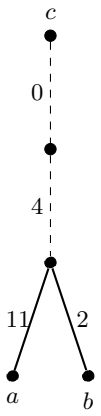
Nakon uklanjanja kolone i reda c ostaje potkresana matrica D_{trim}:

$$\begin{bmatrix} & a & b & c & d \\ a & 0 & 13 & 15 & 15 \\ b & 13 & 0 & 6 & 6 \\ c & 15 & 6 & 0 & 6 \\ d & 15 & 6 & 6 & 0 \end{bmatrix}$$

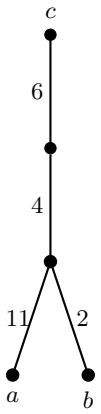
Preostalo je da izračunamo dužinu spoljne grane za čvor b. Na sličan način dobija se $LimbLength(b) = 2$.

Prema algoritmu izlazimo iz rekurzije kada broj listova bude jednak 2. Ostala nam je grana dužine 13 i konstruišemo inicijalno stablo sa jednom granom kao što je prikazano na slici 3.2, između preostalih listova a i b. U drugom koraku posmatramo ogoljenu matricu pre uklanjanja lista c i izračunatu spoljnu granu za b, čime dobijamo stablo prikazano na slici 3.3. U narednom koraku, na sličan način, samo na osnovu potkresane matrice nakon uklanjanja reda i kolone d i izračunate dužine spoljne grane c generišemo stablo prikazano na slici 3.4. U pretposlednjem koraku na osnovu ogoljene matrice iz drugog koraka i spojlne grane, generišemo stablo prikazano na slici 3.5. U poslednjem koraku dodajemo preostalu spoljnu granu i dobijamo finalno stablo prikazano na slici 3.6.

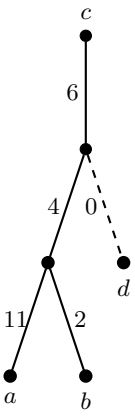
Dobra strana ovog algoritma je da kreira stablo koje odgovara aditivnoj matrici, a loša da ne radi za neaditivne matrice.



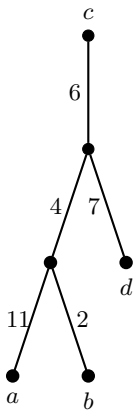
Slika 3.3: Drugi korak rekonstrukcije stabla



Slika 3.4: Treći korak rekonstrukcije stabla



Slika 3.5: Četvrti korak rekonstrukcije stabla



Slika 3.6: Poslednji korak rekonstrukcije stabla

3.2 UPGMA

UPGMA (eng. *Unweighted Pair Group Method with Arithmetic Mean*)) predstavlja metodu hijerarhiskog klasterovanja koja se zasniva na matricama rastojanja. Ovim algoritmom se generiše stablo sa korenom (*dendogram*) i smatra se jednim od jednostavnijih algoritama za generisanje evolutivnih stabala.

Treba uvesti nekoliko pojmova pre prikazivanja rada samog algoritma. Svaka vrsta/kolona u matrici rastojanja predstavlja jednu *taksonomsku kategoriju (takson)*. Prilikom izgradnje stabla, taksoni se povezuju granama i tako grade klastere. Algoritam počinje od jednočlanih klastera (svaki takson predstavlja jednočlani klaster) koji se tokom algoritma spajaju, čineći nove, veće klastere.

Rastojanje između dva klastera C_1 i C_2 se meri kao prosečno rastojanje između njihovih članova i računa se na sledeći način:

$$d_{avg}(C_1, C_2) = \frac{\sum_{i \in C_1, j \in C_2} d_{i,j}}{|C_1| \cdot |C_2|}$$

$|C|$ označava broj elemenata u klasteru C, a $d_{i,j}$ rastojanje između listova i i j dato matricom rastojanja. Nakon spajanja dva klastera C_i i C_j u novi klaster C_{novi}, rastojanje između C_{novi} i nekog drugog klastera C_m je jednako:

$$d(C_{novi}, C_m) = \frac{d(C_i, C_m) \cdot |C_i| + d(C_j, C_m) \cdot |C_j|}{|C_i| + |C_j|}.$$

Visina čvora predstavlja na kojoj visini se nalazi čvor u stablu, pa se u prvom koraku algoritma listovima dodeljuje visina jednaka nuli. Možemo onda da definišemo težinu grane *(v,w)* u stablu kao razliku $visina(v) - visina(w)$. Zbog toga bi dužina putanje između korena i bilo kod čvora u stablu bila jednaka razlici njihovih visina.

U nastavku će biti prikazan pseudokod *UPGMA* algoritma 17.

Primer primene UPGMA algoritma

Označimo inicijalnu matricu rastojanja sa D_1:

Algoritam 2 UPGMA

procedure UPGMA(D, n)
  Klasteri ← n Dodeliti svaki takson sopstvenom klasteru
  Konstruisati graf T sa n čvorova
  for svaki čvor v u stablu T **do**
    visina(v) ← 0
  while Postoji više of jednog klastera **do**
    Pronađi dva najbliža klastera C_i i C_j
    Spoji C_i i C_j u novi klaster C_{novi} sa $|C_i|$ i $|C_j|$ elemenata
    Dodaj novi čvor C_{novi} u stablo T
    Poveži čvor C_{novi} sa C_i i C_j direktnim granama
    Postavi *visina(C)* ← $D_{C_i,C_j}/2$
    Ukloni kolone i redove iz matrice D koje odgovaraju C_i i C_j
    Ukloni C_i i C_j iz *Klastera*
    Dodaj red i kolonu u matrici D za C_{novi} računajući rastojanje $D(C_{novi}, C)$ za svaki klaster C u *Klasterima*.
    Dodaj C_{novi} u *Klastere*
  koren ← čvor u stablu T koji odgovara preostalom klasteru
  for svaku granu *(v, w)* u stablu T **do**
    dužina(v) ← *visina(v)* - *visina(w)*
  return T

$$\begin{bmatrix} & a & b & c & d & e \\ a & 0 & & & & \\ b & 17 & 0 & & & \\ c & 21 & 30 & 0 & & \\ d & 31 & 34 & 28 & 0 & \\ e & 23 & 21 & 39 & 43 & 0 \end{bmatrix}$$

Najpre pronalazimo najmanji element u matrici rastojanja D_1. U ovom slučaju to je vrednost 17 za taksone a i b:

$$\begin{bmatrix} & a & b & c & d & e \\ a & 0 & & & & \\ b & 17 & 0 & & & \\ c & 21 & 30 & 0 & & \\ d & 31 & 34 & 28 & 0 & \\ e & 23 & 21 & 39 & 43 & 0 \end{bmatrix}$$

Definišemo novi klaster *(a,b)* kao i novi čvor u koji će imati visinu $\frac{d_{ab}}{2} = 8.5$. Potom računamo rastojanja od novog klastera do ostalih čvorova na sledeći način:

$D_2((\text{a, b}), \text{c}) = (D_1(\text{a, c}) \times 1 + D_1(\text{b, c}) \times 1) \ / \ (1 + 1) = (21 + 30) \ / \ 2 = 25.5$
$D_2((\text{a, b}), \text{d}) = (D_1(\text{a, d}) + D_1(\text{b, d})) \ / \ 2 = (31 + 34) \ / \ 2 = 32.5$
$D_2((\text{a, b}), \text{e}) = (D_1(\text{a, e}) + D_1(\text{b, e})) \ / \ 2 = (23 + 21) \ / \ 2 = 22$

Novodobijena matrica D_2 nakon uklanjanja redova i kolona a i b i računanja novih rastojanja je prikazana u nastavku:

$$\begin{bmatrix} & (a,b) & c & d & e \\ (a,b) & 0 & & & \\ c & 25.5 & 0 & & \\ d & 32.5 & 28 & 0 & \\ e & 22 & 39 & 43 & 0 \end{bmatrix}$$

Najmanji element u matrici D_2 je $D_2((a,b),e) = 22$. Definišemo novi klaster sa elementima *(a, b)* i *e*. Označimo sa v novi čvor sa kojim će čvorovi *(a, b)* i e biti povezani. Zbog ultrametričnosti dužine grana od a ili b do v i e do v je isti i iznosi:

$$d_{a,v} = d_{b,v} = d_{e,v} = 22/2 = 11.$$

Treba još oduzeti dužinu od novog čvora u koji je dobijen u prethodnom koraku:

$$\begin{aligned} d_{u,v} &= d_{e,v} - d_{a,u} \\ &= d_{e,v} - d_{b,u} \\ &= 11 - 8.5 = 2.5 \end{aligned}$$

Matrica će dobijanjem novog klastera biti umanjena za red i kolonu e i *(a, b)*, pa tako treba izračunati udaljenosti preostalih čvorova do novog klastera. Nove vrednosti se dobijaju na sledeći način:

D_3(((a, b), e), c) = (D_2((a, b), c)× 2 + D_2(e, c)× 1) / (2 + 1) = (25.5 × 2 + 39 × 1) / 3 = 30
D_3(((a, b), e), d) = ((D_2((a, b), d)× 2 + D_2(e, d)× 1) / (2 + 1) = (32.5 × 2 + 43 × 1) / 3 = 36

Novodobijena matrica D_3 je prikazana u nastavku:

$$\begin{bmatrix} & ((a,b),e) & c & d \\ ((a,b),e) & 0 & & \\ c & 30 & 0 & \\ d & 36 & 28 & 0 \end{bmatrix}$$

Najmanja vrednost u matrici D_3 će biti $D_3(c,d)$ tako da formiramo novi klaster w. Slično računamo dužinu novih grana $d_{c,w} = d_{d,w}$, s kojim će čvorovi d i c biti povezani:

$$d_{c,w} = d_{d,w} = 28/2 = 14.$$

Računamo rastojanje između preostalog čvora i novog klastera:

D_4((c, d), ((a, b), e)) = (D_3(c, ((a, b), e))× 1 + D_3(d, ((a, b), e))× 1) / (1 + 1) = (30 × 1 + 36 × 1) / 2 = 33

Novodobijena matrica D_4 će izgledati ovako:

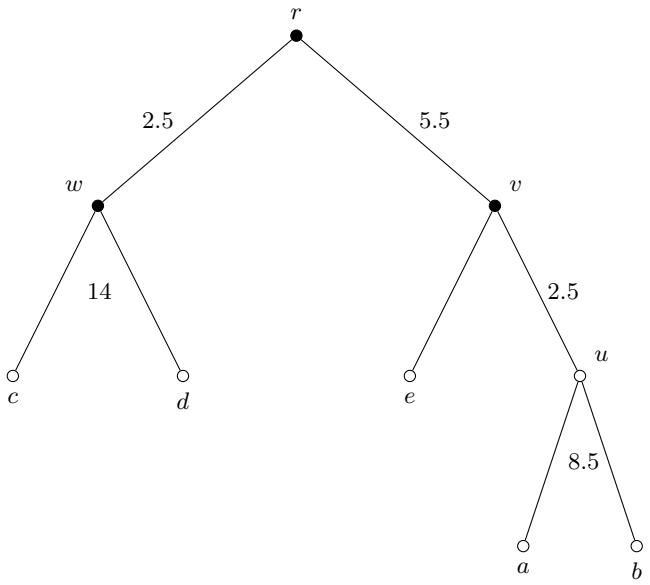
$$\begin{bmatrix} & ((a,b),e) & (c,d) \\ ((a,b),e) & 0 & \\ (c,d) & 33 & 0 \end{bmatrix}$$

Kao poslednji korak grupišemo preostala dva klastera i definišemo novi čvor r koji će povezivati ova dva klastera i ujedno predstavljati i koren stabla. Rastojanje između klastera i čvora r se računa slično kao u prethodnim koracima na sledeći način:

$$d_{((a,b),e),r} = d_{(c,d),r} = 16.5$$

Umanjujemo vrednosti dužina preostalih grana:

$$\begin{aligned} d_{v,r} &= d_{((a,b),e),r} - d_{e,v} = 16.5 - 11 = 5.5 \\ d_{w,r} &= d_{(c,d),r} - d_{c,w} = 16.5 - 14 = 2.5 \end{aligned}$$



Slika 3.7: Filogenetsko stablo dobijeno korišćenjem UPGMA algoritma

Dobijeni dendogram i rezultujuće filogenetsko stablo je prikazano na slici 3.7.

