Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Urban Soban

Implementacija metode asimetričnega srednjega drevesa za iskanje konsenza filogenetskih dreves

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Curk

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

[1]V dogovorju z mentorjem lahko kandidat diplomsko delo s pripadajočo izvorno kodo izda tudi pod katero izmed alternativnih licenc, ki ponuja določen del pravic vsem: npr. Creative Commons, GNU GPL. V tem primeru na to mesto vstavite opis licence, na primer tekst [**?**].

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Besedilo teme diplomskega dela študent prepiše iz študijskega informacijskega sistema, kamor ga je vnesel mentor. V nekaj stavkih bo opisal, kaj pričakuje od kandidatovega diplomskega dela. Kaj so cilji, kakšne metode uporabiti, morda bo zapisal tudi ključno literaturo.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Urban Soban, z vpisno številko **63100344**, sem avtor diplomskega dela z naslovom:

Implementacija metode asimetričnega srednjega drevesa za iskanje konsenza filogenetskih dreves

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Tomaža Curka
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 11. januarja 2011 Podpis avtorja:

Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da pozabite na celo zahvalo.

Družini.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AMT	asymmetric median tree	asimetrično srednje drevo
MIS	maximum independent set	največja neodvisna množica
UPGMA	unweighted pair group method with arithmetic mean	neuteženo gručanje s pomočjo aritmetične sredine
CDAO	comparative data analysis ontology	ontologija s primerjavo podatkov

Povzetek

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja LaTeX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek.

Ključne besede: konsenz, drevo, filogenetika.

Abstract

This sample document presents an approach to typesetting your BSc thesis using LaTeX. A proper abstract should contain around 100 words which makes this one way too short.

Keywords: consensus, tree, phylogeny.

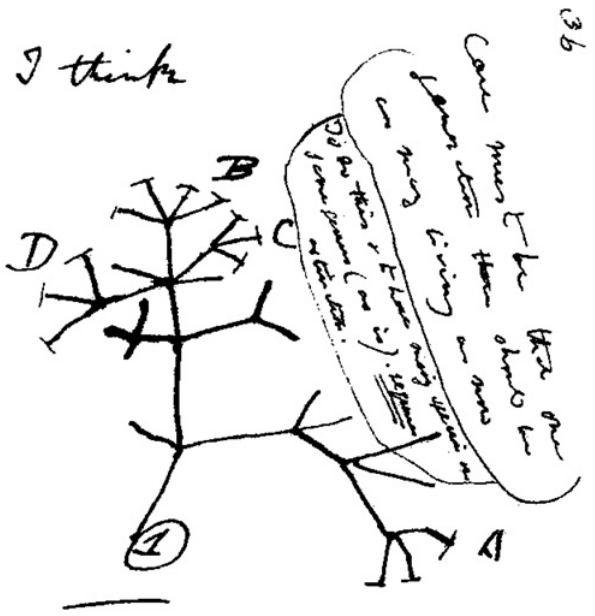
Poglavje 1

Uvod

O evolucijskih razmerjih med živimi ogranizmi se je prvi spraševal Charles Darwin, ko je narisal znamenito "drevo življenja", prikazano na sliki 1.1. Vendar je Darwin lahko organizme razvrščal le glede na njihove morfološke lastnosti. Nato je prišlo do odkritja DNA in izuma računalnika in porodila se je ideja o računski filogenetiki, eni izmed prvih področij bioinformatike, katere cilj je, predvsem iz sekvenc DNA in RNA, odkriti evolucijska razmerja med različnimi taksonomskimi enotami. Mnoge računske filogenetske metode so bile razvite že v prejšnjem stoletju, vendar so nekatere v praktično uporabo prišle šele v zadnjem času zaradi eksponentnega povečanja računske moči. Ker različne metode ali pa celo ena sama metoda lahko proizvede več različnih filogenetskih dreves, mnogokrat želimo rezultate kombinirati v eno drevo in tako pridobiti eno teorijo o evolucijski zgodovini taksonomskih enot.

Sem na pomoč priskočijo konsenzne metode, ki na podlagi različnih kriterijev dele vhodnih dreves v končnem sestavljenem drevesu kombinirajo, ohranijo, ali zavržejo, s ciljem sestaviti drevo, katero kar se da dobro povzema informacije o evolucijski zgodovini, ki jih nosijo vhodna drevesa. Tekom konstrukcije konsenznega drevesa lahko metoda izračuna več različnih dreves, izbrano pa je tisto, ki uživa največjo podporo vhodnih dreves.

V prvem delu diplomske naloge se bomo na kratko seznanili z najbolj uporabljenimi metodami računanja filogenetskih dreves, katerih produkt je



Slika 1.1: "Drevo življenja", prva skica evolucijske zgodovine, narisana s strani Charlesa Darwina leta 1837 [1].

nato vhod v algoritem za izračun konsenza, in nato obravnavali nekatere priljubljene konsenzne metode, kot so striktni konsenz, večinski konsenz in srednji konsenz (konsenz mediane). V drugem delu naloge se bomo osredotočili na novejšo konsenzno metodo, algoritem asimetričnega srednjega drevesa.

Spoznali teoretično ozadje konstrukcije asimetričnega srednjega drevesa, nato pa bo sledila predstavitev programskega paketa Biopython, v katerega smo vključili implementacijo asimetričnega srednjega drevesa, in ostalih orodij, ki smo jih pri tem uporabili. Uspešnost implementirane metode bomo eksperimentalno ocenili na treh vhodnih množicah, ki izražajo različne lastnosti, in na eni realni vhodni množici. Rezultate bomo primerjali s tremi popularnimi konsenznimi metodami glede na razrešenost končnega drevesa in glede na Robinson-Fouldsovo metriko. Za konec bomo preverili, za kakšno število dreves v vhodni množici je uporaba implementirane metode še dovolj hitra in predlagali možne izboljšave.

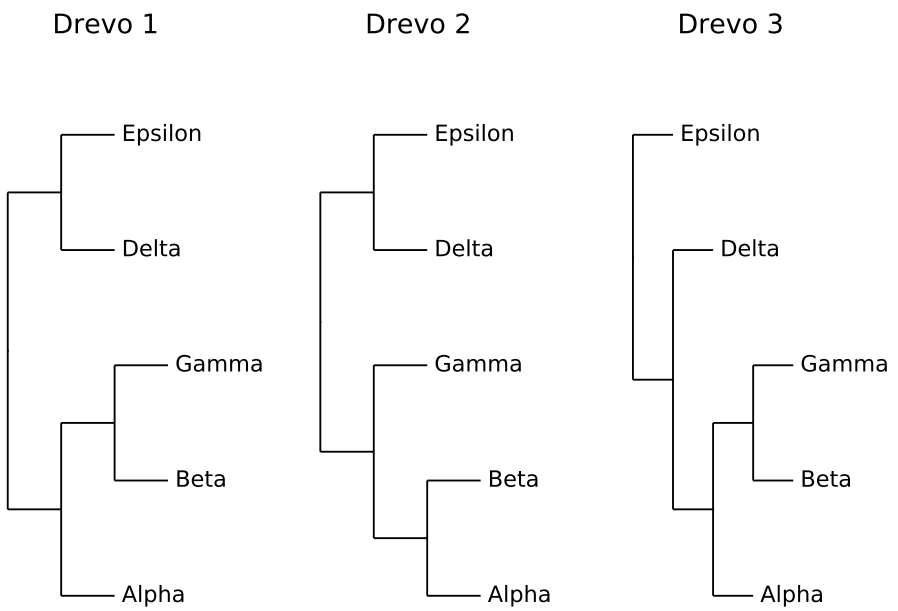
Poglavje 2

Filogenetske in konsenzne metode

V diplomski nalogi se povečamo konsenzni metodi asimetričnega srednjega drevesa. Da bi razumeli, zakaj so konsenzne metode v računski filogenetiki potrebne, bomo v tem razdelku zgolj na kratko predstavili, kako so filogenetska drevesa sploh zgrajena in kakšno vlogo pri tem igrajo konsenzne metode.

2.1 Filogenetske računske metode

Filogenetske računske metode lahko razdelimo na dve večji skupini, distančne in statistične metode. Vse kot vhod prejmejo DNA ali RNA sekvence taksonomskih enot, katerih evolucijsko zgodovino želimo rekonstruirati, kot izhod pa vrnejo eno ali več filogenetskih dreves, ki imajo konice (liste) označene z imeni taksonomskih enot. Nekatere metode so zmožne oceniti tudi dolžino vej drevesa, pri čemer dolžina veje predstavlja čas, ki je bil potreben za divergenco dveh taksonomskih enot iz skupnega prednika. Izračun dolžin vej sicer ni odvisen zgolj od izbrane metode, temveč tudi od izbranega modela molekularne ure. Primere treh filogenetskih dreves za pet taksonomskih enot prikazuje slika 2.1. Drevesa na sliki imajo dolžine vseh vej enake, med sabo pa se razlikujejo po topologiji.



Slika 2.1: Primer treh filogenetskih dreves z različnimi topologijami.

2.1.1 Distančne metode

Distančne metode iz vhodnih sekvenc DNA ali RNA s pomočjo izbranega evolucijskega modela najprej generirajo distančno matriko, v kateri so zapisane razdalje med vsemi pari sekvenc. Razdalje predstavljajo divergentne čase med dvema taksonomskima enotama, zato evolucijske modele lahko uporabimo tudi v kombinaciji z drugimi metodami za izračun dolžin vej filogenetskega drevesa. Za generiranje distančne matrike imamo na voljo več različnih evolucijskih modelov, med njimi:

- Jukes-Cantor (JC69) je najbolj enostaven model, ki predpostavlja, da je frekvenca vseh nukleotidov enaka in da so vse možne substitucije na enem baznem paru enako verjetne [9],
- Kimura (K80) model, ki substitucije baznega para razlikuje glede na tip in upošteva, da se lahko določen tip substitucije pojavi bolj pogosto [10]. Tipe substitucij delimo na tranzicije (sprememba purina v purin ali sprememba pirimidina v pirimidin) ter transverzije (sprememba purina

v pirimidina ali obratno) [5],

- General Time Reversible (GTR) je najnaprednejši model, ki evolucijo modelira kot stohastični proces. Ne predpostavlja neodvisnosti vsek lokacij vhodnih sekvenc, temveč šteje frekvence pojavitve nukleotidov glede na pozicijo v kodonu. Nato oceni šest parametrov, s pomočjo katerih zgradi matriko z verjetnostmi prehoda enega nukleotida v drugega [13].

Na podlagi razdalj, zapisanih v distančni matriki, izbrana metoda v gručo uvrsti po dve najbolj podobni taksonomski enoti hkrati (oz. povedano drugače, najbolj podobnima taksonomskima enotama določi skupnega prednika), dokler v gruče ne uvrsti vseh taksonomskih enot. Primera algoritmov, ki spadata v razred distančnih metod, sta UPGMA in metoda združevanja sosedov (angl. neighbor joining).