Velika mana ovog algoritma je što pretpostavlja da je svaka vrsta evoluirala za isti vremenski period, što znači da su različite vrste na isti način evoluirale, kao i da su svi listovi jednako udaljeni od korena stabla. To dalje implicira da je brzina i broj mutacija isti za svaku vrstu, za šta je verovatnoća u realnosti vrlo mala. Dobra strana ovog algoritmaje što radi i sa neaditivnim matricama, ali može da se desi da proizvede filogenetsko stablo koje ne odgovara polaznoj matrici rastojanja.

3.3 Neighbor-joining algoritam

Neighbor-joining algoritam (u nastavku *NJ*) je još jedan algoritam koji se zasniva na matricama rastojanja, a koji može da rekonstruiše filogenetsko stablo iako matrica nije aditivna. *NJ* algoritam teži da konstruiše stablo koje će minimizirati dužine spoljnih grana. Najpre kreće sa stablom u obliku zvezde, gde postoje jedino direktne grane od korena ka svim listovima, pa zatim iterativno pronalazi najbliže susedne čvorove. Tačnije, u njemu se pronalaze dva najbliža čvora koja će formirati klaster, zatim se pronalaze druga dva najbliža čvora (uključujući u pretragu i formirani klaster) i kreira se ponovo novi klaster, sve dok ne dođemo do jedinstvenog klastera. S tim u vezi, *NJ* algoritam predstavlja vrstu klasterovanja metodom *odozdo-na-gore*.

Razlika u odnosu na *UPGMA* algoritam je taj što rekonstruisano stablo ne mora da bude ujedno i ultrametrično, odnosno dužine grana do zajedničkog pretka mogu da se razlikuju.

U nastavku navodimo *NJ* teoremu bez dokaza.

Teorema 3 *Neka je data aditivna matrica rastojanja D dimenzija $n \times n$ i sledeće veličine:*

1. *$R_D(i)$ koja predstavlja sumu rastojanja lista i od ostalih listova*

$$R_D(i) = \sum_{1 \leq k \leq n} D_{i,k}$$

2. *NJ matrica D^* čiji su elementi na glavnoj dijagonali nule, a na ostalim pozicijama*

$$D^*_{i,j} = (n-2) \cdot D_{i,j} - R_D(i) - R_D(j) \tag{3.1}$$

Tada, ako je $D^*_{i,j}$ minimalni element u NJ matrici D^*, listovi i i j će biti susedni u T(D) ■

Ova teorema je od značaja pošto se ispostavlja da minimalni element u matrici rastojanja ne mora nužno da ukazuje na listove koji su susedni u stablu (slika 3.1). Iz tog razloga ideja *NJ* algoritma je da ukoliko ne možemo na osnovu matrice rastojanja da odredimo susedne listove, onda treba odrediti matricu čiji minimum će predstavljati susedne listove.

Za matricu dimenzije $n = 2$ algoritam će vratiti stablo sa dva čvora i granom koja ih povezuje, dok ukoliko je matrica dimenzije $n > 2$, tada se pronalazi najmanji element u *NJ* matrici i zamenjuju se kolone i redovi u kojima se nalazi najmanji element novom kolonom i redom m i računaju se rastojanja od novonastalog čvora m do svih ostalih čvorova koristeći formulu

$$D_{k,m} = (D_{k,i} + D_{k,j} - D_{i,j}) \,/\, 2$$

za svaki čvor k iz *NJ* matrice.

Time se dobija problem manje dimenzije *(n - 1) × (n - 1)*, pa se rekurzivnom primenom *NJ* algoritma na matricu *D'* dobija stablo *T(D')* koji se razlikuje od početnog stabla *T(D)* po tome što nema listove i i j povezane sa čvorom m.

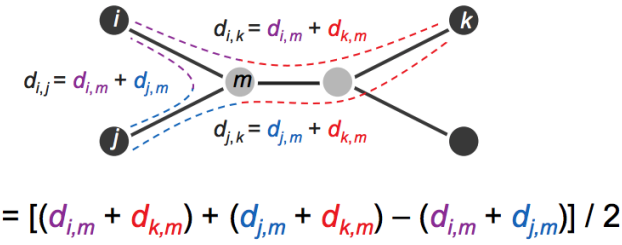
Dužine spoljnih grana od čvorova i i j do m računamo koristeći naredne formule:

$$\triangle_{i,j} = \frac{R_D(i) - R_D(j)}{(n-2)}$$

$$LimbLength(i) = \frac{D_{i,j} + \triangle_{i,j}}{2}, LimbLength(j) = \frac{D_{i,j} - \triangle_{i,j}}{2} \tag{3.2}$$

Možemo uočiti da će se dužine spoljnih grana za aditivne matrice računati kao što je prikazano na slici 3.8 jer izborom bilo kog lista različitog od i i j vrednost *LimbLength(i)* ostaje ista. Za neaditivne matrice je slučaj drugačiji, vrednosti će biti različite za različit izbor čvora k i tada uzimamo prosek po svim takvim vrednostima.

Pokažimo kako je dobijena formula za računanje spoljnih grana za neaditivne matrice.



Slika 3.8: Prikaz rekonstrukcije filogenetskog stabla na osnovu rastojanja [6]

$$
\begin{aligned}
LimbLength(i) &= \frac{D_{i,j}}{2} + \frac{1}{n-2} \cdot \sum_{k...l \neq i,j} \frac{D_{i,k} - D_{j,k}}{2} \\
&= \frac{D_{i,j}}{2} + \frac{1}{n-2} \cdot \Big(\sum_{k...l \neq i,j} \frac{D_{i,k}}{2} - \sum_{k...l \neq i,j} \frac{D_{j,k}}{2} \Big) \\
&= \frac{1}{2} \cdot \Big(D_{i,j} + \frac{1}{n-2} \cdot \Big(\sum_{k...l \neq i,j} D_{i,k} - \sum_{k...l \neq i,j} D_{j,k} \Big) \Big) \\
&= \frac{1}{2} \cdot \Big(D_{i,j} + \frac{R_D(i) - R_D(j)}{n-2} \Big) \\
&= \frac{1}{2} \cdot (D_{i,j} + \triangle_{i,j})
\end{aligned}
$$

Pseudokod algoritma je prikazan u nastavku.

Algoritam 3 Neighbor-Joining

procedure NJ(D, n)
  if $n = 2$ **then**
    $T \leftarrow$ stablo koje se sastoji od grane dužine $D_{1,2}$ **return** T
  $D^* \leftarrow NJ$ matricu konstruisanu iz matrice rastojanja D
  Pronađi elemente i i j takve da $D^*_{i,j}$ predstavlja najmanji nenula element van dijagonale u matrici D^*
  $\Delta \leftarrow (TotalDistance_D(i) - TotalDistance_D(j))$ / (n - 2)
  $limbLength_i \leftarrow frac12(D_{i,j} + \Delta)$
  $limbLength_j \leftarrow frac12(D_{i,j} - \Delta)$
  Dodaj novi red i kolonu m u matricu D takve da važi $D_{m,k} = \frac{1}{2}(D_{k,i} + D_{k,j} - D_{i,j})$ za bilo koje k
  Ukloni redove i kolone i i j iz matrice D
  $T \leftarrow$ **NeighborJoining(D, n - 1)**
  Dodaj dve nove grane od čvora m do listova i i j u stablo T
  Dodeli dužinu $limbLength_i$ Limb(i)
  Dodeli dužinu $limbLength_j$ Limb(j)
  return T

U nastavku će biti prikazan primer primene NJ algoritma na matrici rastojanja.

Primer primene *NJ* algoritma

Pretpostavimo da imamo inicijalnu matricu rastojanja D_1:

$$\begin{bmatrix} & A & B & C & D & E \\ A & 0 & & & & \\ B & 5 & 0 & & & \\ C & 9 & 10 & 0 & & \\ D & 9 & 10 & 8 & 0 & \\ E & 8 & 9 & 7 & 3 & 0 \end{bmatrix}$$

Konstruišemo NJ matricu D^* računanjem novih rastojanja koristeći formulu (3.1). Prikažimo kako se vrednost $(a,\ b)$ računa srednje rastojanje od ostalih čvorova:

$$\begin{aligned} D^*_{a,b} &= (5-2) \times 5 - (5+9+9+8) - (5+10+10+9) \\ &= 15 - 31 - 34 \\ &= -50 \end{aligned}$$

Na isti način računamo za ostale parove čvorova i dobija se naredna matrica:

$$\begin{bmatrix} & A & B & C & D & E \\ A & 0 & & & & \\ B & -50 & 0 & & & \\ C & -38 & -38 & 0 & & \\ D & -34 & -34 & -40 & 0 & \\ E & -34 & -34 & -40 & -48 & 0 \end{bmatrix}$$

Najmanja vrednost je *-50* tako da spajamo čvorove a i b u novi klaster. Računamo dužine spoljnih grana do čvora u koristeći formulu 3.2.

$$\begin{aligned} LimbLength(a,u) &= \frac{1}{2} \cdot D_{a,b} + \frac{1}{2 \cdot (5-2)} \cdot [\sum_{k=1}^{5} \cdot D_{a,k} - \sum_{k=1}^{5} \cdot D_{b,k}] \\ &= \frac{5}{2} + \frac{31-34}{6} \\ &= 2 \\ LimbLength(b,u) &= D_{a,b} - LimbLength(a,u) \\ &= 5 - 2 \\ &= 3 \end{aligned}$$

Potom računamo rastojanja ostalih čvorova do novog čvora u kako bismo formirali matricu D_1 smanjenu za jedan red i kolonu spajanjem čvorova a i b. Nove vrednosti računamo koristeći formulu 3.3.

$$
\begin{aligned}
D_{u,c} &= \frac{1}{2} \cdot [D_{a,c} + D_{b,c} - D_{a,b}] \\
&= \frac{9 + 10 - 5}{2} = 7 \\
D_{u,d} &= \frac{1}{2} \cdot [D_{a,d} + D_{b,d} - D_{a,b}] \\
&= \frac{9 + 10 - 5}{2} = 7 \\
D_{u,e} &= \frac{1}{2} \cdot [D_{a,e} + D_{b,e} - D_{a,b}] \\
&= \frac{8 + 9 - 5}{2} = 6
\end{aligned}
$$

Rezultujuća matrica D^1 će onda izgledati ovako:

$$
\begin{bmatrix}
 & u & c & d & e \\
u & 0 & & & \\
c & \mathbf{7} & 0 & & \\
d & \mathbf{7} & 8 & 0 & \\
e & \mathbf{6} & 7 & 3 & 0
\end{bmatrix}
$$

Naglašene vrednosti u matrici predstavljaju novoizračunate vrednosti, dok se ostatak matrice nije menjao. Potom će odgovarajuća NJ matrica izgledati ovako:

$$
\begin{bmatrix}
 & u & c & d & e \\
u & 0 & & & \\
c & -28 & 0 & & \\
d & -24 & -24 & 0 & \\
e & -24 & -24 & -28 & 0
\end{bmatrix}
$$

Najmanje vrednosti se nalaze za čvorove *(u, c)* i *(d, e)* tako biramo proizvoljne čvorove koje ćemo spojiti u novi klaster. Biramo čvorove u i c i dodajemo novi čvor v.

Nova rastojanja dobijena uklanjanjem kolone u i c se dobijaju na isti način kao što je u prethodnom koraku prikazano. Onda se dobijaju sledeće vrednosti:

$$
\begin{aligned}
LimbLength(u,v) &= \frac{1}{2}\cdot D_{u,c} + \frac{1}{2\cdot(4-2)}\cdot[\sum_{k=1}^{4}\cdot D_{u,k} - \sum_{k=1}^{4}\cdot D_{c,k}] \\
&= \frac{7}{2} + \frac{20-22}{4} \\
&= 3 \\
LimbLength(c,v) &= D_{u,c} - LimbLength(u,v) \\
&= 7-3 \\
&= 4 \\
D_{v,d} &= \frac{1}{2}\cdot[D_{u,d} + D_{c,d} - D_{u,c}] \\
&= \frac{7+8-7}{2} = 4 \\
D_{v,e} &= \frac{1}{2}\cdot[D_{u,e} + D_{c,e} - D_{u,c}] \\
&= \frac{6+7-7}{2} = 3
\end{aligned}
$$

Pa će matrica D^3 biti:

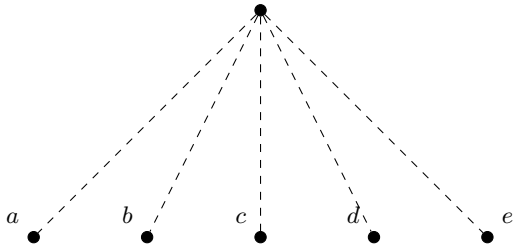
$$
\begin{bmatrix}
 & v & d & e \\
v & 0 & & \\
d & 4 & 0 & \\
e & 3 & 3 & 0
\end{bmatrix}
$$

Sada je već dobijena topologija stabla, ali radi tačnosti izračunaćemo dužine preostalih spoljnih grana do čvorova. U NJ matrici D_3^* koja je prikazana ispod, najmanja vrednost je -10, pa biramo da spojimo čvorove v i d i kreiramo novi čvor w.