2.1.2 Metoda največje varčnosti

Metoda največje varčnosti spada med starejše, vendar še vedno zelo uporabljane metode. Glavna ideja algoritma je analogna principu Occamovega rezila - med več medseboj tekmujočimi teorijami evolucije izberemo tisto, ki minimizira število evolucijskih dogodkov. Metoda pregleda vsa potencialna drevesa in izbere tisto, ki vhodne sekvence lahko pojasni z najmanjšim številom potrebnih substitucij baznih parov. Tako drevo imenujemo najbolj varčno drevo (angl. most parsimonious tree) [14]. Najbolj varčnih dreves je lahko več, zato je izhodna množica najbolj varčnih dreves idealen primer vhodne množice za eno izmed metod za grajenje konsenznega drevesa.

2.1.3 Metoda največjega verjetja

Metoda maksimalnega verjetja (angl. maximum likelihood) je ena izmed statističnih metod za grajenje filogenetskih dreves. Če za neko drevo predpostavimo, da predstavlja pravilno evolucijsko zgodovino, potem njegovo verjetje predstavlja verjetnost pojavitve vhodnih sekvenc pri danem drevesu.

Metoda poišče drevo, ki verjetje maksimizira in ob tem predpostavlja, da so verjetnosti substitucij na različnih baznih parih medsebojno neodvisne. Najprej izračuna kako verjetno je predlagano drevo za vsako lokacijo vhodnih sekvenc posebej, nato pa še produkt teh vrendosti, ki predstavlja končno verjetje drevesa. Verjetje drevesa ne predstavlja verjetnosti, da je to drevo dejansko pravilno.

2.1.4 Bayesova inferenca

Bayesova inferenca je še en statistični pristop k iskanju filogenetskega drevesa. Zamisli za uporabo bayesovega teorema so se pojavile že konec šestdesetih let, vendar do nedavnega implementacija takih algoritmov ni bila smiselna zaradi velike računske zahtevnosti. Od metode maksimalnega verjetja se razlikuje zaradi uporabe vnaprej izbrane apriorne porazdelitve verjetnosti dreves [5]. Stroka ima zaradi tega sicer deljena mnenja o primernosti uporabe, vendar s tem pridobimo lepo lastnost, in sicer zmožnost interpretacije rezultata kot porazdelitev verjetnosti dreves glede na vhodne sekvence.

2.1.5 Ponovno vzorčenje

Ker nobena izmed metod ne zagotavlja pravilno generirane evolucijske zgodovine oz. pravilnosti ne moremo preveriti (razen v primeru laboratorijskih organizmov, kjer je razmerje vnaprej znano) [20], raziskovalci ponavadi uporabijo eno izmed metod ponovnega vzorčenja, npr. jackknife ali bootstrap, s katero vzorčijo lokacije vhodnih sekvenc in za vsak vzorec zgradijo svoje filogenetsko drevo. Vsako drevo, zgrajeno iz vzorca, se primerja s prvotno pridobljenim in izračuna se delež dreves, ki vsebujejo veje prvotnega drevesa. S pomočjo teh deležev lahko ocenimo, kolikšna je negotovost prvotno izračunane topologije filogenetskega drevesa [5].

Težava metod ponovnega vzorčenja je, da ocenjujejo zgolj negotovost v okviru ene metode. Raziskovalci lahko pridobijo več nasprotujočih si teorij o evolucijski zgodovini, naj si bo z uporabo različnih metod za računanje

filogenetskega drevesa ali iz različnih virov podatkov. Ob tem se pojavi potreba po eni teoriji evolucije, ki bi kar se da dobro zajemala vse evolucijske dogodke, ki jih nosijo med sabo tekmujoče si teorije. To lahko dosežemo z uporabo konsenznih metod.

2.2 Konsenzne metode

Konsenzne metode nad filogenetskimi drevesi so uporabljene po vsakem filogenetskem algoritmu, ki kot izhod proizvede več filogenetskih dreves. Načeloma lahko delujejo nad katero koli množico dreves, v kateri imajo drevesa enake množice oznak listov. V tem se bistveno razlikujejo od metod super-dreves, katere so sposobne kombinirati tudi drevesa, katerih množice oznak listov niso enake [7]. Poleg metode asimetričnega srednjega drevesa, ki jo obravnavamo v nadaljevanju tega dela, naštejmo nekatere najbolj pogoste konsenzne metode:

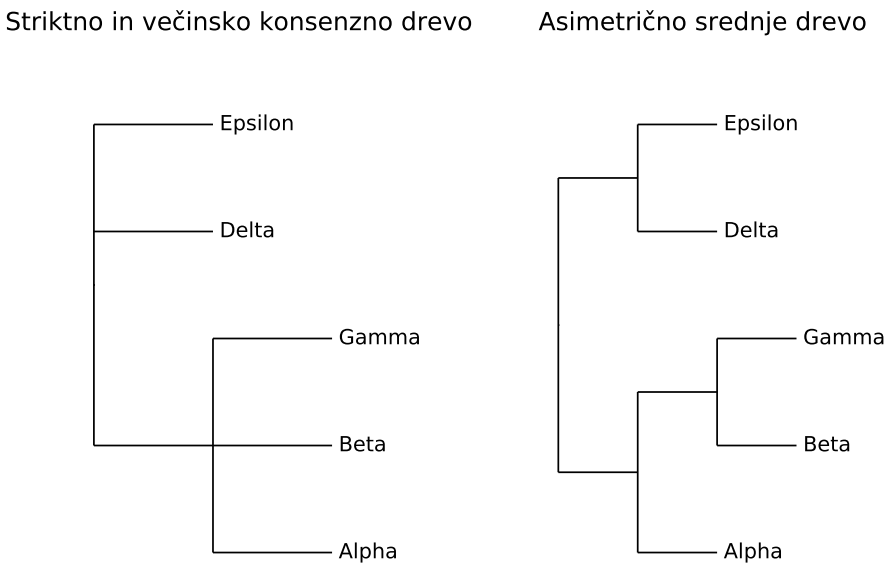
- kompatibilno drevo je sestavljeno iz delov vseh vhodnih dreves; ni nujno, da za vhodno množico dreves kompatibilno drevo dejansko tudi obstaja [2],

- striktni konsenz je najbolj konzervativna metoda, saj v končnem drevesu ohrani le tiste dele dreves, ki so prisotni v vseh drevesih vhodne množice [7]. Primer striktnega konsenznega drevesa za vhodna drevesa iz slike 2.1 je prikazan na sliki 2.2,

- večinski konsenz vsebuje le tiste dele drevesa, ki so prisotni v več kot polovici dreves vhodne množice [7]. Primer večinskega konsenznega drevesa za vhodna drevesa iz slike 2.1 je prikazan na sliki 2.2,

- srednji konsenz oz. mediana je drevo, ki minimizira vsoto simetričnih razlik glede na drevesa v vhodni množici. Večinsko drevo je prav tako srednje drevo, kar pomeni, da vedno obstaja vsaj eno srednje drevo [2].

- adamov konsenz je drevo, ki ohrani strukturo poddreves, prisotnih v vseh vhodnih drevesih. Vedno obstaja, vendar lahko vsebuje poddrevesa, ki niso prisotna v nobenem drevesu vhodne množice [3].
- Nelson-Pageva konsenza metoda je bila osnovana na ideji, da se eno drevo lahko moti, dve drevesi pa ne. Predpostavlja, da so deli dreves, prisotni v dveh ali več vhodnih drevesih, zelo verjetno resnični. Take dele imenujemo replicirane komponente. Nelson-Pagevo drevo je tako drevo, ki vsebuje vse replicirane komponente vhodne množice in vse preostale dele dreves vhodne množice, ki so z repliciranimi komponentami kompatibilni [3].



Slika 2.2: Na levi strani je prikazano striktno in večinsko konsenzno drevo za vhodna drevesa iz slike 2.1. Drevesi sta v tem primeru enaki. Na desni je za isto vhodno množico prikazano asimetrično srednje drevo.

Večina konsenznih metod ignorira dolžine vej v vhodnih drevesih, tudi če so te navedene. Namen konsenznih metod je torej pridobiti topologijo drevesa, s katero se čim bolj strinjajo vhodna drevesa, ne pa tudi usklajevanje divergentnih časov taksonomskih enot in njihovih skupnih prednikov.

2.3 Programska oprema za izračun filogenetskih dreves

Vse zgoraj opisane metode so že implementirane v samostojnih programskih paketih. Med najbolj znane spadajo:

- ClustalW2 Phylogeny, ki omogoča izračun filogenetskega drevesa s pomočjo distančnih metod UPGMA in združevanja sosedov, korekcijo razdalj pa lahko opravi s pomočjo evolucijskega modela K80 [11]. Poleg možnosti samostojne namestitve ga lahko uporabljamo tudi preko spletnega vmesnika [1],

- PHYLIP, paket samostojnih programov, ki ponuja izračun filogenetskih dreves s pomočjo distančnih metod, metode največje varčnosti in metode največjega verjetja. Poleg tega ponuja programe za ponovno vzorčenje ter program za izračun konsenznega drevesa s pomočjo striktnega in večinskega konsenza. Korekcije razdalj je mogoče opraviti z evolucijskimi modeli JC69, K80 in F84 [12].

- MrBayes, ki izračun filogenetskega drevesa opravi s pomočjo bayesove inference. Za izračun posteriorne verjetnostne porazdelitve uporablja markove verige v kombinaciji z Metropolis-Hastingsovim algoritmom za učinkovito preiskovanje prostora topologij. Ker na izhodu proizvede mnogo dreves, ponuja tudi izračun konsenznega drevesa s pomočjo metode večinskega konsenza [15].

- MEGA6, eno najbolj celovitih programskih okolij za izračun filogenetskih dreves, med drugim ponuja poravnavanje sekvenc pred njihovo uporabo z distančnimi metodami, metodami največje varčnosti ali metode maksimalnega verjetja. Ponuja široko paleto evolucijskih modelov z izjemo modela GTR, izračun konsenznih dreves s pomočjo striktnega ali večinskega konsenza in možnost ponovnega vzorčenja. Program

[1] `http://www.ebi.ac.uk/Tools/phylogeny/clustalw2_phylogeny/`

MEGA6 lahko uporabljamo preko ukazne vrstice, vse funkcionalnosti pa so podprte tudi preko grafičnega vmesnika [16].

- HashCS, program za izračun konsenza filogenetskih dreves, ki se šteje za enega najhitrejših programov te vrste. Implementira lastni algoritem, ki teče v polinomskem času. Ker je bil izdelan z namenom hitrejšega povzemanja dreves, ki so rezultat enega izmed algoritmov z uporabo bayesove inference, predpostavlja, da je število vhodnih dreves mnogo večje kot število taksonomskih enot [17].