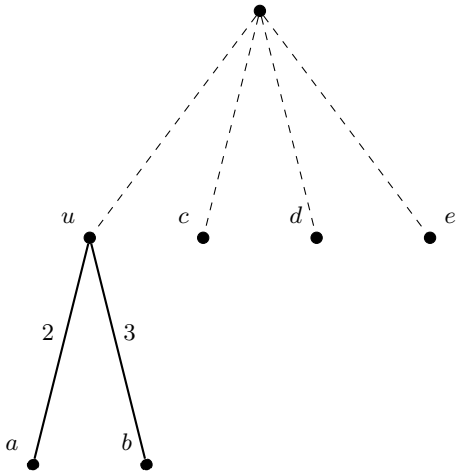
$$
\begin{bmatrix}
 & v & d & e \\
v & 0 & & \\
d & -10 & 0 & \\
e & -10 & -10 & 0
\end{bmatrix}
$$

$$
\begin{aligned}
LimbLength(v,w) &= \frac{1}{2}\cdot D_{v,d} + \frac{1}{2\cdot(3-2)}\cdot[\sum_{k=1}^{3}\cdot D_{v,k} - \sum_{k=1}^{3}\cdot D_{d,k}] \\
&= \frac{4}{2} + \frac{7-7}{2} = 2 \\
LimbLength(w,d) &= D_{v,d} - LimbLength(v,w) \\
&= 4-2 = 2 \\
LimbLength(w,e) &= D_{v,e} - LimbLength(v,w) \\
&= 3-2 = 1
\end{aligned}
$$

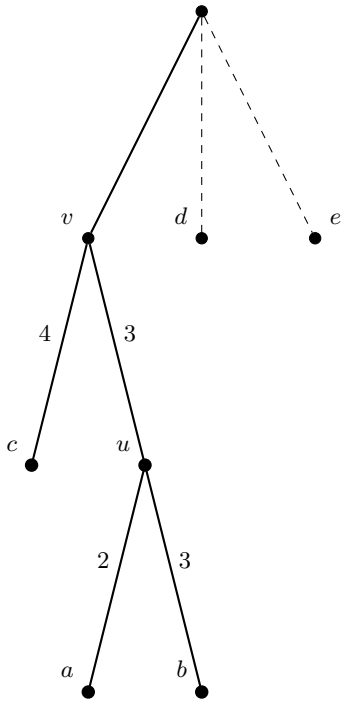
Na slikama 3.9, 3.10, 3.11, 3.12 su prikazani koraci rekonstrukcije NJ stabla.



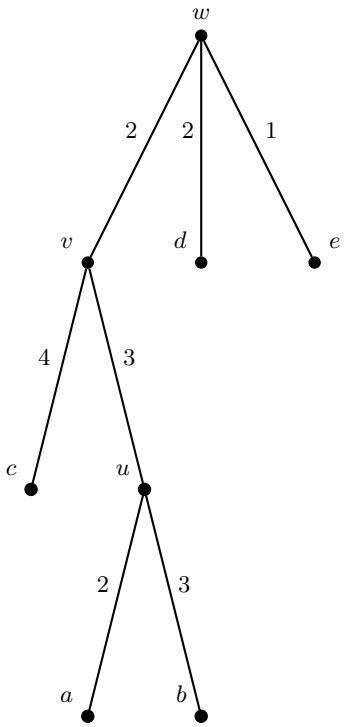
Slika 3.9: Prvi korak rekonstrukcije *NJ* stabla



Slika 3.10: Drugi korak rekonstrukcije *NJ* stabla



Slika 3.11: Treći korak rekonstrukcije *NJ* stabla



Slika 3.12: Poslednji korak rekonstrukcije *NJ* stabla

Možemo da primetimo da zbir svih dužina grana na putanji od jednog čvora do drugog je jednaka vrednosti u početnoj matrici rastojanja u koloni i vrsti koje odgovaraju tim čvorovima.
Na primer, posmatrajmo zbir grana duž putanje od čvora a do čvora e:

$$2 + 3 + 2 + 1 = 8 = D_{a,e}$$

Glavna prednost *NJ* algoritma je u tome što je brz u poređenju sa algoritmima kao što je mala parsimonija, o čemu će biti više reči u poglavlju 3.4. Iz tog razloga ga je moguće primenjivati u slučaju analize velikih skupova podataka.

NJ algoritam minimizira sumu dužina grana pri svakom klasterovanju, počevši sa stablom u obliku zvezde. S tim u vezi, finalno stablo možda neće biti minimalno evolutivno stablo u odnosu na sva ostala moguća stabla. Treba naglasiti da minimalno evolutivno stablo nije nužno i pravo evolutivno stablo.

3.4 Algoritam male parsimonije

Umesto korišćenjem matrica rastojanja, filogenetska stabla se mogu rekonstruisati i na osnovu matrice karakteristika. U svakom redu matrice nalazi se vrsta, a u kolonama karakteristike koje se proučavaju. Svaka vrsta matrice predstavlja vektor vrednosti koje predstavljaju prisustvo ili odustvo neke karakteristike. Karakteristike mogu da budu fizičke, morfološke, bihevioralne ili molekularne. Da bi bila moguća klasifikacija vrsta, potrebno je da se karakteristike podele u diskretna stanja. Diskretna

stanja mogu biti boja očiju - plava, smeđa, zelena... U karakteristike ponašanja može da se svrsta na primer agresivno, melanholično, veselo... Primer matrice generisane na osnovu sličnosti između organizama se može videti na slici 3.13.

	Four legs	Amniote egg	Hair	Mammary glands	Live birth	Pouch	Placenta
Angel fish	0	0	0	0	0	0	0
Frog	1	0	0	0	0	0	0
Crocodile	1	1	0	0	0	0	0
Playpus	1	1	1	1	0	0	0
Kangaroo	1	1	1	1	1	1	0
Elephant	1	1	1	1	1	0	1
Koala	1	1	1	1	1	1	0
Gorilla	1	1	1	1	1	0	1

Slika 3.13: Primer matrice karakteristika za utvrđivanje filogeneze kod različitih vrsta životinja

Pokazalo se da rekonstrukcija filogenetskih stabala kada se kao karakteristike koriste DNK sekvence daje najbolje rezultate. U ovom radu će se posmatrati molekularni podaci, gde će matrica karakteristika izgledati upravo kao matrica višestrukog poravnanja za jedan unapred izabrani gen zajednički za sve vrste. Treba na osnovu ovakve matrice generisati filogenetsko stablo gde će listovi biti navedene vrste. Vrste koje imaju najviše zajedničkih karakteristika treba da se nalaze bliže u stablu. Unutrašnji čvorovi će biti označeni nizovima karakteristika, pa se može izvesti pretpostavka kako su se karakteristike menjale kroz vreme.

Prilikom rekonstrukcije filogenetskih stabala na osnovu matrica karakteristika, važno pitanje je da li je unapred poznata topologija stabla ili nije. Ukoliko nije, tada se moraju ispitivati sva moguća stabla koja imaju dati broj listova, a kojih ima eksponencijalno mnogo u odnosu na broj listova. Ovakav problem se ne može rešiti u polinomijalnom vremenu. Sa druge strane, ukoliko je topologija stabla poznata, tada se ispituju sve kombinacije nizova karakteristika za datu topologiju što je mnogo manji broj stabala nego u prethodnom slučaju. Iz tog razloga ćemo se u radu ograničiti na problem kada topologija jeste poznata.

Pretpostavimo da imamo nekoliko vrsta koje posmatramo i da je nad njima izvršeno višestruko poravnanje, kao i da je poznata topologija stabla. Poznate sekvence dodeljujemo listovima, a treba pronaći niske karaktera koje će se nalaziti u unutrašnjim čvorovima stabla. Za datu topologiju stabla, unutrašnji čvorovi mogu biti označeni različitim niskama pa tako postoji više rezultujućih stabala. Da bismo ih rangirali, uvodimo *funkciju skora* koja meri kako takvo stablo obeleženo niskama odgovara višestrukom poravnanju. Grane označavamo razlikama između listova i unutrašnjih čvorova. Tada bi funkcija skora bila suma svih težina grana.

Predstavljeni problem pronalaska optimalnog skora predstavlja problem *male* parsimonije. Dakle, treba za datu matricu poravnanja označiti unutrašnje čvorove tako da *skor parsimonije* bude minimalan.

Mala parsimonija se zasniva na filozofiji Okamove oštrice, odnosno ukoliko postoji više mogućnosti, treba izabrati onu jednostavniju koja ispunjava uslove. Primena ovog

principa na izgradnju filogenetskog stabla zasniva se na pretpostavci da ono stablo koje ima najmanji broj mutacija između vrsta je ujedno i najbolje.

Definišimo problem male parsimonije:

Problem male parsimonije: Označiti unutrašnje čvorove datog filogenetskog stabla tako da njegov skor parsimonije za datu matricu bude minimalan.

Ulaz: Koreno binarno stablo gde je svaki list obeležen niskom dužine m.
Izlaz: Označavanje svakog unutrašnjeg čvora niskama dužine m tako da skor parsimonije stabla bude minimalan.

Primetimo da su karakteri niske dužine m nezavisni. Samim tim, moguće je svesti problem male parsimonije na rešavanje problema pojedinačno za svaku kolonu matrice poravnanja. Posmatra se m stabala $T_1, T_2, \ldots, T_m$ u kojima je svaki čvor označen karakterom i iz oznaka unutar stabla T. Tako će grane stabla imati težinu 1 ukoliko spajaju čvorove različitih oznaka ili 0 u suprotnom. Koja bi onda bila ocena parsimonije podstabla T_v u oznaci $s_k(v)$, gde v predstavlja čvor u korenu podstabla, a k simbol u datoj azbuci?

Ukoliko je stablo binarno, tada svaki čvor v ima dva potomka koje ćemo označiti sa *Daughter(v)* i *Son(v)*. Tada skor $s_k(v)$ možemo izračunati na sledeći način:

$$s_k(v) = min_{i \in T}\{s_i(Daughter(v)) + \delta_{i,k}\} + min_{j \in T}\{s_j(Son(v)) + \delta_{j,k}\} \tag{3.3}$$

gde δ predstavlja Kronekerov δ-simbol, čija je vrednost jednaka nuli ukoliko je $i = j$, a u suprotnom jedan.

Samim tim, pronalazak skora parsimonije stabla T se svodi na pronalazak najmanjeg skora parsimonije za svako podstablo $T_1, T_2, \ldots, T_m$, a njihova suma će predstavljati skor parsimonije stabla T.

U nastavku će biti prikazan pseudokod za problem male parsimonije (Algoritam 5). Kao ulaz u algoritam su stablo T, i niz *karakteri* koji predstavlja karaktere koji se nalaze u listovima stabla. U svakoj iteraciji bira se čvor v i računa $s_k(v)$ za svaki simbol k u azbuci. Za svaki čvor ćemo imati informaciju da li je obrađen ili nije u nizu *oznacen*, tako u slučaju da je obrađen čvor v, *oznacen(v)* će biti jednako 1, a u suprotnom 0. Unutrašnji čvor u stablu T nazivamo *zrelim* ako mu je vrednost *oznacen* jednak 0, a *oznacen* od potomaka jednak 1. Algoritam obrađuje čvorove od listova prema gore tako što pronalazi zreli čvor v za koji je moguće izračunati $s_k(v)$. U nastavku sledi primer primene algoritma za problem male parsimonije.