Poglavje 3

Asimetrično srednje drevo

Metoda asimetričnega srednjega drevesa je nastala ob opažanju, da navadno srednje drevo kaznuje tiste dele dreves, ki ne nastopajo vsaj v polovici dreves vhodne množice, čeprav bi bilo smotrno, da se uporabi kar se da veliko informacije iz vseh vhodnih dreves [2].

Metoda kot vhod prejme množico k filogenetskih dreves $T = \{T_1, T_2, ..., T_k\}$, katera imajo liste označene z n taksonomskimi enotami iz množice $S = \{s_1, s_2, ..., s_n\}$. Sledijo naslednji koraki algoritma, katerega dele bomo podrobneje obravnavali v naslednjih poglavjih:

- drevesa pretvorimo v primerno reprezentacijo,
- poiščemo nekompatibilne pare poddreves in konstruiramo graf nekompatibilnosti,
- v grafu nekompatibilnosti poiščemo največjo neodvisno množico vozlišč,
- rekonstruiramo drevo iz vozlišč največje neodvisne množice.

3.1 Kodiranje dreves

Izbira predstavitve drevesa je pomembna za njihovo nadaljno manipulacijo. Izhajamo iz opažanja, da vsak rob drevesa, $e \in E(T_i)$, drevo razdeli na dva

dela oz. particiji. Do vsakega lista (taksonomske enote s) vodi enolična pot, predstavljena z množico robov, ki na tej poti nastopajo. Označimo jo s $\pi(s)$. Nato definiramo preslikavo dveh argumentov (3.1), ki zavzame vrednost 1 v kolikor pot do podanega lista poteka preko podanega roba, sicer pa 0. $c(e, s)$ za vsak rob $e \in E(T_i)$ torej definira biparticijo drevesa T_i [2]. V prvi množici so listi, do katerih lahko pridemo, če pot vodi preko roba e, v drugi pa tisti, do katerih v tem primeru ne moremo.

$$c(e, s) = \begin{cases} 1 & e \in \pi(s) \\ 0 & e \notin \pi(s) \end{cases} \tag{3.1}$$

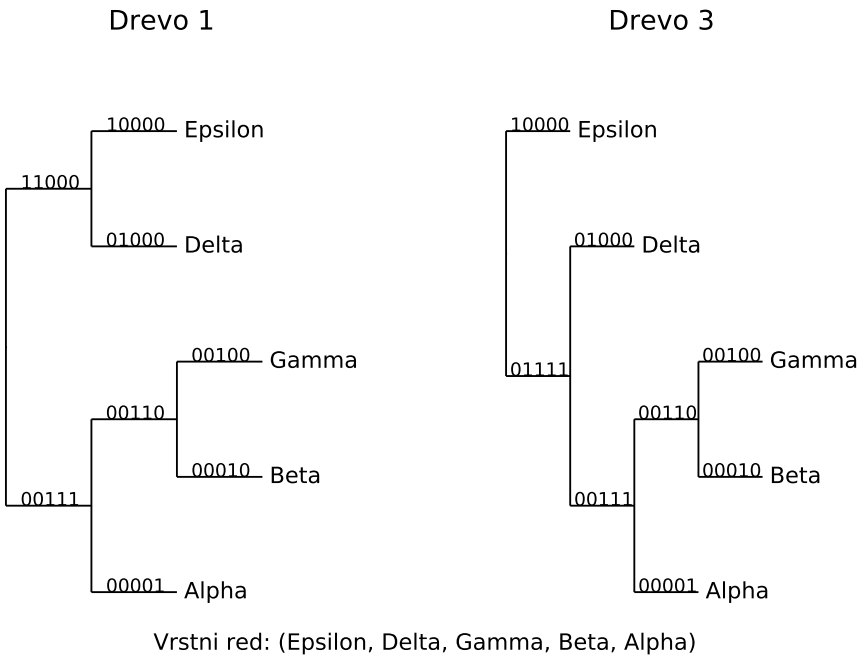
Vsak rob drevesa določa poddrevo, to pa predstavlja skupnega prednika s svojimi potomci. Takemu poddrevesu pravimo tudi klad (angl. clade). Ne bo nas zanimala celotna struktura poddreves, temveč zgolj njihovi listi. Vsak klad drevesa lahko predstavimo z binarnim nizom, ki ga dobimo z uporabo preslikave 3.2, pri čemer je potrebno povdariti, da je vrstni red posameznih znakov v binarnem nizu pomemben, saj je na vsak znak vezan pomen; vsako posamezno mesto v nizu pripada enemu izmed listov drevesa, znak na tem mestu pa zgolj indicira prisotnost ali neprisotnost lista oz. taksonomske enote v kladu, kodiranem z binarnim nizom.

$$c_e = \{c(e, s) : s \in S\} \tag{3.2}$$

V kolikor binarne nize vseh kladov drevesa zberemo v množico (3.3), nam $C(T_i)$ določa kodiranje celotnega drevesa T_i. Primer dveh takih množic je prikazan na sliki 3.1, kjer so binarni nizi prikazani na vejah, ki vodijo do korenov kladov.

$$C(T_i) = \{c_e : e \in E(T_i)\} \tag{3.3}$$

Kodiranje sedaj lahko izvedemo nad vsemi drevesi vhodne množice T, pri čemer je ponovno potrebno povdariti, da se pomen posameznih mest v



Slika 3.1: Vsi binarni nizi za prvo in tretje drevo iz slike 2.1.

binarnih nizih med različnimi drevesi ne sme spreminjati, saj sicer kladov iz različnih dreves ne bi mogli medsebojno primerjati.

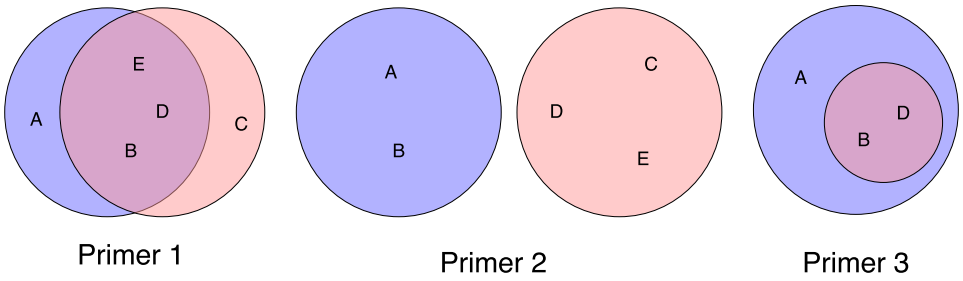
3.2 Kompatibilnost binarnih nizov

Za konstrukcijo grafa nekompatibilnosti najprej potrebujemo kriterij kompatibilnosti kladov oz. binarnih nizov, ki jih kodirajo. Za potrebe lažje predstavitve kompatibilnosti za vsak binarni niz c_e definiramo množico $\phi(c_e)$ (3.4), ki vsebuje oznake taksonomskih enot, prisotnih v kladu oz. njegovem pripadajočem binarnem nizu c_e.

$$\phi(c_e) = c_e^{-1}(1) = \{s \in S : c(e, s) = 1\} \tag{3.4}$$

$$\phi(j) \cap \phi(k) = \emptyset \vee \phi(j) \subseteq \phi(k) \vee \phi(k) \subseteq \phi(j) \tag{3.5}$$

Če za binarna niza j in k iz poljubnih dveh dreves vhodne množice T



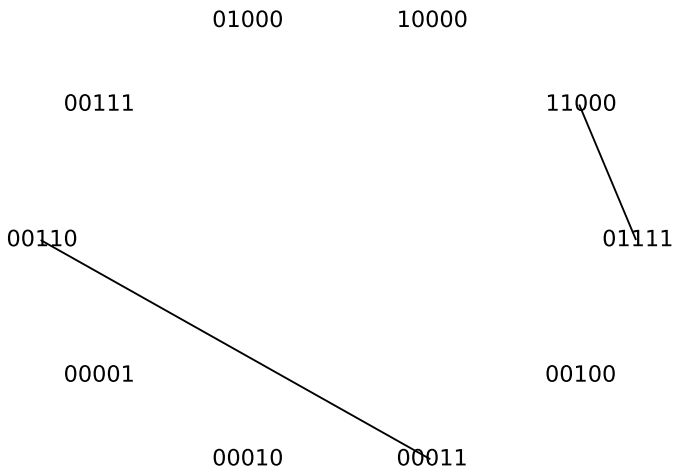
Slika 3.2: Primera 2 in 3 prikazujeta kompatibilni množici ϕ, v primeru 1 pa sta množici nekompatibilni.

velja pogoj 3.5, potem sta binarna niza kompatibilna. Vennov diagram pripadajočih množic $\phi(j)$ in $\phi(k)$ prikazuje slika 3.2. V kolikor velja $|\phi(j)| = 1$ ali $|\phi(k)| = 1$, potem sta binarna niza j in k vedno kompatibilna.

V kolikor kompatibilnost kladov interpretiramo v smislu filogenetskega drevesa, potem posamezen klad predstavlja poddrevo s korenom, ki je notranje vozlišče drevesa T_i. Notranje vozlišče filogenetskega drevesa predstavlja skupnega prednika vsem vozliščem v poddrevesu pod njim. Nekompatibilni sta tisti poddrevesi, ki trdita, da se je iz obeh skupnih prednikov razvilo nekaj skupnih taksonomskih enot, a je hkrati v vsaj enem poddrevesu prisotna taksonomska enota, ki je v drugem poddrevesu ni. V tem primeru poddrevesi torej ponujata nasprotujoči si informaciji o evolucijski zgodovini.

3.3 Graf nekompatibilnosti

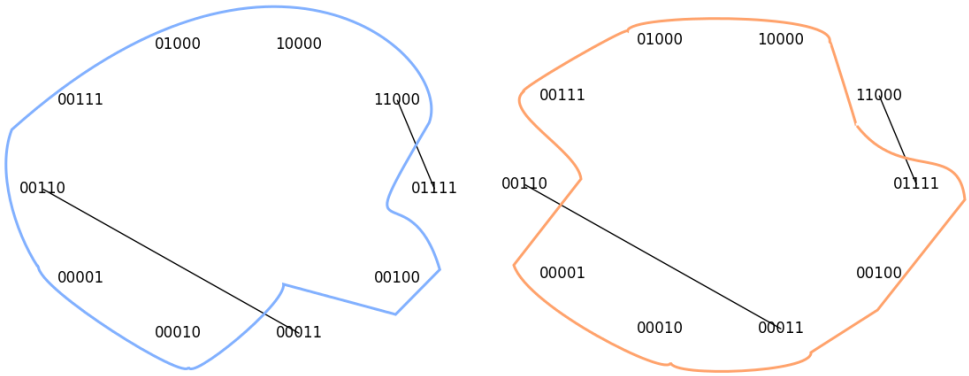
Z uporabo kriterija nekompatibilnosti binarnih nizov lahko zgradimo graf nekompatibilnosti $I(V_1, V_2, ..., V_k, E) = I(V, E)$. Vsaka množica vozlišč V_i predstavlja binarne nize $C(T_i)$, povezave v grafu pa vzpostavimo med tistimi vozlišči, katerih pripadajoči binarni nizi so med sabo nekompatibilni [2]. S povezovanjem vozlišč označimo binarne nize, ki v končnem drevesu skupaj ne morejo biti prisotni, sicer bi filogenetsko drevo podajalo dvoumne informacije



Slika 3.3: Graf nekompatibilnosti biznih nizov za vhodna drevesa iz slike 2.1. Vozlišča predstavljajo unikatne binarne nize iz vseh dreves vhodne množice T, povezave pa so vzpostavljene med pari vozlišč, katerih binarni nizi medsebojno niso kompatibilni.

o evolucijski zgodovini.

Graf nekompatibilnosti vseh treh vhodnih dreves iz slike 2.1 je prikazan na sliki 3.3. V prvem in drugem drevesu sta medsebojno nekompatibilna klada $(Gamma, Alpha)$ in $(Gamma, Beta)$. Drevesi ponujata nasprotujoči si informaciji - prvo za najbolj sorodno taksonomski enoti *Gamma* ponuja taksonomsko enoto *Alpha*, drugo pa taksonomsko enoto *Beta*. Prvo in drugo drevo imata medsebojno kompatibila klada $(Epsilon, Delta)$, ki pa sta nekompatibilna s kladom $(Delta, Gamma, Beta, Alpha)$ iz tretjega vhodnega drevesa. Medtem ko prvo in drugo drevo trdita, da imata taksonomski enoti *Epsilon* in *Delta* neposrednega skupnega prednika, tretje drevo temu nasprotuje.



Slika 3.4: Dve različni največji neodvisni množici za graf nekompatibilnosti iz slike 3.3. Na danem grafu sicer obstajajo štiri največje neodvisne množice.

3.4 Največja neodvisna množica

Graf nekompatibilnosti vsebuje vse informacije, ki jih potrebujemo za izbiro binarnih nizov, ki bodo prisotni v končnem filogenetskem drevesu. Vanj želimo vključiti kar se da veliko informacij, prisotnih v drevesih vhodne množice T. Najprej je potrebno zagotoviti, da vključimo binarne nize, ki so skupni vsem vhodnim drevesom T_i (drevo, ki bi vsebovalo le te binarne nize, bi bilo striktno konsenzno drevo). Obenem želimo vključiti čimveč preostalih binarnih nizov, vendar ne takih, da bi bil katerikoli par nekompatibilen.

Matematično orodje, ki nam iz grafa $I(V, E)$ omogoča izbiro vozlišč, ki paroma ne bodo kršile pogoja nekompatibilnosti, se imenuje neodvisna množica (angl. independent set). Neodvisna množica grafa je katerakoli množica vozlišč $V_{Indep} \subseteq V$, med katerimi ni medsebojnih povezav. Največja neodvisna množica grafa (angl. maximum independent set - MIS) je tista neodvisna množica $V_{MIS} \subseteq V$, ki vsebuje največ vozlišč. Potrebno je povdariti, da za nek graf $I(V, E)$ lahko obstaja več različnih V_{MIS}. Ker vozlišča V_{MIS} predstavljajo binarne nize oz. klade končnega drevesa, to pomeni, da za en graf $I(V, E)$ lahko obstaja več asimetričnih srednjih dreves.

Vozlišča grafa $I(V, E)$, katerega sestavimo iz k dreves vhodne množice

T, sestavljajo neodvisne množice $V_1, V_2, ..., V_k$. Vozlišča znotraj ene množice nikoli niso medsebojno povezana, saj njihovi pripadajoči binarni nizi pripadajo istemu drevesu. Vozlišči iz različnih množic pa lahko tvorita povezavo, v kolikor sta nekompatibilni. Tak graf imenujemo tudi k-partitni graf. Za bipartitne grafe obstaja trivialen polinomski algoritem s časovno zahtevnostjo $O(n^2)$, v splošnem pa problem največje neodvisne množice za k-partitne grafe sodi v razred NP-težkih algoritmov [2].

3.5 Rekonstrukcija drevesa

S tem, ko smo določili največjo neodvisno množico V_{MIS} grafa $I(V, E)$, smo določili klade končnega drevesa, katerega strukturo pa je potrebno še rekonstruirati iz izbranih binarnih nizov. Problem rekonstrukcije evolucijske zgodovine n taksonomskih enot iz njihovih binarnih nizov je dobro znan in se imenuje problem filogenije. Zanj obstaja algoritem s časovno kompleksnostjo $O(nm)$, pri čemer je n število taksonomskih enot, m pa število binarnih nizov v največji neodvisni množici [4].

Najprej kreiramo matriko M dimenzij $n \times m$. Binarne nize, ki pripadajo vozliščem množice V_{MIS} kot stolpce zložimo v matriko M. Nato stolpce interpretiramo kot števila v binarnem zapisu, jih pretvorimo v desetiško obliko in sortiramo. Odstranimo stolpce s podvojeno vrednostjo, tako da ostanejo le stolpci z unikatnimi števili. S tem pridobimo matriko M'.

Naj bo I_i množica indeksov vrstic, I_j pa množica indeksov stolpcev matrike M'. Množica E (3.6) predstavlja vse pare indeksov $(i, j) \in I_i \times I_j$ matrike M', katerih vrednost je enaka ena.

$$E = \{(i, j) : i \in I_i, j \in I_j, M'(i, j) = 1\} \tag{3.6}$$

$$P(i, j) = \begin{cases} \max(\{k : (i, k) \in E, k < j\}) \\ 0 \text{ če tak } k \text{ ne obstaja} \end{cases} \tag{3.7}$$

Vsak par $(i, j) \in E$ predstavlja klad j, v katerem je prisotna taksonomska enota i. Za vsak tak par poiščemo vrednost $P(i, j)$, ki ustreza indeksu

$$\begin{array}{l} \\ Epsilon \\ Delta \\ Gamma \\ Beta \\ Alpha \end{array}\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{array}{c} i/j \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} = M'$$

$$E = \{(1,1),(1,2),(2,1),(2,3),(3,4),(3,5),(3,6),(4,4),(4,5),(4,7),(5,4),(5,8)\}$$

$$P[i,j] = \begin{array}{c} i/j \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 1 & / & / & / & / & / & / \\ 0 & / & 1 & / & / & / & / & / \\ / & / & / & 0 & 4 & 5 & / & / \\ / & / & / & 0 & 4 & / & 5 & / \\ / & / & / & 0 & / & / & / & 4 \end{pmatrix} \end{array} \quad P[j] = \begin{array}{c} j \\ \end{array}\begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 4 & 5 & 5 & 4 \end{pmatrix} \end{array}$$

Slika 3.5: Primer izračuna matrike M', množice parov indeksov E in vseh vrednosti $P(i,j)$ ter $P(j)$ za prvo največjo neodvisno množico iz slike 3.4. Ker enačba 3.9 v tem primeru velja, iz matrike M' lahko zgradimo filogenetsko drevo.

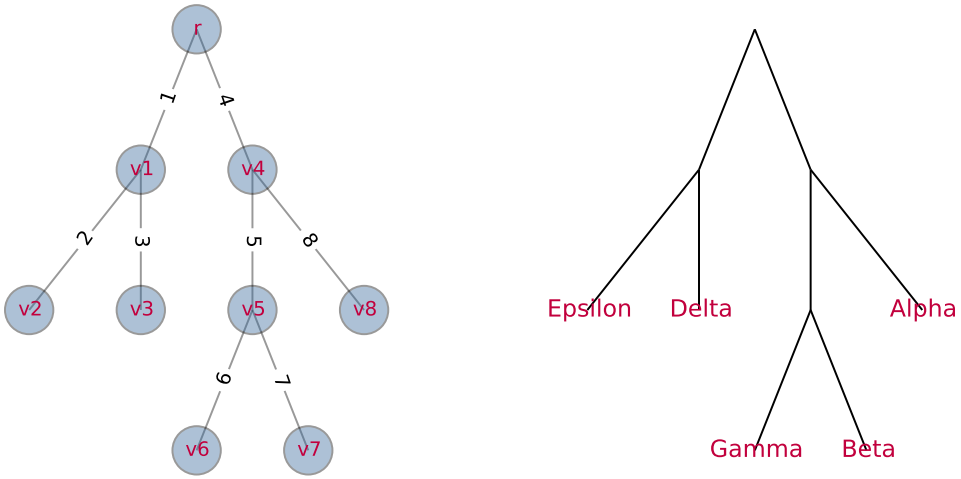
klada k, v katerem je vsebovan klad j (3.7), pri čemer tudi klad k vsebuje taksonomsko enoto i. V kolikor tak klad ne obstaja, to pomeni, da ima klad j svoj koren pozicioniran najbolj blizu korena celotnega končnega drevesa izmed vseh kladov s prisotno taksonomsko enoto i.

Ko imamo poračunane vse vrednosti $P(i,j)$, lahko za vsak $j \in I_j$ poiščemo vrednost $P(j)$(3.8). Ta predstavlja najvišji indeks klada, v katerem je klad j vsebovan. Ker so kladi sortirane padajoče glede na njihove desetiške vrednosti, najvišji indeks pravzaprav pomeni, da iščemo očeta klada j, ki je na čim nižjem nivoju v končnem drevesu, tako da je klad j še v celoti vsebovan.

$$P(j) = \max(\{P(i,j) : (i,j) \in E\}) \tag{3.8}$$

$$P(i,j) = P(j) \qquad \forall (i,j) \in E \tag{3.9}$$

V kolikor velja enačba 3.9, potem za matriko M' obstaja filogenetsko drevo [4]. V tem primeru se lahko lotimo njegove konstrukcije. Najprej ustvarimo graf, v katerega kot vozlišča dodamo koren bodočega drevesa r



Slika 3.6: Na levi strani je prikazan graf, ki je rekonstruiran iz vrednosti $P[j]$ iz slike 3.5. Da iz grafa pridobimo končno drevo, prikazano na desni strani, za označevanje listov uporabimo matriko M' iz slike 3.5.

in za vsak $j \in I_j$ svoje vozlišče v_j. Nato za vsako vozlišče v_j, za katerega velja $P(j) > 0$, ustvarimo povezavo $(v_{P(j)}, v_j)$ in povezavo označimo z j. Vsa preostala vozlišča v_j, za katere velja $P(j) = 0$, povežemo s korenom drevesa s pomočjo povezave (r, v_j), in prav tako označimo z j. Primer takega grafa, konstruiranega za matriko M' iz slike 3.5 je prikazan na levi strani slike 3.6. Tak graf je že drevo, vendar še ne filogenetsko.