Primer primene algoritma male parsimonije

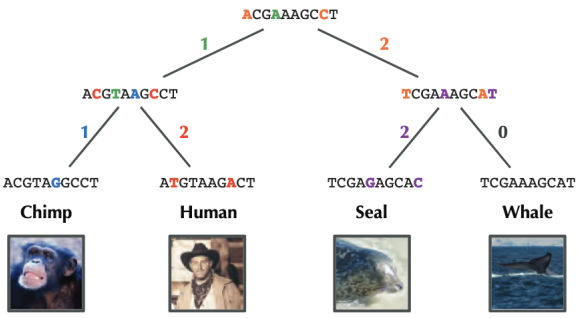
Pretpostavimo da imamo matricu karakteristika D^k:

$$\begin{bmatrix} Majmun & ACGTAGGCCT \\ Covek & ATGTAAGACT \\ Foka & TCGAGAGCAC \\ Kit & TCGAAAGCAT \end{bmatrix}$$

Algoritam 5 Pseudokod za problem male parsimonije

```
1: procedure MalaParsimonija(T, karakter)
2:     for svaki čvor v u stablu T do
3:         oznacen(v) ← 0
4:         if v je list then
5:             oznacen(v) ← 1
6:             for svaki simbol k u azbuci do
7:                 if karakter(v) = k then
8:                     s_k(v) ← 0
9:                 else
10:                    s_k(v) ← ∞
11:    while postoji zreo čvor u stablu T do
12:        v ← zreo čvor u T
13:        oznacen(v) ← 1
14:        for svaki simbol k u azbuci do
15:            s_k(v) ← min_{i∈T}{ s_i(Daughter(v)) + δ_{i,k}} + min_{j∈T}{ s_j(Son(v)) + δ_{j,k}}
    return minimalni skor parsimonije za sve simbole k {s_k(v)}
```

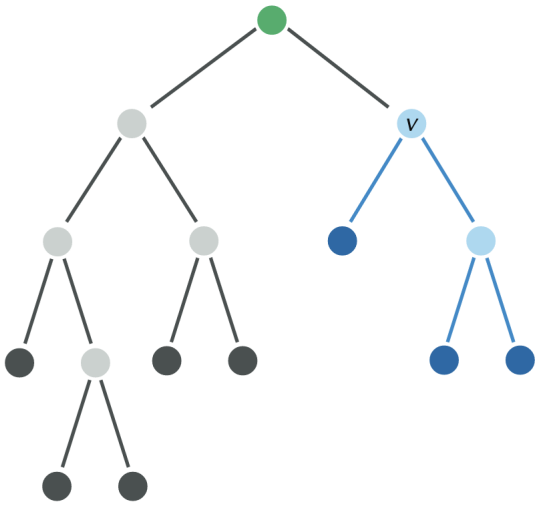
Za matricu karakteristika D^k pretpostavimo da nam je poznata topologija stabla T za četiri vrste koje posmatramo. Treba dodeliti sekvence predaka unutrašnjim čvorovima stabla tako da je suma Hamingovih rastojanja duž svake grane najmanja, odnosno treba dodeliti sekvence unutrašnjim čvorovima stabla tako da je skor parsimonije najmanji. Na primeru 3.14 možemo videti da je skor parsimonije 8.



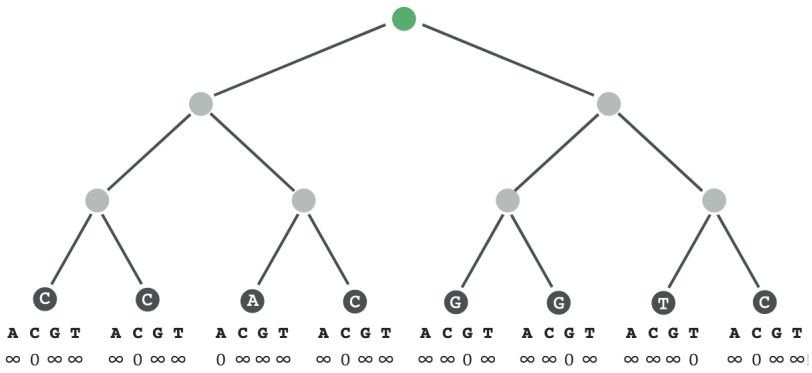
Slika 3.14: Prikaz topologije stabla za gore navedenu matricu karakteristika [6]

Kao što je već pomenuto, posmatraćemo podstablo T_v stabla T gde koren podstabla predstavlja čvor v kao što je prikazano na slici 3.15.

Neka je k simbol iz alfabeta i v čvor u stablu T. Skor parsimonije $s_k(v)$ listova je jednak beskonačnosti u slučaju da je simbol različit od simbola u čvoru, 1 u suprotnom. Podstablo će izgledati kao što je prikazano na slici 3.16.



Slika 3.15: Prikaz (plavog) podstabla T_v čvora v u okviru stabla T [6]



Slika 3.16: Inicijalizacija $s_k(v)$ za sve listove v [6]

Dalje računamo minimalni skor parsimonije za unutrašnje čvorove koristeći formulu 3.3. Prikazaćemo kako se računa skor za čvorove na nivou 2[1], slično će se računati za ostale nivoe, pa će biti samo prikazana rekonstrukcija stabla na slikama redom 3.17, 3.18, 3.19.

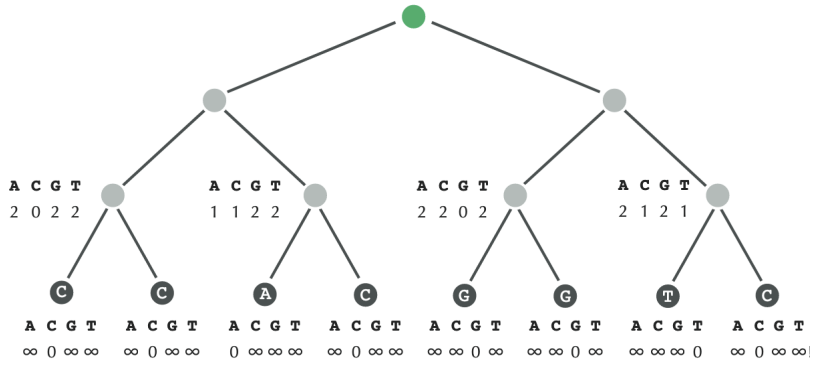
[1]Nivo 2 u odnosu na koren, gde je koreni čvor na nivou 0.

$$
\begin{aligned}
s_k(A_2) &= min(\infty + 0,\ 0 + 1,\ \infty + 1,\ \infty + 1) \\
&\quad + min(\infty + 0,\ 0 + 1,\ \infty + 1,\ \infty + 1) = 1 + 1 = 2 \\
s_k(C_2) &= min(\infty + 1,\ 0 + 0,\ \infty + 1,\ \infty + 1) \\
&\quad + min(\infty + 1,\ 0 + 0,\ \infty + 1,\ \infty + 1) = 0 + 0 = 0 \\
s_k(G_2) &= min(\infty + 1,\ 0 + 1,\ \infty + 0,\ \infty + 1) \\
&\quad + min(\infty + 1,\ 0 + 1,\ \infty + 0,\ \infty + 1) = 1 + 1 = 2 \\
s_k(T_2) &= min(\infty + 1,\ 0 + 1,\ \infty + 1,\ \infty + 0) \\
&\quad + min(\infty + 1,\ 0 + 1,\ \infty + 1,\ \infty + 0) = 1 + 1 = 2
\end{aligned}
$$

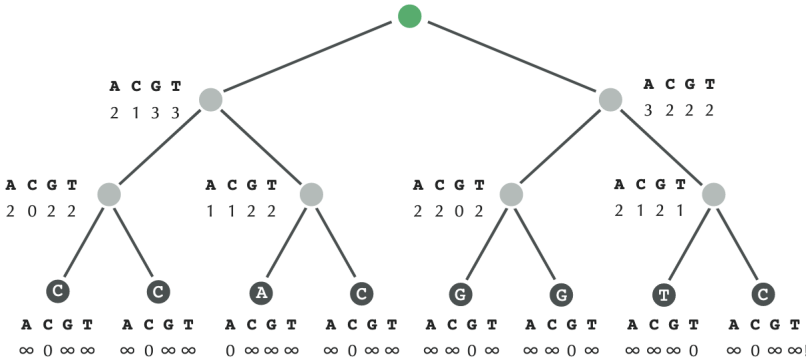
$$
\begin{aligned}
s_k(A_2) &= min(0 + 0,\ \infty + 1,\ \infty + 1,\ \infty + 1) \\
&\quad + min(\infty + 0,\ 0 + 1,\ \infty + 1,\ \infty + 1) = 0 + 1 = 1 \\
s_k(C_2) &= min(0 + 1,\ \infty + 0,\ \infty + 1,\ \infty + 1) \\
&\quad + min(\infty + 1,\ 0 + 0,\ \infty + 1,\ \infty + 1) = 0 + 0 = 1 \\
s_k(G_2) &= min(0 + 1,\ \infty + 1,\ \infty + 0,\ \infty + 1) \\
&\quad + min(\infty + 1,\ 0 + 1,\ \infty + 0,\ \infty + 1) = 1 + 1 = 2 \\
s_k(T_2) &= min(0 + 1,\ \infty + 1,\ \infty + 1,\ \infty + 0) \\
&\quad + min(\infty + 1,\ 0 + 1,\ \infty + 1,\ \infty + 0) = 1 + 1 = 2
\end{aligned}
$$

$$
\begin{aligned}
s_k(A_2) &= min(\infty + 0,\ \infty + 1,\ 0 + 1,\ \infty + 1) \\
&\quad + min(\infty + 0,\ \infty + 1,\ 0 + 1,\ \infty + 1) = 0 + 1 = 2 \\
s_k(C_2) &= min(\infty + 1,\ \infty + 0,\ 0 + 1,\ \infty + 1) \\
&\quad + min(\infty + 1,\ \infty + 0,\ 0 + 1,\ \infty + 1) = 0 + 0 = 2 \\
s_k(G_2) &= min(\infty + 1,\ \infty + 1,\ 0 + 0,\ \infty + 1) \\
&\quad + min(\infty + 1,\ \infty + 1,\ 0 + 0,\ \infty + 1) = 1 + 1 = 0 \\
s_k(T_2) &= min(\infty + 1,\ \infty + 1,\ 0 + 1,\ \infty + 0) \\
&\quad + min(\infty + 1,\ \infty + 1,\ 0 + 1,\ \infty + 0) = 1 + 1 = 2
\end{aligned}
$$

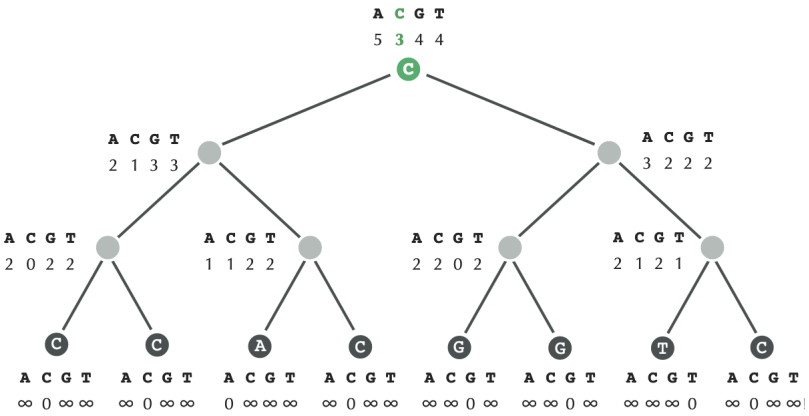
$$
\begin{aligned}
s_k(A_2) &= min(\infty + 0,\ \infty + 1,\ \infty + 1,\ 0 + 1) \\
&\quad + min(\infty + 0,\ 0 + 1,\ \infty + 1,\ \infty + 1) = 0 + 1 = 2 \\
s_k(C_2) &= min(\infty + 1,\ \infty + 0,\ \infty + 1,\ 0 + 1) \\
&\quad + min(\infty + 1,\ 0 + 0,\ \infty + 1,\ \infty + 1) = 0 + 0 = 1 \\
s_k(G_2) &= min(\infty + 1,\ \infty + 1,\ \infty + 0,\ 0 + 1) \\
&\quad + min(\infty + 1,\ 0 + 1,\ \infty + 0,\ \infty + 1) = 1 + 1 = 2 \\
s_k(T_2) &= min(\infty + 1,\ \infty + 1,\ \infty + 1,\ 0 + 0) \\
&\quad + min(\infty + 1,\ 0 + 1,\ \infty + 1,\ \infty + 0) = 1 + 1 = 1
\end{aligned}
$$



Slika 3.17: Inicijalizacija $s_k(v)$ unutrašnjih čvorova na nivou 2 [6]



Slika 3.18: Inicijalizacija $s_k(v)$ unutrašnjih čvorova na nivou 3 [6]



Slika 3.19: Inicijalizacija korenog čvora [6]

Glava 4

Aplikacija PhyTReeV

U narednom poglavlju dat je opis tehnologije koja je korišćena za izradu aplikacije *PhyTreeV (Phylogenetic Tree Reconstruction and Visualisation)* kao i način na koji su prethodno opisani algoritmi implementirani.