Preden konstruirano drevo postane filogenetsko drevo, moramo razrešiti oznake na povezavah. Razreševanja se lotimo po vrsticah matrike M', saj vsaka vrstica pravzaprav predstavlja eno taksonomsko enoto. Za vsak $i \in I_i$ najdemo največji j, za katerega velja $M'(i, j) = 1$. Dobljeni j nam predstavlja oznako povezave, na koncu katere je prisoten list, kateremu dodelimo oznako taksonomske enote trenutne vrstice i. To storimo za vse vrstice, s čimer označimo vse liste. Kot prikazuje primer dokončanega drevesa na desni strani slike 3.6, za konec notranjim vozliščem in povezavam odstranimo oznake. S tem smo prišli do končnega filogenetskega drevesa, ki ga označimo s τ.

3.6 Vrednost asimetričnega srednjega drevesa

Omenili smo že, da za en graf nekompatibilnosti lahko obstaja več največjih neodvisnih mnnožic in da to pomeni, da lahko sestavimo več različnih asimetričnih srednjih dreves. Prvotni razlog za uporabo konsenznih metod je iz večih dreves pridobiti eno drevo, zato potrebujemo metriko, s pomočjo katere se bomo lahko odločili za eno, najbolj podprto drevo.

Najprej uvedemo utež binarnega niza $w(c_e)$ (3.10), ki je preprosto število dreves v vhodni množici T, ki vsebujejo binarni niz c_e. Vrednost asmetričnega srednjega drevesa τ glede na vhodno množico T je določena (3.11) kot vsota uteži binarnih nizov, ki so prisotni v asimetričnem srednjem drevesu, vendar niso prisotni v vseh vhodnih drevesih (oz. striktnem konsenznem drevesu) [2].

$$w(c_e) = |\{i : c_e \in C(T_i)\}| \tag{3.10}$$

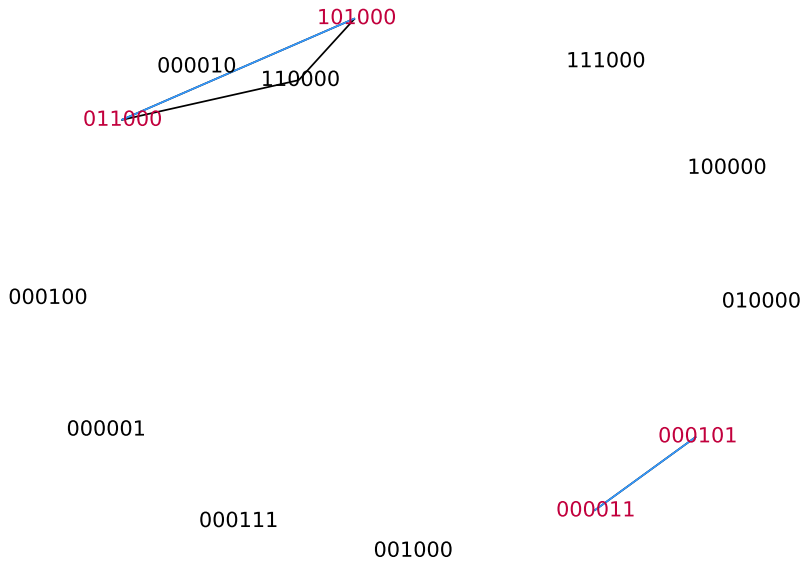
$$value(T, \tau) = \sum_{c \in C(\tau) - \cap C(T_i)} w(c) \tag{3.11}$$

Najboljše najdeno asimetrično srednje drevo je tisto, ki maksimizira dano vsoto uteži 3.11.

3.7 Aproksimacijski algoritmi

Ker je problem konstrukcije asimetričnega srednjega drevesa polinomsko prevedljiv na problem največje neodvisne množice [2] in zato za več kot dve drevesi sodi v razred NP-težkih problemov, je lahko izvajanje algoritma prepočasno, če imamo v vhodni množici T veliko dreves. Zato si bomo ogledali dva aproksimacijska algoritma, ki imata polinomsko časovno zahtevnost.

Prvi, $\frac{2}{k}$-aproksimacijski algoritem, je enostaven. Za vsak par dreves $T_i, T_j \in T$ izračunamo asimetrično srednje drevo τ_{ij} in izberemo tistega, ki maksimizira vrednostno funkcijo $value(T, \tau_{ij})$. Ker je tako asimetrično srednje drevo sestavljeno iz dveh dreves, posledično lahko vsebuje le klade, ki so prisotni v teh dveh drevesih.



Slika 3.7: Primer računanja bitnih nizov asimetričnega srednjega drevesa s pomočjo največjega ujemanja. Robovi, ki spadajo v množico največjega ujemanja E_M so obarvani modro, njihova krajišča pa rdeče. Vsi bitni nizi, ki so obarvani črno, sestavljajo končno drevo.

V kolikor imajo drevesa vhodne množice T veliko skupnih kladov, je primernejši drugi algoritem. Naj bo $V^* \subseteq V$ podmnožica vozlišč grafa nekompatibilnosti $I(V, E)$, in sicer tistih, ki so skupna vsem drevesom. Takih vozlišč v naslednjem koraku ne potrebujemo, saj niso povezana z nobenim drugim vozliščem. Nato izračunamo največje ujemanje (angl. maximum matching) v grafu $I(V - V^*, E)$, s čimer pridobimo množico robov $E_M \subseteq E$, ki nimajo skupnih vozlišč [22]. Vozlišča V_M, ki so krajišča robov iz množice E_M, odstranimo iz množice V in ostanejo nam le neujemajoča vozlišča $V_N = V - V_M$. Primer iskanja množice V_N je prikazan na sliki 3.7. Iz binarnih nizov, pripadajočih vozliščem množice V_N, rekonstruiramo asimetrično srednje drevo po običanjem postopku. Tako drevo, za razliko od drevesa izračunanega s pomočjo prvega aproksimacijskega algoritma, lahko vsebuje klade iz večih dreves vhodne množice hkrati [2].

Poglavje 4

Implementacija algoritma

4.1 Biopython

Biopython[1] je odprtokodni paket modulov, napisan v programskem jeziku Python, ki ponuja mnoge funkcionalnosti s področja bioinformatike. Olajša nam delo s formati datotek, kot so BLAST, ClustralW, FASTA ter GenBank in ponuja enostaven dostop do spletnih storitev, npr. NCBI [21]. Poleg funkcionalnosti, implementiranih v programskem jeziku Python ponuja tudi enostaven dostop do zunanjih programov. Nekateri glavni moduli paketa Biopython, ki so relavantni tudi za računanje filogenetskih dreves, so:

- *Bio.SeqIO*: razredi, ki nam omogočajo branje, pisanje in manipuliranje sekvenc v različnih formatih ($FASTA$, $Nexus$, ...),
- *Bio.AlignIO*: razredi, ki nam omogočajo branje, pisanje in manipuliranje poravnanih sekvenc,
- *Bio.Application*: razredi, s pomočjo katerih enostavno dostopamo do zunanje namenske programske opreme, kot npr. PhyML in RAxML
- *Bio.Phylo*: razredi in funkcije za izračun filogenetskih dreves

[1] `http://biopython.org/`

Modul *Bio.Phylo* že omogoča izračun filogenetskih dreves s pomočjo metod UPGMA, Neighbor Joining in Maximum Parsimony. Poleg tega ponuja tudi metode ponovnega vzorčenja bootstrap in jackknife.

Izračun konsenznih filogenetskih dreves je mogoč s pomočjo podmodula *Bio.Phylo.Consensus* v katerem so bile najprej implementirane metode striktnega, večinskega in adamovega konsenza, sedaj pa izračun lahko opravimo tudi s pomočjo metode asimetričnega srednjega drevesa. Branje in pisanje dreves je že omogočeno za formate Newick, CDAO in PhyloXML. Modul ponuja tudi vmesno reprezentacijo drevesa, ki ga uporabljamo med izračunom in ni vezan na končni format.

4.2 Podrobnosti implementacije

Glavna entiteta, ki je nastopala v obravnavi metode asimetričnega srednjega drevesa je bil bitni niz. Zato je pomembno, da zanj uporabimo primerno hitro podatkovno strukturo. Razred, ki podatkovno strukturo že implementira se imenuje *_BitString*, prebiva pa v podmodulu *Bio.Phylo.Consensus*. *_BitString* razširja Pythonovo vgrajeno podatkovno strukturo *String*, zato ga lahko uporabljamo na enak način. Med operatorji, ki jih lahko izvajamo nad podatkovno strukturo, so konjunkcija, disjunkcija in ekskluzivna disjunkcija nad posameznimi mesti, vsebuje pa tudi metodi za preverjanje vsebovanosti enega bitnega niza v drugem in preverjanje kompatibilnosti dveh binarnih nizov.

Modul *Bio.Phylo* za izrisovanje filogenetskih dreves uporablja paket networkx[2]. Ta poleg samega izrisovanja ponuja tudi konstrukcijo grafov in njihovo nadaljno manipulacijo. Na grafu zmore izračunati aproksimacijo največje neodvisne množice, a se s tem nismo mogli zadovoljiti, saj je bila aproksimacija že za majhna vhodna drevesa preveč nenatančna, da bi dobili uporaben rezultat. Na srečo je problem največje neodvisne množice polinomsko prevedljiv na problem največje klike v komplementarnem grafu,

[2] `https://networkx.github.io/`

katero networkx zna poiskati. Kljub temu se je iskanje vseh največjih klik komplementarnega grafa izkazalo za dokaj počasno. Ob opazovanju, da naš algoritem izrablja le eno procesorsko jedro, smo se odločili, da lahko iskanje največje klike opravimo s pomočjo zunanjega programa Parallel Maximum Clique (PMC) [18], v kolikor zaznamo prisotnost programa na uporabnikovem sistemu. Čeprav smo s tem vnesli dodatno odvisnost od zunanje programske opreme, lahko odločitev utemeljimo s precej večjo hitrostjo implementacije PMC v primerjavi z lastno implementacijo v programskem jeziku Python. V kolikor program ni najden, se iskanje največjih klik kljub temu opravi z uporabo paketa networkx.

Pri uporabi druge aproksimacijske metode za izračun asimetričnega srednjega drevesa ne računamo največje neodvisne množice, temveč največje ujemanje grafa. Paket networkx za reševanje problema vsebuje funkcijo *max_weight_matching()*, pri čemer so vse uteži robov grafa enake 1.

Za rekonstrukcijo drevesa smo potrebovali orodje, ki je zmožno dela z matrikami. Ker je paket Biopython že odvisen od paketa numpy[3], je ta bil logična izbira. Med kreiranjem matrike je bilo potrebno binarne nize pretvoriti v stolpce matrike. Za to potrebo smo razredu _BitString dodali metodo *to_numpy*(), katera niz ničel in enic pretvori v binarni vektor tipa *numpy.ndarray* z eno dimenzijo. Za potrebe sortiranja smo razredu *_BitString* dodali še metodo *__int__*(), ki izračuna desetiško vrednost binarnega niza.

Slika 4.1 prikazje glavo implementirane funkcije, ki se nahaja v modulu *Bio.Phylo.Consensus*. Opis parametrov funkcije se nahaja v tabeli 4.1. Privzeto metoda izračuna točno asimetrično srednje drevo, vendar je računska kompleksnost za več kot dve drevesi eksponentna, zato lahko uporabnik, v kolikor to želi, z drugim parametrom izbere eno izmed aproksimacijskih metod.

[3] `http://www.numpy.org/`

```
1 |       def amt_consensus(trees, method='no_approx')
```

Slika 4.1: Glava funkcije za izračun asimetričnega srednjega drevesa iz modula *Bio.Phylo.Consensus*.

Parameter	Tip	Vrednost parametra	Opis
trees	seznam objektov tipa *BaseTree.Tree*	/	Seznam vhodnih dreves v kateremkoli formatu, ki ga podpira Biopython.
method	niz		Parameter za izbiro metode izračuna.
		no_approx	Izračun točnega drevesa.
		bi_approx	Izračun iz največ dveh dreves.
		maxmatch_approx	Izračun s pomočjo največjega ujemanja.

Tabela 4.1: Opis parametrov in njihovih vrednosti metode *Bio.Phylo.Consensus.amt_consensus*.

4.3 Primer uporabe

Primer uporabe programske kode je najbolj enostavno prikazati na primeru. Recimo, da imamo pripravljeno datoteko (4.2) z imenom *primer_a.tre*, ki vsebuje pet dreves v formatu Newick. Iz teh dreves želimo izračunati asimetrično srednje drevo in ga izrisati.

Uporaba konsenzne metode asimetričnega srednjega drevesa v paketu Biopython je enostavna. Slika 4.3 predstavlja osnoven primer uporabe. V prvi vrstici uvozimo modul *Bio.Phylo*, v drugi iz podmodula *Bio.Phylo.Consensus* uvozimo funkcijo *amt_consensus*, v tretji pa uvozimo paket za risanje *pyplot*.

V peti vrstici uvozimo filogenetska drevesa iz datoteke *primer_a.tre* in jih razčlenimo, s čimer dobimo seznam objektov tipa *Bio.Phylo.BaseTree.Tree*. V šesti vrstici s preprostim klicem funkcije iz obstoječega seznama dreves

```
((A, (B, C)), (D, (E, F)));
((B, (A, C)), (D, (E, F)));
((A, (B, C)), (E, (D, F)));
((B, (A, C)), (E, (D, F)));
((C, (B, A)), (E, (D, F)));
```

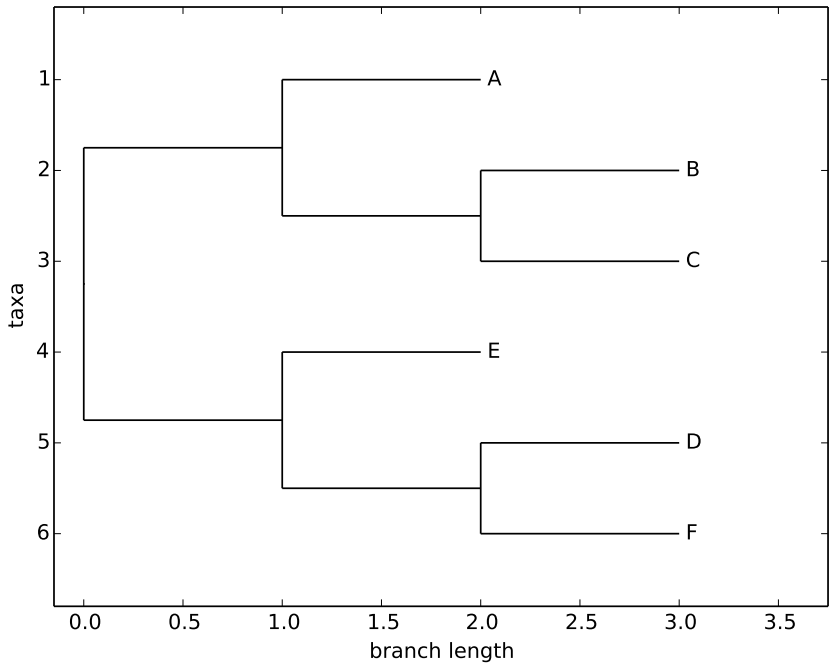
Slika 4.2: Primer petih polno razrešenih filogenetskih dreves s šestimi taksonomskimi enotami, ki so kodirana v formatu Newick.

```
1 from Bio import Phylo
2 from Bio.Phylo.Consensus import amt_consensus
3 from matplotlib import pyplot
4
5 trees = list(Phylo.parse('primer_a.tre', 'newick'))
6 amt = amt_consensus(trees)
7 amt.ladderize()
8 Phylo.draw(amt)
9 pyplot.show()
```

Slika 4.3: Primer programske kode za branje dreves iz datotek in izračun asimetričnega srednjega drevesa.

izračunamo asimetrično srednje drevo. V sedmi vrstici drevo preoblikujemo tako, da bodo globlja poddrevesa prikazana na vrhu, in v zadnjih dveh vrsticah drevo izrišemo na ekran.

Rezultat je programa iz figure 4.3 je prikazan na sliki 4.4. Poleg drevesa so prikazane še osi, ki v našem primeru sicer niso pomembne, v splošnem pa iz osi x lahko razberemo divergentne čase taksonomskih enot.



Slika 4.4: Rezultat programa iz figure 4.3.

Poglavje 5

Eksperimentalna primerjava

Vse tri implementirane metode računanja asimetričnega srednjega drevesa želimo primerjati med sabo in glede na metode striktnega, večinskega in adamovega konsenza. Za ta namen smo pripravili nekaj vhodnih množic in jih preizkusili glede na razrešenost končnega drevesa ter glede na Robinson-Fouldsovo metriko. Programska koda s katero smo pridobili rezultate se nahaja v dodatku A. Poleg lastnosti izhodnih dreves nas je zanimal tudi čas izvajanja vseh treh implementiranih metod, s čimer smo ocenili velikost vhodne množice in število taksonomskih enot v drevesih, za katere zaradi predolgega časa izvajanja ni več smiselno uporabiti natančne metode. Koda za merjenje časa se nahaja v dodatku A.

5.1 Razrešenost drevesa in Robinson-Fouldsova metrika

Razrešenost drevesa (5.1) je definirana kot število povezav, ki nastopajo v drevesu. Ker imajo vsa drevesa enako število listov, se je drevo z največjim številom povezav največkrat razvejalo. Najbolj razrešeno drevo je binarno drevo (takrat rečemo, da je *polno razrešeno*). To ponuja največ informacije o evolucijski zgodovini, saj za vsak par taksonomskih enot vemo, iz katerega skupnega prednika sta se enoti razvili. Zaželjeno je torej, da je število povezav

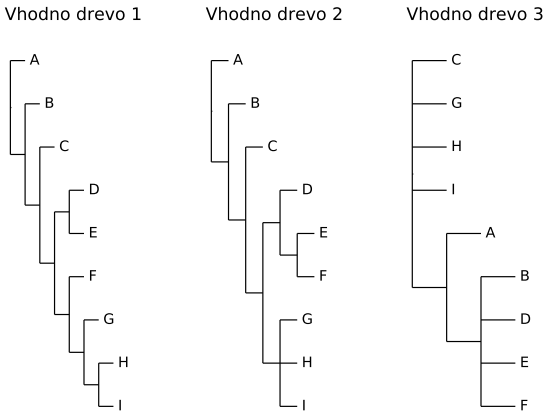
v drevesu čim večje.

$$Res(T) = |E(T)| \tag{5.1}$$

$$RF(T_1, T_2) = |C(T_1) - C(T_2)| + |C(T_2) - C(T_1)| \tag{5.2}$$

Robinson-Fouldsova (RF) metrika je najbolj razširjena metrika za primerjanje filogenetskih dreves. Šteje število kladov (5.2), ki si jih drevesi ne delita [8]. Za izračun vrednosti RF smo uporabili funkcijo *symmetric_difference()* iz paketa DendroPy[1], saj v paketu Biopython RF metrika še ni implementirana. Vrednost RF smo izračunali za vsak par konsenznega in vhodnega drevesa, ter za vsako konsenzno drevo vrednosti RF sešteli.

5.2 Vhodna množica 1



Slika 5.1: Prva vhodna množica dreves.

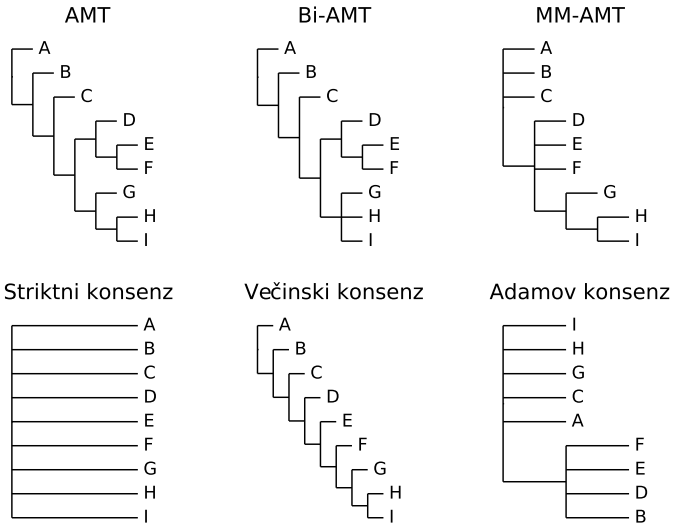
Tabela 5.2 prikazuje razrešenost in RF vrednosti asimetričnega srednjega drevesa, dveh aproksimiranih asimetričnih srednjih dreves (Bi-AMT in MM-AMT), striktnega, večinskega in adamovega konsenznega drevesa iz slike 5.2

[1] `https://pythonhosted.org/DendroPy/`

	Res (T)	RF (T_1)	RF (T_2)	RF (T_3)	RF (vsota)
AMT	17	4	1	8	13
Bi-AMT	16	5	0	7	12
MM-AMT	13	3	4	5	12
Striktni konsenz	10	6	5	2	13
Večinski konsenz	17	2	5	4	11
Adamov konsenz	11	5	6	1	12

Tabela 5.1: Razrešenost konsenznih dreves in njihove RF vrednosti za vhodna drevesa na sliki 5.1.