Generator filogenetskih stabala predstavlja veb aplikaciju koja se sastoji iz serverskog i klijentskog dela. Izbor klijent-server arhitekture je bio uslovljen potrebom korišćenja širokog spektra alata koji pruža programski jezik Pajton (engl. *Python*), čime se omogućava lakša implementacija algoritama na strani servera. Na strani klijenta korišćen je programski jezik Javaskript (engl. *Javascript*) koji omogućava lako baratanje DOM-om (engl. *Domain Object Model*) čime je bitno pojednostavljen proces kreiranja i menjanja stabla.

Jedan od glavnih ciljeva ovog rada je kreiranje aplikacije koja omogućava prikaz koraka u izvršavanju izabranog algoritma i samim tim olakšava njihovo razumevanje. Aplikacija je javno dostupna na *github*-u i može se naći na sledećem linku: https://github.com/i-djurdjevic/phytreev

4.1 Klijent

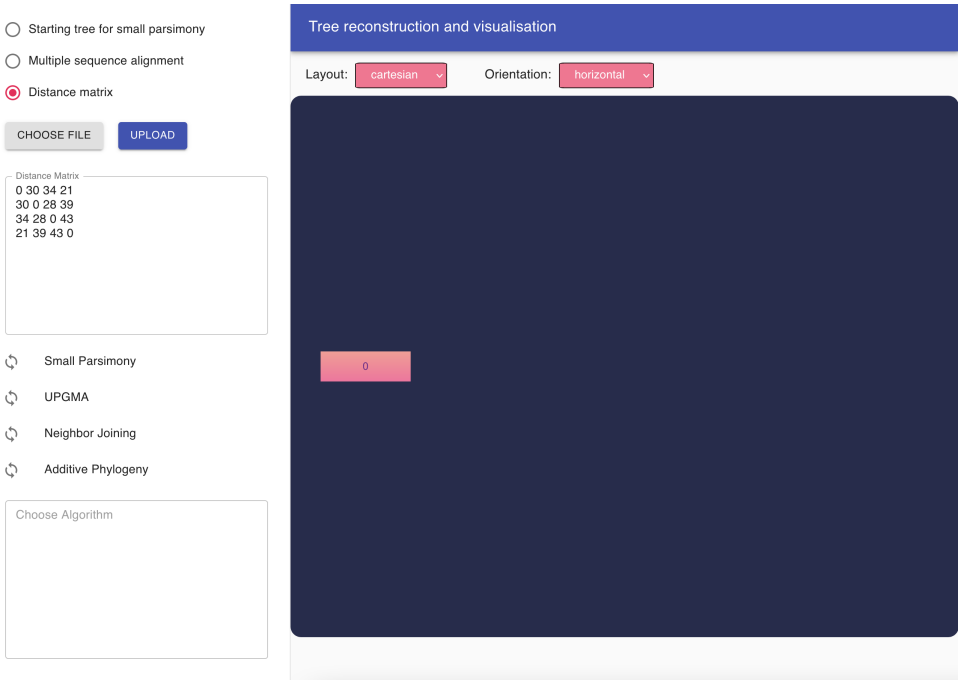
Klijentska strana aplikacije je pisana u programskom jeziku *JavaScript*. Za kreiranje korisničkog interfejsa korišćena je *React* biblioteka, pre svega jer omogućava interaktivan rad sa filogenetskim stablima i izmenu korisničkog interfejsa bez potrebe za renderovanjem celog *DOM*-a, već samo delova kod kojih je došlo do promene stanja. *React* omogućava pisanje *HTML* elemenata u *JavaScript*-u i postavlja ih u *DOM*. *JSX* prebacuje *HTML* tagove u *React* elemente.

Za većinu *HTML* elemenata u aplikaciji (dugmići, tekstualna polja, navigacioni meni) korišćena je *Google*-ova *MaterialUI* biblioteka koja ima *out-of-the-box* stilove. Za posebna stilizovanja korišćen je *CSS*. Za vizualizaciju filogenetskih stabala korišćena je *D3* biblioteka, kao i *visx Airbnb* biblioteka.

Kod unosa na klijenskoj strani zahteva se najpre odabir tipa ulaznih podataka koji će biti prosleđeni serveru. Moguće opcije su početno stablo za malu parsimoniju (*Starting tree for small parsimony*), višestruko poravnanje (*Multiple sequence alignment*) i matrica rastojanja (*Distance matrix*). Zatim, korisnik može da izabere na koji način želi da unese odabrani tip podataka. Moguće je učitati datoteku odabirom na dugme *Choose file*, gde će se nakon učitane datoteke promeniti tekst na dugmetu, i to nazivom učitane datoteke. Nakon pritiska na dugme *Upload* u tekstualnom polju će se prikazati učitani podaci nakon čega korisnik treba da odabere koji algoritam želi

da se izvrši.

Prikaz korisničkog interfejsa se može videti na slici 4.1.



Slika 4.1: Prikaz korisničkog interfejsa

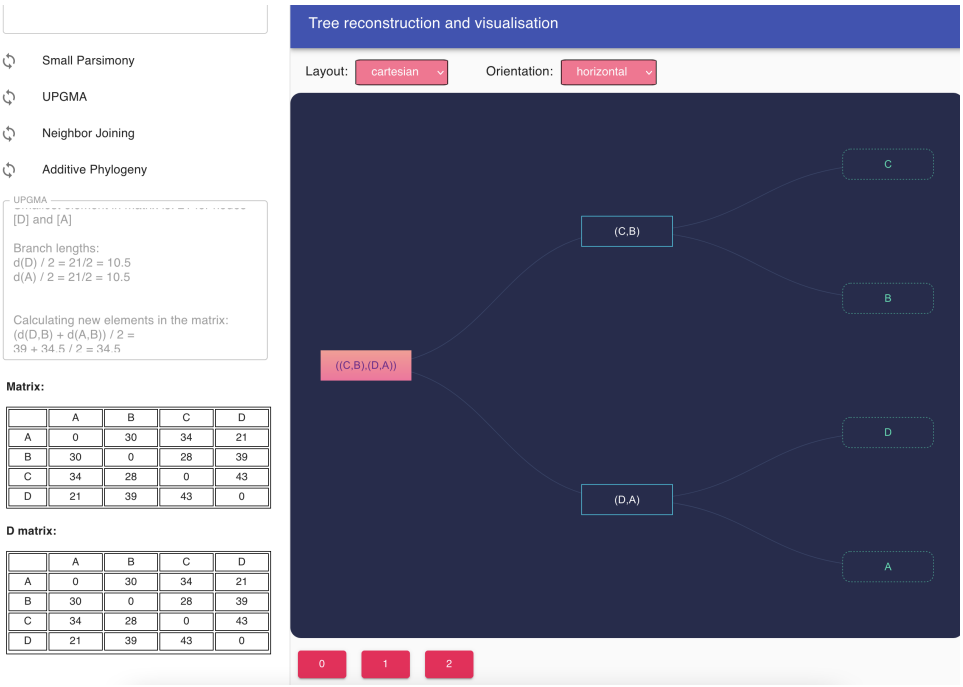
Nakon što server obradi podatke, rezultati se vraćaju klijentu uz prikaz rekonstruisanog filogenetskog stabla. U tekstualnom polju se prikazuje prvi korak postupka za odabrani algoritam, a ispod tekstualnog polja matrica rastojanja. Ispod polja u kojem se nalazi stablo, prikazuju se dugmići koji označavaju redne brojeve koraka algoritma. Odabirom koraka menja se sadržaj u tekstualnom polju, kao i matrice ispod njega u zavisnosti od izabranog algoritma.

Ukoliko je izabran *UPGMA* algoritam, ispod unete matrice pojaviće se još jedna matrica koja će se menjati pri svakom koraku. Na slici 4.2 prikazan je prvi korak algoritma.

Ukoliko je izabran *Neighbor-Joining* algoritam, ispod unete matrice pojaviće se još dve matrice, D i D^*. Na slici 4.3 prikazan je prvi korak algoritma.

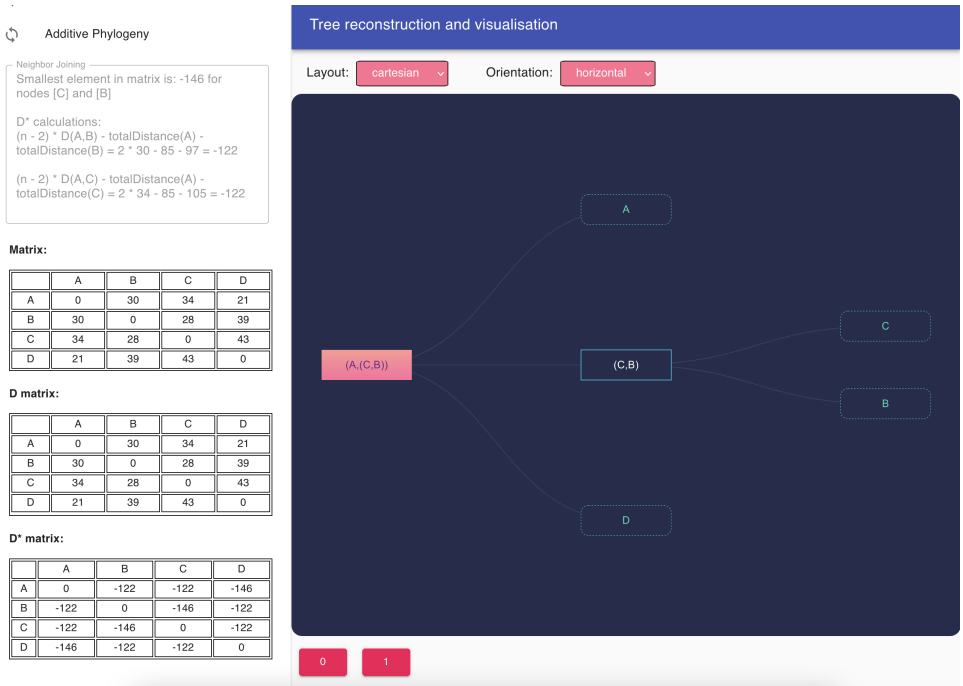
Ukoliko je izabran algoritam aditivne filogenije, ispod unete matrice pojaviće se još dve matrice, D_{Bald} i D_{Trim}. Na slici 4.4 prikazan je prvi korak algoritma.

Ukoliko je izabran algoritam male parsimonije, potrebno je uneti drugačije podatke u poređenju sa prethodnim algoritmima. Treba izabrati opciju *"Starting tree for small parsimony"* i u tekstualno polje uneti topologiju stabla u čijim listovima će se nalaziti genomske sekvence. Na slici 4.5 prikazan je prvi korak algoritma. Prikazani su koraci samo za prvi karakter genomske sekvence, jer bi u suprotnom za dugačke sekvence bio ogroman broj koraka.