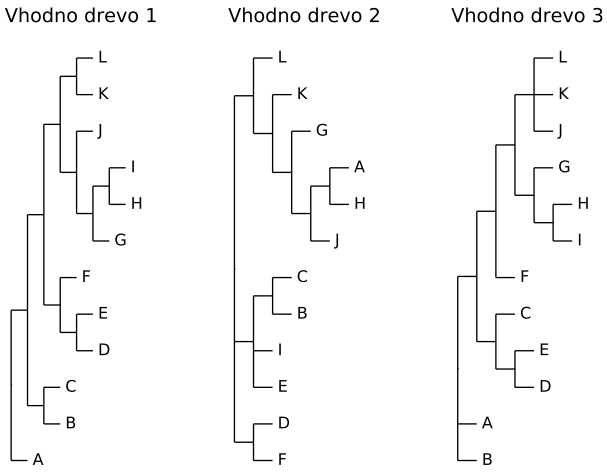
glede na vhodna drevesa iz slike 5.1. Opazimo, da je asimetrično srednje drevo glede na razrešenost prav tako dobro kot večinsko konsenzno drevo. Obe drevesi sta polno razrešeni. Glede na vrednosti RF je vhodnim drevesom najbolj podobno večinsko konsenzno drevo. Tako lahko brez dvoma večinsko konsenzno drevo v tem primeru razglasimo za najboljše.



Slika 5.2: Konsenzna drevesa, zgrajena iz množice dreves na sliki 5.1. Oznaka Bi-AMT označuje približek asimetričnega srednjega drevesa zgrajenega iz dveh vhodnih dreves, MM-AMT pa aproksimirano drevo zgrajeno s pomočjo največjega ujemanja.

Rezultat je pričakovan, saj je večina kladov prisotnih v več kot polovici vhodnih dreves, preostali pa so povečini medsebojno kompatibilni, kar je za večinsko drevo ugodno. Najbolj podobno asimetrično srednje drevo je MM-AMT, vendar je njegova razrešenost med najslabšimi. Drevo Bi-AMT ne podaja informacije o skupnem predniku taksonomskih enot H in I, zaradi česar je bolj podobno drugemu in tretjemu vhodnemu drevesu kot drevo AMT, posledično pa je njegova razrešenost zaradi tega slabša.

5.3 Vhodna množica 2



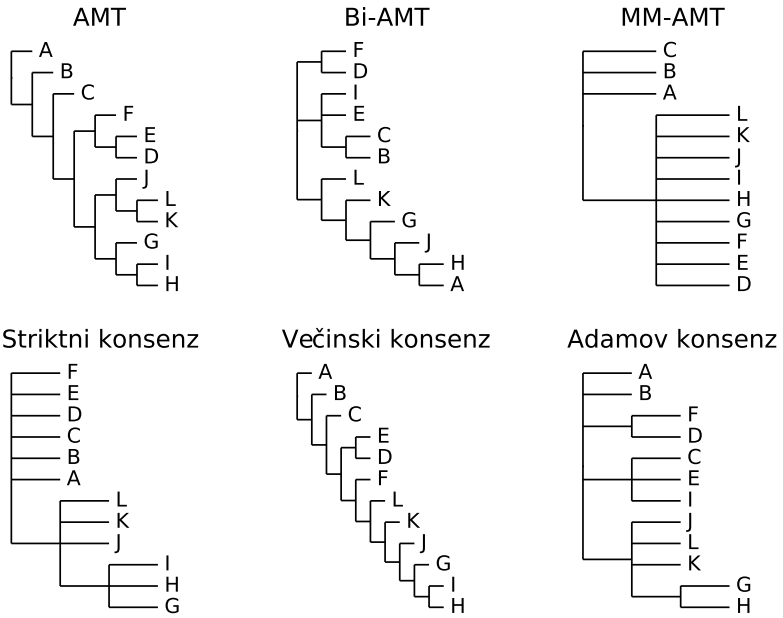
Slika 5.3: Druga vhodna množica dreves.

Rezultati konsenznih dreves iz slike 5.4 glede na vhodna drevesa iz slike 5.3 so prikazani v tabeli 5.3. Ponovno sta asimetrično srednje drevo ter večinsko konsenzno drevo polno razrešeni. Če ti dve drevesi medsebojno primerjamo še po podobnosti vhodnim drevesom, se je najbolje odrezalo asimetrično srednje drevo. Nekoliko slabše razrešeno je drevo Bi-AMT, ki je sicer glede na RF metriko nekoliko bolj podobno vhodnim drevesom kot asimetrično srednje drevo. Drevo MM-AMT je sicer po podobnosti še boljše, vendar je manj podobno in za povrh še manj razrešeno kot striktno konsen-

	Res (T)	RF (T_1)	RF (T_2)	RF (T_3)	RF (vsota)
AMT	23	12	9	7	28
Bi-AMT	21	9	10	8	27
MM-AMT	14	10	9	7	26
Striktni konsenz	15	7	8	8	23
Večinski konsenz	23	10	11	9	30
Adamov konsenz	17	13	10	8	31

Tabela 5.2: Razrešenost konsenznih dreves in njihove RF vrednosti za vhodna drevesa na sliki 5.3.

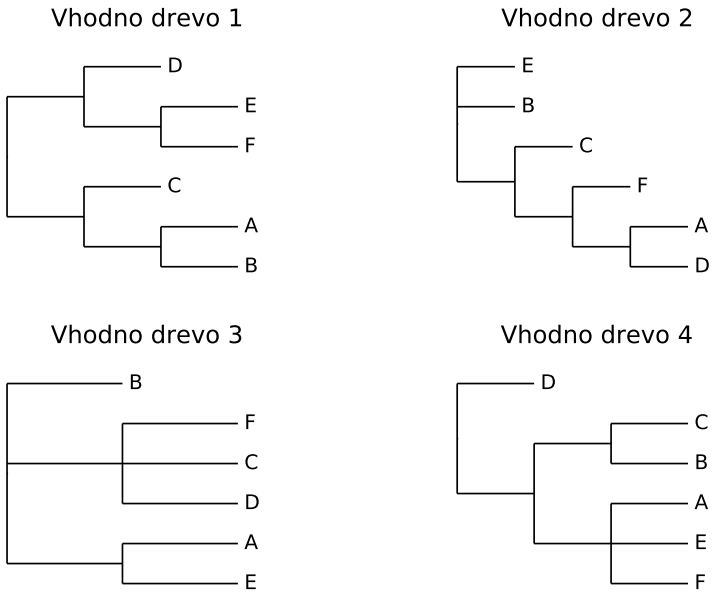
zno drevo. Adamovo konsenzno drevo je kljub slabši razrešenosti najmanj podobno vhodnim drevesom.



Slika 5.4: Konsenzna drevesa, zgrajena iz množice dreves na sliki 5.3.

Ker večina kladov ni prisotnih v več kot polovici vhodnih dreves, povečini pa so medsebojno kompatibilni, je rezultat pričakovan. Slaba razrešenost drevesa MM-AMT je najverjetneje posledica majhnega števila skupnih kladov in majhnega števila vhodnih dreves.

5.4 Vhodna množica 3

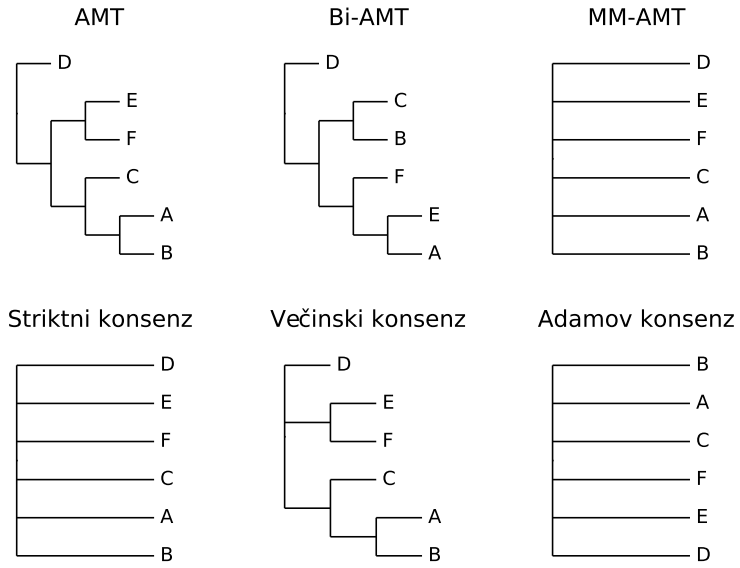


Slika 5.5: Tretja vhodna množica dreves.

Tretja vhodna množica dreves iz slike 5.4, je sestavljena tako, da je število skupnih kladov majhno, posamezni kladi ne nastopajo v več kot polovici vhodnih dreves in so večinoma medsebojno nekompatibilni. Rezultate dobljenih konseznih dreves na sliki 5.5 prikazuje tabela 5.4.

	Res (T)	RF (T_1)	RF (T_2)	RF (T_3)	RF(T_4)	RF (vsota)
AMT	11	0	2	3	1	6
Bi-AMT	11	0	2	3	1	6
MM-AMT	7	3	3	2	2	10
Striktni konsenz	7	3	3	2	2	10
Večinski konsenz	10	0	2	3	1	6
Adamov konsenz	7	3	3	2	2	10

Tabela 5.3: Razrešenost konsenznih dreves in njihove RF vrednosti za vhodna drevesa na sliki 5.5.



Slika 5.6: Konsenzna drevesa, zgrajena iz množice dreves na sliki 5.5.

Opazimo, da sta polno razrešeni le asimetrično srednje drevo in aproksimirano drevo Bi-AMT. Vsa ostala drevesa vsebujejo vsaj eno nerazrešeno vozlišče. Drevo MM-AMT je v tem primeru enako striktnemu konsenznemu drevesu in adamovem konsenznem drevesu, tako po topologiji kot glede na razrešenost in RF metriko. Preostala drevesa so glede na RF metriko sicer enakovredna, vendar ker drevo večinskega konsenza ni polno razrešeno, lahko za najboljši drevesi razglasimo asimetrično srednje drevo in Bi-AMT drevo, ki sicer nimata popolnoma enakih topologij.

Slaba razrešenost drevesa MM-AMT je ponovno posledica majhnega števila skupnih kladov in v takem primeru je aproksimacija Bi-AMT boljša. Ker posamezni kladi ne nastopajo v več kot polovici vhodnih dreves, metodama striktnega in adamovega konsenza ni uspelo popolnoma razrešiti vseh vozlišč vhodnih dreves, saj so kladi s premalo pojavitvami kaznovani. Asimetrično srednje drevo za razliko take klade vključi, če to doprinese k informiranosti drevesa.