	A	B	C	D
A	0	30	34	21
B	30	0	28	39
C	34	28	0	43
D	21	39	43	0

Slika 4.2: Prikaz *UPGMA* algoritma

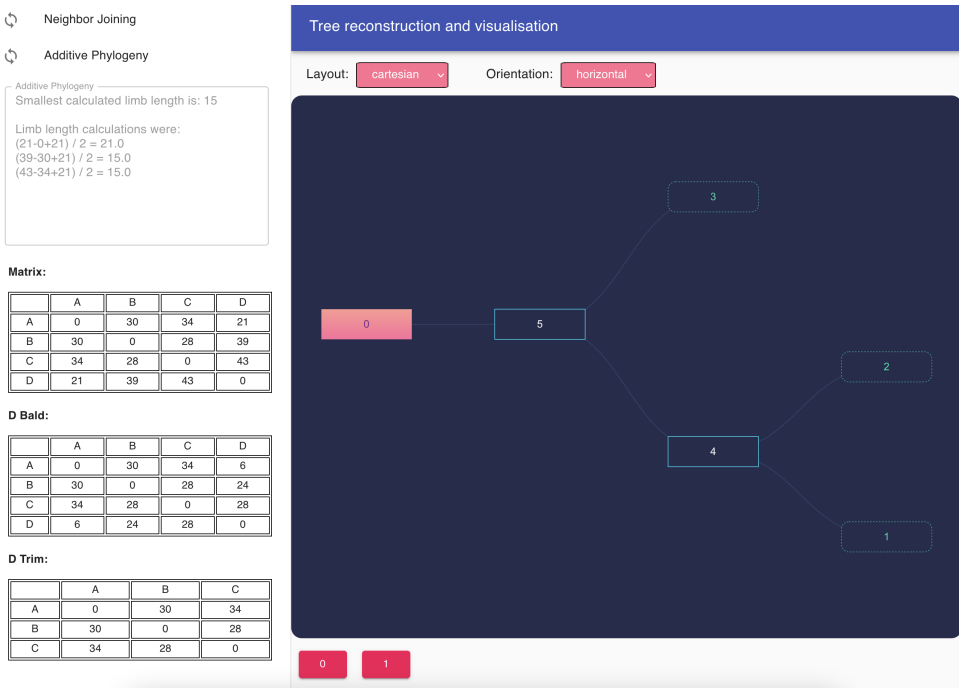


	A	B	C	D
A	0	30	34	21
B	30	0	28	39
C	34	28	0	43
D	21	39	43	0

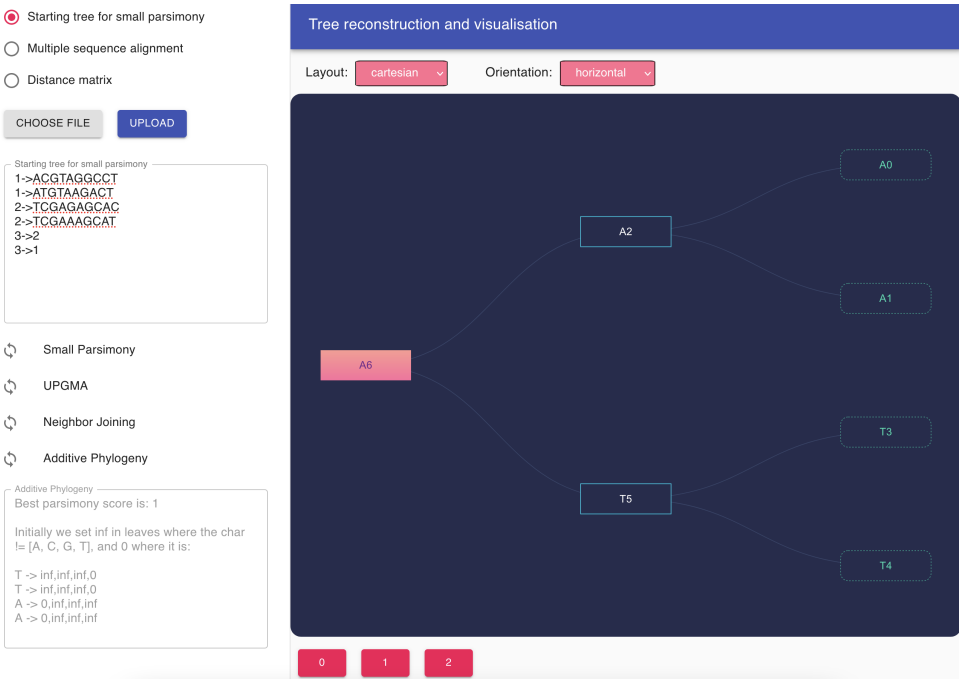
	A	B	C	D
A	0	-122	-122	-146
B	-122	0	-146	-122
C	-122	-146	0	-122
D	-146	-122	-122	0

Slika 4.3: Prikaz *NJ* algoritma

Ukoliko je unet pogrešan podatak, ili je došlo do greške bilo kog tipa, početna strana se resetuje na početno stanje (slika 4.1). Stablo je moguće prikazati na različite načine. Moguće je izabrati drugačiju orijentaciju i smer prikazivanja čvorova. Orijentacija (*Orientation*) može da bude vertikalna (*Vertical*) ili horizontalna (*Horizontal*), dok koordinatni sistem u kom su prikazani čvorovi (*Layout*) može biti polaran (*Polar*) ili Dekartov (*Cartesian*). Prikaz ovih konfiguracija je dat na slikama 4.6, 4.7, 4.8.



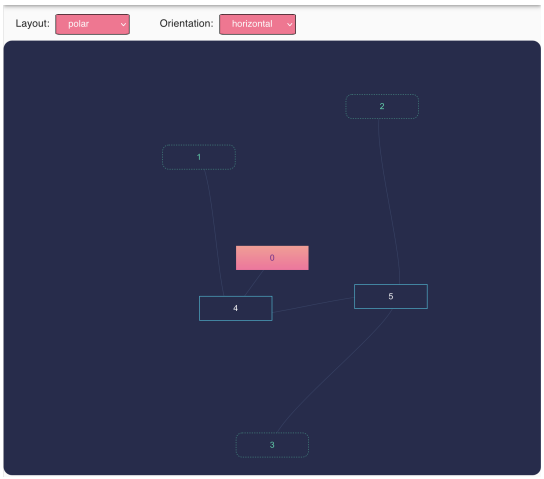
Slika 4.4: Prikaz *AdditivePhylogeny* algoritma



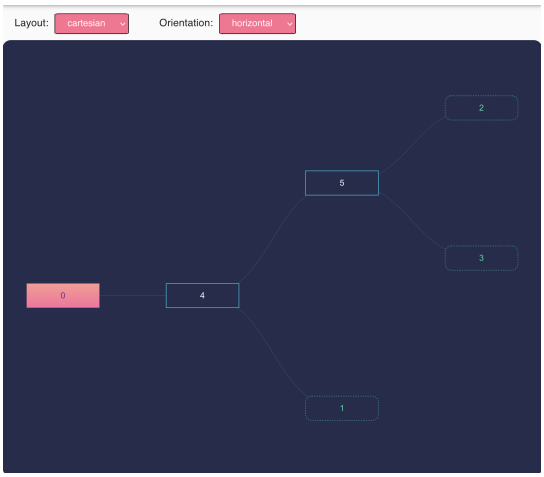
Slika 4.5: Prikaz *SmallParsimony* algoritma

4.1.1 Implementacija klijenta

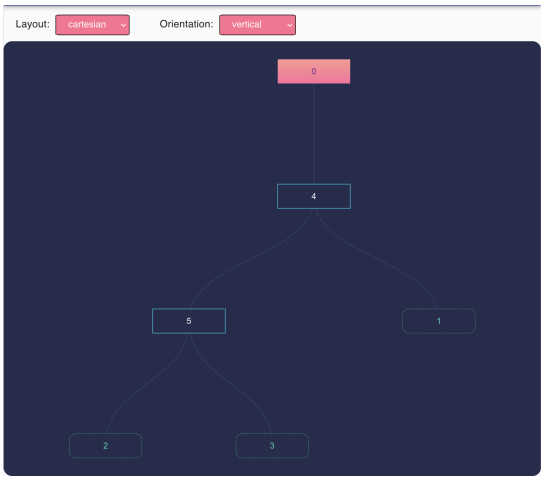
Ceo kod klijenta sastoji se iz narednih datoteka:



Slika 4.6: Prikaz čvorova u polarnom koordinatnom sistemu



Slika 4.7: Prikaz čvorova u Dekartovom koordinatnom sistemu sa horizontalnom orijentacijom



Slika 4.8: Prikaz vertikalne orijentacije stabla

- **index.js** - Glavna datoteka koja od korisnika zahteva unos i izbor matrice, višestrukog poravnanja ili početnog stabla. U njoj se nalazi većina *HTML* koda iz aplikacije.
- **fetchUtils.js** - U ovoj datoteci se nalazi implementacija metoda za *HTTP* zahteve.
- **algorithmsService.js** - Datoteka u kojoj se nalazi implementacija metode koja postavlja izabrani algoritam i poziva odgovarajuću metodu iz *fetchUtils.js* za slanje *HTTP* zahteva.
- **utils/stepsParser.js** - Datoteka u kojoj se nalaze implementacije metoda za parsiranje dobijenih rezultata sa servera.
- **utils/treeNewickParser.js** - U ovoj datoteci se nalazi implementacije metoda za parsiranje dobijenog stabla sa servera u *newick* formatu u format prikladan za prikazivanje stabla.
- **tree/Tree.js**, **tree/Node.js**, **tree/NodesMove.js**, **tree/Links.js**, **tree/LinksMove.js** - U ovim datotekama se nalaze implementacija stabla, čvorova kao i metoda za razvijanje stabla.

Za komunikaciju sa serverom koristi se *Fetch API*. Podaci se šalju *POST* zahtevom u čijem telu zahteva se nalaze podaci potrebni za izvršavanje algoritma.

```
function getOptions(data, type) {
  let requestData = {"data": data, "type": type};
  return {
    method: 'POST',
    body: JSON.stringify(requestData),
    headers: {
      'Content-Type': 'application/json'
    }
  };
}

export async function postData(url, data, type) {
  const options = getOptions(data, type);
  const response = await fetch(url, options);
  return await response.json();
}
```

Nakon *HTTP* poziva čeka se asihrono na odgovor sa servera. Nakon što su primljeni podaci, moraju se pozvati određene metode za parsiranje rezultata u zavisnosti od izabranog algoritma. Takođe, postavljaju se vrednosti u poljima korišćenjem *React State*-a.

```
const [matrix, setMatrix] = React.useState(initialMatrix);
```

Izborom odgovarajućeg algoritma, poziva se metoda `onClickAlgorithm(inputData, alg)` koja poziva metodu za pravljenje *HTTP* zahteva i čeka na odgovor.

```
function onClickAlgorithm(inputData, alg) {
    setStepTable1(null);
    if (alg === "Neighbor Joining") {
      applyAlgorithm("nj", inputData, radio)
      .then((result) => {
            if (isMountedRef.current) {
              setAlgorithm("nj");
              setResult(result);
              setNumberOfSteps(result.steps);
```

```
10              setStepsTextFieldLabel('Neighbor Joining');
11              setStepsValue(parseNjSteps(result, 0));
12              setStepTable2(parseNjMatrix(result, result.dStarMatrices,
     0));
13              setStepTable1(parseTable(result, 0));
14              setStepTable0(parseArrayTable(parseInitialMatrix(
    inputMatrix, result),
15                  "Matrix:"));
16              setTreeData(treeNewickParser(result.tree));
17            } else {
18              console.log("not mounted");
19            }
20          }
21      )
22      .catch(() => {
23        resetAllValues();
24      });
25    }
```

Za svaki algoritam postoje metode za parsiranje koraka, matrica i pretvaranje stabla u *newick*[1] formatu u format pogodan za prikazivanje. Ukoliko dođe do greške, poziva se metoda `resetAllValues` koja vraća stanje aplikacije u početno stanje.

4.2 Server

Sa serverske strane korišćen je programski jezik Pajton (engl. *Python*) za implementaciju algoritama. Dodatne biblioteke koje su uključene uz osnovne funkcija su `numpy, sys, json, pandas i os`. Pored toga korišćene su neke metode iz paketa `Bio` za konstruisanje matrice rastojanja na osnovu višestrukih poravnanja. Za uspostavljanje veze između klijenta i servera odabran je razvojni okvir *Flask* koji predstavlja jednu od najpoznatijih veb razvojnih okvira za ovaj jezik, prvenstveno zbog svoje jednostavnosti. Komunikacija sa klijentskom stranom se vrši preko *REST API*-ja[2].

4.2.1 API - Programski interfejs aplikacije

U aplikaciji je korišćen *REST API* koji definiše pravila pomoću kojih je moguće dobiti informacije putem *URL*-a. Detalji programskog interfejsa aplikacije su prikazani u tabelama 4.1 i 4.2.

Različiti algoritmi obrađuju različite tipove podataka, pa je tako za *Neighbor-Joining*, aditivnu filogeniju i *UPGMA* algoritme moguće proslediti matricu rastojanja ili višestruko poravnanje genoma. Za malu parsimoniju očekivan ulaz je inicijalno stablo, odnosno struktura stabla u čijim listovima se nalaze genomske sekvence za koje treba izračunati najbolji skor i sekvence koje bi se nalazile u unutrašnjim čvorovima stabla. Potom se primenjuje izabrani algoritam i kao rezultat klijentu se vraća objekat sa pojedinačnim koracima algoritma, kao i rezultujuće stablo u *newick* formatu.

Korišćenje aplikacije obuhvata sledeće korake:

1. Izbor tipa podatka koji se šalje serveru. Moguće je odabrati početno stablo za malu parsimoniju (engl. *Starting tree for small parsimony*), višestruko poravnanje (engl. *Multiple sequence alignment*) ili matricu rastojanja (engl. *Distance*

[1] Format za prikazivanje stabla i grafova koristeći zagrade i zareze. Primer stabla napisanog u *newick* formatu: (A,B,(C,D));

[2] REST API predstavlja interfejs za komunikaciju sa serverom.

matrix). Ovo polje prvenstveno omogućava da se na serverskoj strani izvrši provera ispravnosti unosa očekivanog tipa podataka.

2. Unos podatka u tekstualno polje ili izbor datoteke u kojima se nalaze podaci.
3. Izbor jednog od raspoloživih algoritama: *UPGMA*, *Neighbor-Joining*, *Small Parsimony* ili *Additive Phylogeny*.
4. Nakon što server vrati rezultat, prikazaće se rezultujuće filogenetsko stablo, kao i dugmići ispod njega koji označavaju korake.
5. Klikom na broj koji određuje korak u tekstualnom polju se pojavljuje objašnjenje procesa u tom koraku algoritma.

U deklaraciji API-ja kao unos za gore navedene algoritme navodi se samo matrica rastojanja.

Http metoda	URL zahteva	Opis
POST	/nj	Nad prosleđenom matricom se primenjuje *NeighborJoining* algoritam i kao rezultat vraća *json* objekat koji u sebi sadrži: rezultujuće stablo predstavljeno u *newick* formatu, listu matrica D', listu D^* matrica i listu dužina spoljnih grana *LimbLength* za svaki korak algoritma.
POST	/upgma	Nad prosleđenom matricom se primenjuje *UPGMA* algoritam i kao rezultat vraća *json* objekat koji u sebi sadrži: rezultujuće stablo predstavljeno u *newick* formatu, listu izabranih čvorova i najmanje vrednosti između njih *steps*, listu *matrices* matrica i listu labela matrica za svaki korak algoritma.
POST	/additive_phylogeny	Nad prosleđenom matricom se primenjuje algoritam *aditivne filogenije* i kao rezultat vraća *json* objekat koji u sebi sadrži: rezultujuće stablo i listu objekata koje sadrže čvor, dužinu spoljnih grana, ogoljenu matricu *dBald* i potkresanu matricu *dTrim* za svaki korak algoritma.
POST	/small_parsimony	Nad prosleđenim inicijalnim stablom se primenjuje algoritam *male parsimonije* i kao rezultat vraća *json* objekat koji u sebi sadrži: rezultujuće stablo, najbolji skor parsimonije i korake.
POST	/upload_file	Prosleđuje se datoteka iz koje treba da se učita matrica rastojanja, višestruko poravnanje ili inicijalno filogenetsko stablo.

Tabela 4.1: Prikaz API deklaracije

Naziv	Tip	Opis
nj	json	Ispravne vrednosti su $N \times N$ matrica rastojanja ili N višestrukih poravnanja
upgma	json	Ispravne vrednosti su $N \times N$ matrica rastojanja ili N višestrukih poravnanja
additive_phylogeny	json	Ispravne vrednosti su $N \times N$ matrica rastojanja ili N višestrukih poravnanja
small_parsimony	json	Ispravna vrednost je usmereno stablo u čijim listovima se nalazi sekvenca genoma. U svakom redu se nalaze dva čvora u obliku: *čvor1 -> čvor2* tako da je *čvor1* roditeljski čvor čvora *čvor2*
upload_file	form-data	Ispravna vrednost je datoteka u *.csv* ili *.txt* formatu sa sadržajem u skladu sa željenim algoritmom

Tabela 4.2: Prikaz parametara zahteva

4.2.2 Implementacija servera

Implementacija servera je u potpunosti odrađena u programskom jeziku *Python* i organizovana je kroz sledeće datoteke:

- **app.py** - U ovoj datoteci se nalaze sve krajnje tačke API-ja i srž serverske strane programa. U datoteci se nalazi logika primanja podataka, kao i emitovanje povratih informacija.
- **tree_construction/additive_phylogeny.py** - U ovoj datoteci se nalazi implementacija algoritma aditivne filogenije.
- **tree_construction/small_parsimony.py** - U ovoj datoteci se nalazi implementacija algoritma male parsimonije.
- **tree_construction/upgma.py** - U ovoj datoteci se nalazi implementacija *UPGMA* algoritma.
- **tree_construction/nj.py** - U ovoj datoteci se nalazi implementacija *Neighbor-Joining* algoritma.
- **utils/file_utils.py** - Datoteka sadrži metode za učitavanje datoteka i metode za parsiranje podataka iz datoteke.
- **utils/matrix_utils.py** - Datoteka sadrži metode za parsiranje matrica.
- **utils/newick_utils.py** - Datoteka sadrži metode za prebacivanje stabla iz jednog formata, matrice ili mape u *newick* format.
- **utils/request_convert_utils.py** - Datoteka sadrži metode za parsiranje vrednosti iz zahteva u odgovarajući format u zavisnosti od izabranog algoritma.

Za komunikaciju sa klijentom koristi se razvojni okvir *Flask* koji se jednostavno inicijalizuje na sledeći način:

```
app = Flask(__name__)
CORS(app)

if __name__ == '__main__':
    app.run()
```

`CORS` (engl. *Cross-origin resource sharing*) je ekstenzija razvojnog okvira *Flask* koja omogućava preuzimanje resursa sa različitih izvora.

Za svaki algoritam kreira se *POST* zahtev sa različitim *URL*-om u zavisnosti od algoritma, kao što je prikazano u tabeli 4.2. Deklaracija metode u primeru *Neighbor-Joining* algoritma izgleda na sledeći način:

```
@app.route("/nj", methods=['POST'])
def nj():
```

Najpre se navodi ruta, a potom *HTTP* metoda koja je korićena za prosleđivanje zahteva.

Procedura za obrađivanje i vraćanje podataka za sve algoritme je slična, pa će iz tog razloga biti prikazani koraci na primeru *UPGMA* algoritma.

```
@app.route('/upgma', methods=['POST'])
def upgma():
    global filepath
    global hasImportedFile

    distance_matrix, dim, m_labels = rcu.get_matrix_and_labels(request,
     has_imported_file, file_labels, file_data)
    matrices, steps, upgma_tree, branch_lengths,
    branch_length_calculations, new_distance_calculations = upgma(
    distance_matrix)

    matrix_string_array = mx.get_matrices_string_array(matrices)
    has_imported_file = False

    return make_response(json.dumps(
        {"steps": steps, "matrix": upgma_tree.format('newick'), "
    matrices": matrix_string_array, "labels": m_labels,
         "branch_lengths": branch_lengths, "branch_length_calculations"
    : branch_length_calculations,
         "distance_calculations": new_distance_calculations}, default=
    default_json), 200)
```

Za svaki algoritam se najpre obrađuju prosleđeni podaci u zavisnosti od formata u kome su poslati. Proverava se da li je podatak učitan iz datoteke i da li je prosleđena matrica rastojanja ili višestruko poravnanje. Za malu parsimoniju se vrši odvojeno provera da li je u pitanju početno stablo.

```
def get_matrix_and_labels(request, has_imported_file, file_labels,
    file_data):
    request_data = request.json
    if not has_imported_file:
        if request_data["type"] == "distance":
            m, dim = mx.parse_matrix(request_data["data"])
            m_labels = alpha_labels("A", chr(ord('A') + dim - 1))
        elif request_data["type"] == "alignment":
            m, dim, m_labels = get_data_from_alignment(requestData["
    data"])
```

```
    else:
        if request_data["type"] == "alignment":
            m, dim, m_labels = get_data_from_alignment(file_data)
        elif request_data["type"] == "distance":
            m, dim = mx.parse_matrix(file_data)
            m_labels = file_labels
    return m, dim, m_labels
```

Za dobijanje matrice rastojanja na osnovu prosleđenog višestrukog poravnanja korišćena je *Biopython* biblioteka:

```
from Bio import AlignIO
from io import StringIO
from Bio.Phylo.TreeConstruction import DistanceCalculator

def get_data_from_alignment(aln_data):
    handle = StringIO(aln_data)
    aln = AlignIO.read(handle, 'phylip')
    calculator = DistanceCalculator('identity')
    distance_matrix = calculator.get_distance(aln)
    dim = len(distance_matrix)
    # round up matrix values if necessary:
    parsed_matrix = round_up_values_of_matrix(distance_matrix.matrix)
    m = parsed_matrix
    m_labels = distance_matrix.names
    return m, dim, m_labels
```

Nakon obrađivanja podataka koje je uneo korisnik, pozivaju se sami algoritmi. Potom je neophodno parsiranje dobijene matrice u formatu koji je pogodan za prikazivanje na klijentskoj strani, kao i resetovanje globalne promenljive koja označava da li je datoteka prethodno učitana. Rekonstruisano filogenetsko stablo se dalje šalje u *newick* formatu.

4.2.3 Korišćenje aplikacije

Korišćenje aplikacije je jednostavno i intuitivno. Od korisnika se očekuje da obezbedi matricu rastojanja, višestruko poravnanje ili početno filogenetsko stablo. Postoje dva načina unosa podataka:

- Prvi način podrazumeva odabir datoteke. Datoteka može biti u *.txt* ili *.csv* formatu. Podaci iz datoteke će se prikazati u tekstualnom polju.
- Drugi način se ostvaruje unosom vrednosti u tekstualno polje ispod *Upload* dugmeta.

Pre odabira algoritma korisnik treba da označi koji tip podataka je unet (*Starting tree for small parsimony, Multiple sequence alignment, Distance matrix*). Potom korisnik treba da izabere algoritam koji želi da se izvrši nad podacima. Na serveru se onda vrši provera poslatih podataka, kao i tipa ovih podataka. Nakon toga server vraća rezultate koji se prikazuju na klijentu.

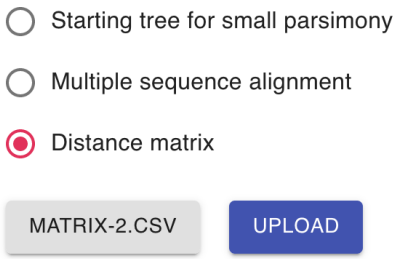
4.2.4 Primer upotrebe

Prvi primer upotrebe - učitavanje podataka iz datoteke

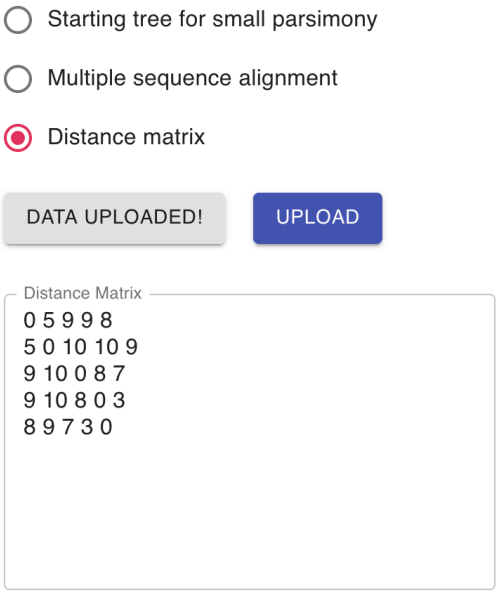
Za prvi primer upotrebe biće prikazan unos matrice rastojanja u *.csv* formatu i primena *UPGMA* algoritma nad njom. Korisnik najpre treba da odabere datoteku

koju želi da učita. Tip podataka može biti označen pre ili posle učitavnja datoteke, ali je bitno da bude odabran pre izvršavanja algoritma. U ovom slučaju biramo tip *"Distance matrix"*. Prikaz dela veb stranice za odabir datotke je prikazano na slici 4.9. Na slici 4.10 je prikazan deo veb stranice nakon učitavanja datoteke.

Nakon učitavanja datoteke, klikom na dugme *UPGMA* šalje se zahtev serveru da izvrši ovaj algoritam. Nakon što server vrati rezultat, veb stranica će izgledati kao što je prikazano na slici 4.11.



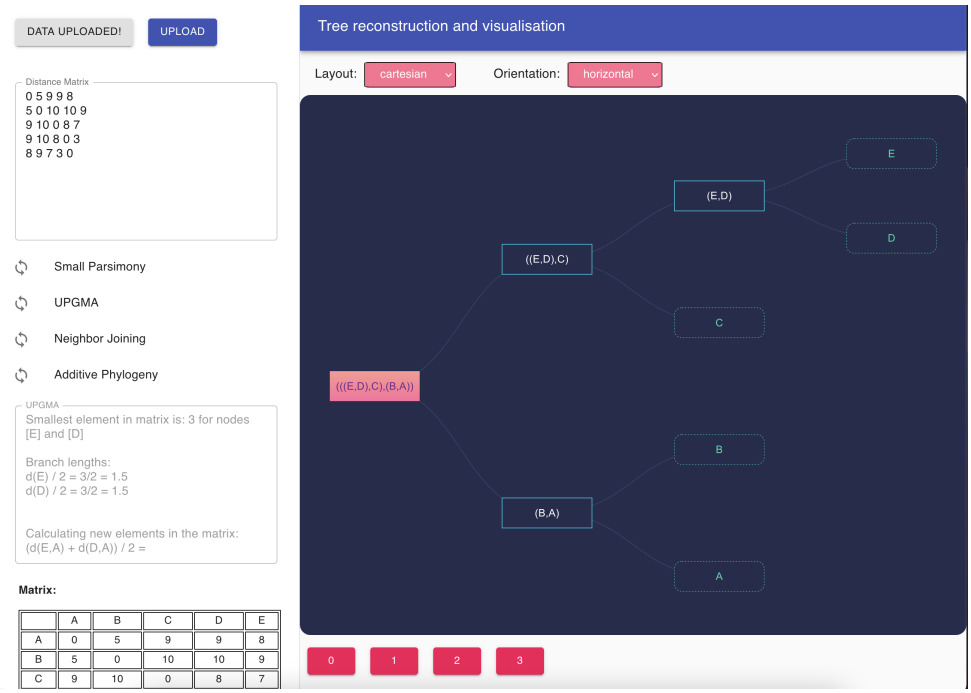
Slika 4.9: Prikaz odabrane datoteke



Slika 4.10: Prikaz učitane datoteke

Drugi primer upotrebe - unošenje podataka u tekstualno polje

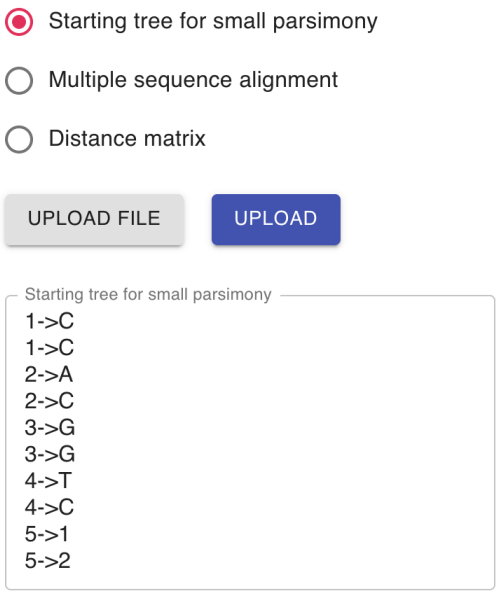
U drugom primeru će biti prikazan unos početnog stabla za malu parsimoniju. Najpre korisnik treba da odabere tip *Starting tree for small parsimony*. Potom je potrebno u svakom redu tekstualnog polja uneti čvorove stabla u obliku *čvor1* → *čvor2*, tako da *čvor1* predstavlja roditeljski čvor čvora *čvor2* Prikaz dela veb stranice sa unosom ovih podataka dat je na slici 4.12. Nakon izvršavanja algoritma veb aplikacija



	A	B	C	D	E
A	0	5	9	9	8
B	5	0	10	10	9
C	9	10	0	8	7

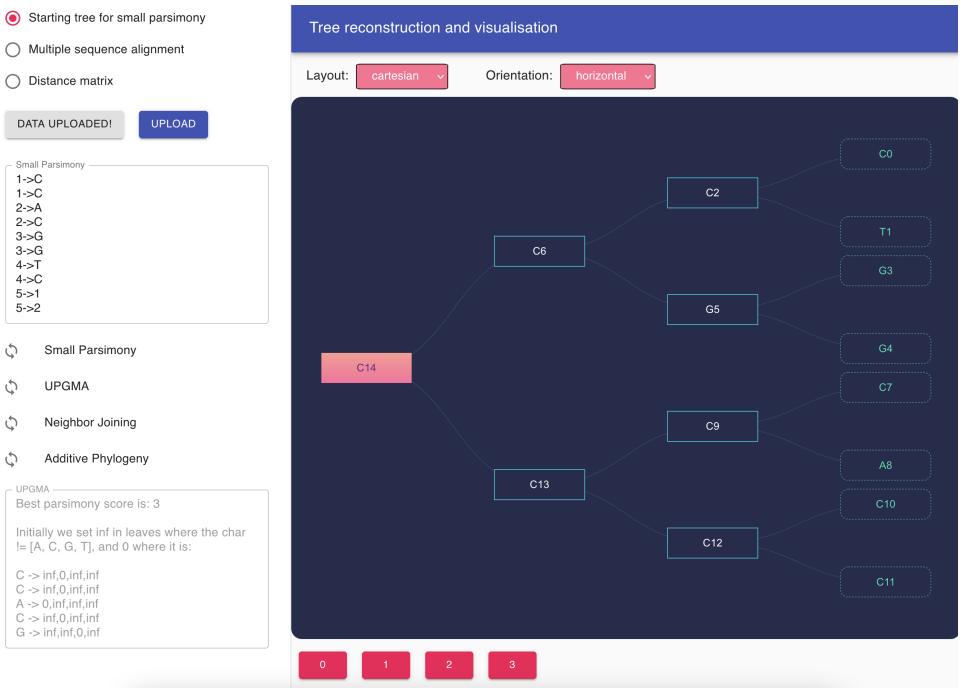
Slika 4.11: Prikaz korisničkog interfejsa nakon izvršavanja *UPGMA* algoritma

će izgledati kao na slici 4.13.



Slika 4.12: Prikaz početnog stabla

Treba napomenuti da su brojevi pored karaktera u čvorovima stabla dodati kako bi se razlikovali *ključevi* u stablu (umesto samo $x \in \{A, C, G, T\}$, biće označeni sa $xy, x \in \{A, C, G, T\}, y \in \{1,...,N\}$). Stablo se generiše na osnovu prosleđene mape, a kod male parsimonije je moguće imati četiri različita karaktera u svakom čvoru

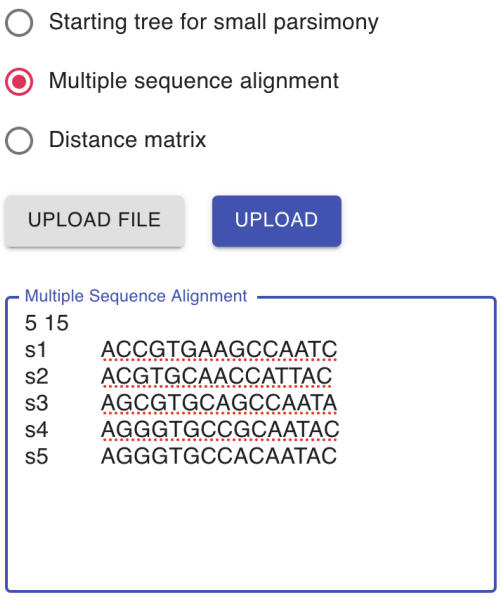


Slika 4.13: Prikaz korisničkog interfejsa nakon izvršavanja *Small Parsimony* algoritma

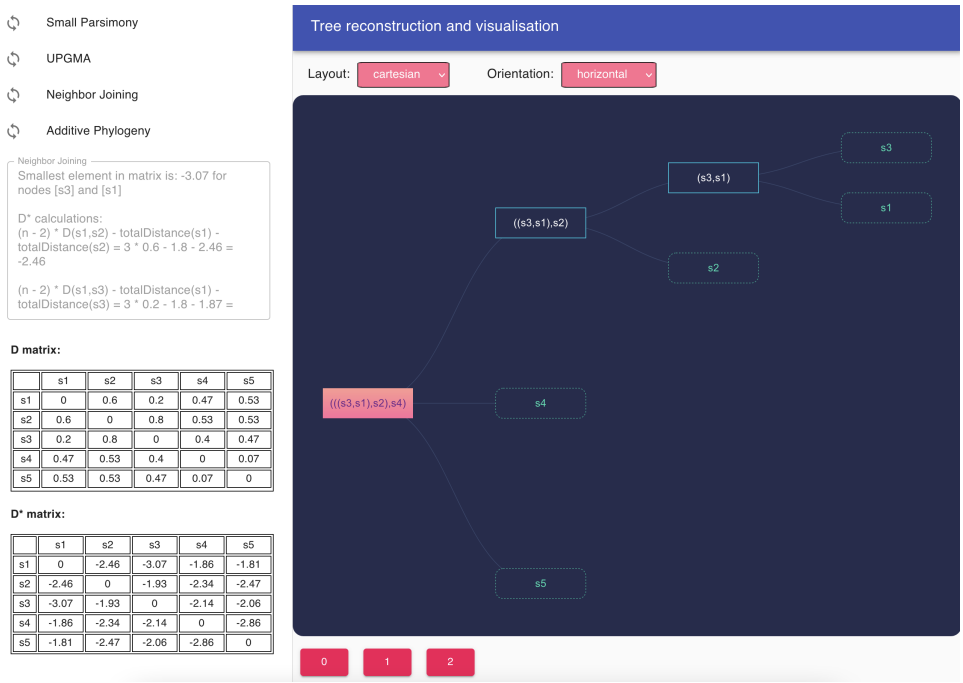
te je samim tim, i vrlo velika mogućnost za ponavljanje istih karaktera u stablu. U suprotnom bi aplikacija izbacila grešku jer ne bi bila u mogućnosti da isparsira mapu sa više istih ključeva.

Treći primer upotrebe - unošenje višestrukog poravnanja u tekstualno polje

Treći primer upotrebe zahteva od korisnika da se unese višestruko poravnanje u tekstualno polje. Treba da izabere opciju *Multiple sequence alignment*. Pravilan upis višestrukog poravnanja podrazumeva prvo navođenje broja sekvenci, a potom dužinu sekvenci. U narednim redovima potrebno je navesti naziv organizma, pa razmakom razdvojeno - genomsku sekvencu. Prikaz dela veb aplikacije nakon unosa višestrukog poravnanja je dat na slici 4.14, nakon čega je na slici 4.15 prikazan rezultat izvršavanja *Neighbor-Joining* algoritma.



Slika 4.14: Prikaz unetog višestrugog poravnanja u tekstualno polje



	s1	s2	s3	s4	s5
s1	0	0.6	0.2	0.47	0.53
s2	0.6	0	0.8	0.53	0.53
s3	0.2	0.8	0	0.4	0.47
s4	0.47	0.53	0.4	0	0.07
s5	0.53	0.53	0.47	0.07	0

	s1	s2	s3	s4	s5
s1	0	-2.46	-3.07	-1.86	-1.81
s2	-2.46	0	-1.93	-2.34	-2.47
s3	-3.07	-1.93	0	-2.14	-2.06
s4	-1.86	-2.34	-2.14	0	-2.86
s5	-1.81	-2.47	-2.06	-2.86	0

Slika 4.15: Prikaz korisničkog interfejsa nakon izvršavanja *Neighbor-Joining* algoritma

Glava 5

Zaključak

Kreirana aplikacija objedinjuje nekoliko najznačanijih algoritama za rekonstrukciju filogenetskih stabala i na veoma jednostavan način, kroz jasan interfejs, pojedinačno daje opis svakog algoritma. Najveći doprinos ovog rada leži u tome što je postavljanjem više različitih algoritama pod okrilje jedne aplikacije pružena mogućnost za lakše razumevanje filogenetske analize, a time i primena u nastavnom procesu, kao i u procesu učenja. Aplikaciju je moguće unaprediti uvođenjem dodatnih algoritama za rekonstrukciju stabala, kao i različitih načina unosa podataka. Dodatni mogući vid unapređenja predstavljalo bi prikazivanje rekonstrukcije stabla kroz svaki korak.

Bibliografija

[1] A.Krogh G. Mitchison R. Durbin S. Eddy. *Biological Sequence Analysis, Probabilistic models of proteins and nucleic acids.*

[2] Richard Kliman. *Encyclopedia of Evolutionary Biology 1st Edition.*

[3] Denver Pearson Education University of Colorado. *Symbiosis: The Benjamin Cummings Custom Laboratory Program for the Biological Sciences.* Moorpark College Biology Department, 2003.

[4] Jonathan Pevsner. *Bioinformatics and functional genomics 3rd edition.* John Wiley & Sons Inc.

[5] Steven A. Wasserman Peter V. Minorsky Jane B. Reece Lisa A. Urry Michael L. Cain. *Campbell Biology.*

[6] Pavel Pevzner Phillip Compeau. *Bioinformatics Algorithms, An Active Learning Approach 2nd Edition.*