Razlika med asimetričnim srednjim drevesom in aproksimiranim dreve-

som Bi-AMT ni tako očitna, ker je vhodna množica majhna. Drevo Bi-AMT je namreč zgrajeno le iz dveh vhodnih dreves in v kolikor bi vhodno množico povečali, bi postale razlike bolj očitne, saj bi bilo asimetrično srednje drevo za razliko od približka sposobno vključiti informacije iz večih dreves hkrati.

5.5 Primer puščavskih zelenih alg

Zadnjo vhodno množico predstavlja deset dreves s 150 taksonomskimi enotami iz zbirke primerov programa HashCS [17]. Zbirka je sicer bila zgrajena za potrebe uvrstitve devetih prej še neobjavljenih sekvenc puščavskih zelenih alg. Poleg njih vsebuje še štirinajst znanih puščavskih zelenih alg in 127 taksonomskih enot prisotnih v tekoči vodi, morski vodi ali tleh. Drevesa so bila zgrajena s pomočjo programa MrBayes z uporabo GTR evolucijskega modela. Poravnane sekvence, ki so bile vhod v metodo bayesove inference, so bile dolge 1651 baznih parov [19].

	AMT	Bi-AMT	MM-AMT	Striktni k.	Večinski k.	Adamov k.
Res(T)	298	279	242	151	284	259
RF(T_1)	238	233	206	147	220	225
RF(T_2)	260	255	204	147	232	215
RF(T_3)	252	251	218	147	244	231
RF(T_4)	258	243	190	147	248	233
RF(T_5)	248	245	208	147	234	219
RF(T_6)	258	229	206	147	232	221
RF(T_7)	266	216	216	147	242	233
RF(T_8)	256	241	212	147	242	221
RF(T_9)	250	243	196	147	232	231
RF(T_{10})	260	233	212	147	232	213
RF (vsota)	2546	2389	2068	1470	2358	2242

Tabela 5.4: Razrešenost konsenznih dreves in njihove RF vrednosti za deset dreves s 150 taksonomskimi enotami [19].

Rezultati v tabeli 5.5 kažejo, da je najbolje razrešeno asimetrično srednje

drevo. Sledita mu večinsko konsenzno drevo in aproksimirano drevo Bi-AMT. Striktno konsenzno drevo je sicer najbolj podobno vhodnim drevesom, vendar najslabše razrešeno. Opazimo, da so vsa drevesa do neke mere žrtvovala podobnost za razrešenost. V tem smislu je metoda striktnega konsenza najbolj konzervativna, metoda asimetričnega srednjega drevesa pa najbolj drzna in s tem podaja največ informacij o evolucijskih dogodkih.

5.6 Primerjava izvajalnih časov

Zaradi eksponentne časovne kompleksnosti metode asimetričnega srednjega drevesa nas zanima, kako velika drevesa lahko v doglednem času še izračunamo. Merjenje izvajalnega časa smo opravili na dvojedrnem procesorju Intel i5, pri čemer smo izrabili le eno jedro, ki teče na frekvenci 1.7 GHz.

Hitrost računanja je odvisna od dveh parametrov, in sicer števila dreves v vhodni množici in števila taksonomskih enot, ki so v drevesih zastopane.

Št. dreves	Št. taksonomskih enot	AMT [s]	Bi-AMT [s]	MM-AMT [s]
5	5	0.017	0.058	0.020
5	10	0.119	0.339	0.146
5	20	1.252	1.641	0.904
5	22	6.321	2.121	1.255
5	25	26.448	2.789	1.085
5	28	163.676	3.753	1.509
5	31	527.254	4.712	1.760
20	5	0.043	1.091	0.045
20	9	2.340	4.912	0.740
20	11	18.915	8.190	3.334
20	14	421.215	14.066	7.456
20	15	3654.519	17.084	9.444

Tabela 5.5: Časi izvajanja metode asimetričnega srednjega drevesa in obeh aproksimacijskih metod za različne velikosti vhodne množice in različnih števil taksonomskih enot.

Poglavje 6

Zaključek

Najpomembnejši rezultat dela je zagotovo prva znana implementacija konsenzne metode asimetričnega srednjega drevesa v programski paket Biopython. Čeprav je računska kompleksnost metode v splošnem eksponentna, kar je nezaželjeno, na probleme z izvajalnim časom nismo naleteli. V kolikor ta predstavlja oviro, ima uporabnik možnost izbire ene izmed aproksimacijskih metod. Pravilna izbira aproksimacijske metode je, kot smo pokazali tudi eksperimentalno, odvisna predvsem od lastnosti dreves v vhodni množici.

Kot smo pokazali na treh vhodnih množicah, asimetrično srednje drevo ni nujno tisto, ki drevesa vhodne množice povzema najbolje. Prav tako kot z izbiro aproksimacijske metode tudi tukaj velja, da je izbira konsenzne metode odvisna predvsem od predhodnega poznavanja lastnosti dreves vhodne množice. V kolikor ta vsebujejo veliko število nekompatibilnih parov poddreves, potem je izbira metode asimetričnega srednjega drevesa smotrna, v nasprotnem primeru pa lahko že večinsko ali adamovo konsenzno drevo da primerljive, če ne celo boljše rezultate.

Kar se tiče razrešenosti dreves, je v naših testnih primerih asimetrično srednje drevo vedno bilo polno razrešeno in tako ponujalo največ informacije o evolucijski zgodovini, s čimer je prekašalo vse ostale metode. Tega sicer ne moremo trditi za obe aproksimacijski metodi. Metoda z izračunom največjega ujemanja je po razrešenosti v nekaterih primerih bila primerljiva

zgolj s striktnim konsenznim drevesom, metoda z izračunom drevesa iz dveh vhodnih dreves pa je ob dobri razrešenosti običajno proizvedla drevo, ki se slabše ujema z vhodnimi drevesi.

Možnosti za izboljšave vidimo predvsem pri aproksimacijskih algoritmih. Nad drevesi, izračunanimi s pomočjo prvega aproksimacijskega algoritma, bi npr. lahko ponovno izračunali konsenzna drevesa in izbrali najboljšega. Prav tako so potrebne dodatne študije, s katerimi bi identificirali lastnosti dreves vhodne množice, za da metoda asimetričnega srednjega drevesa boljše rezultate.

Literatura

[1] C. Darwin. "Notebook B: Transmutation of species", str. 36, 1837-1838.

[2] C. Phillips, T. J. Warnow. "The asymmetric median tree - A new model for building consensus trees", *Discrete Applied Mathematics*, št. 71, str. 311–335, 1996.

[3] D. Bryant. "A classification of consensus methods for phylogenetics.", *Bioconsensus*, str. 163–185.

[4] D. Gusfeld. "Efficient algorithms for inferring evolutionary trees", *Networks*, št. 21, str. 19-28, 1991.

[5] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, 2004.

[6] J. Felsenstein. "Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach", *Journal of Molecular Evolution*, št. 17, str. 368-376, 1981.

[7] T. Y. Berger-Wolf. "Properties of compatibility and consensus sets of phylogenetic trees". *UNM Computer Science Technical Report*, TR-CS-2004-24, 2004.

[8] T. Asano, J. Jansson, K. Sadakane, R. Uehara, G. Valiente. "Faster computation of the Robinson–Foulds distance between phylogenetic networks", *Information Sciences*, št. 197, str. 77-90, 2012.

[9] T. H. Jukes, C. R. Cantor. "Evolution of protein molecules", *Mammalian protein metabolism*, št. 3, str. 21-132, 1969.

[10] M. Kimura. "A Simple Method for Estimating Evolutionary Rates of Base Substitutions Through Comparative Studies of Nucleotide Sequences", *Journal of Molecular Evolution*, št. 16, str. 111-120, 1980.

[11] M. A. Larkin, G. Blackshields, N. P. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, J. D. Thompson, T. J. Gibson, D. G. Higgins. "ClustalW and ClustalX version 2", *Bioinformatics*, št. 23, str. 2947–2948, 2007.

[12] J. Felsenstein. *PHYLIP (Phylogeny Inference Package) version 3.6*. Department of Genome Sciences, University of Washington, Seattle.

[13] S. Tavare. "Some probabilistic and statistical problems in the analysis of DNA sequences.", *Lectures on Mathematics in the Life Sciences*, št. 17, str. 57-86, 1986.

[14] P. Lemey, M. Salemi, A. Vandamme. *The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing*. Cambridge University Press, 2009.

[15] F. Ronquist, M. Teslenko, P. van der Mark, D. L. Ayres, A. Darling, S. Höhna, B. Larget, L. Liu, M. A. Suchard, J. P. Huelsenbeck. "MrBayes 3.2: Efficient Bayesian Phylogenetic Inference and Model Choice Across a Large Model Space", *Systematic Biology*, št. 61, str. 539-542, 2012.

[16] K. Tamura, G. Stecher, D. Peterson, A. Filipski, S. Kumar. "MEGA6: Molecular Evolutionary Genetics Analysis Version 6.0", *Molecular biology and evolution*, št. 30, str. 2725–2729, 2013.

[17] S. S. Sul, T. L. Williams. "An Experimental Analysis of Consensus Tree Algorithms for Large-Scale Tree Collections", *5th Intl. Symposium on Bioinformatics Research and Applications*, str. 100-111, 2009.

[18] R. A. Rossi, D. F. Gleich, A. H. Gebremedhin, M. M. Patwary. "A Fast Parallel Maximum Clique Algorithm for Large Sparse Graphs and Temporal Strong Components", *arXiv preprint 1302.6256*, 2013.

[19] L. A. Lewis, P. O. Lewis. "Unearthing the Molecular Phylodiversity of Desert Soil Green Algae (Chlorophyta)", *Systematic Biology*, št. 54, str. 936-947, 2005.

[20] Phylogenetics dostopno na:
http://en.wikipedia.org/wiki/Phylogenetics

[21] Biopython tutorial and cookbook dostopno na:
http://biopython.org/DIST/docs/tutorial/Tutorial.html

[22] Matching (Graph Theory) dostopno na:
http://en.wikipedia.org/wiki/Matching_(graph_theory)

Dodatek A

Testna programska koda

```
from Bio import Phylo
from Bio.Phylo import Consensus as cons
import dendropy
from matplotlib import pyplot

trees = list(Phylo.parse('vhodna_drevesa.tre', 'newick'))
amts = {
    'AMT': cons.amt_consensus(trees, method='no_approx'),
    'Bi-AMT': cons.amt_consensus(trees, method='bi_approx')
      ↪ ,
    'MM-AMT': cons.amt_consensus(trees, method='
      ↪ maxmatch_approx')
}
other = {
    'Strict': cons.strict_consensus(trees),
    'Majority': cons.majority_consensus(trees),
    'Adam': cons.adam_consensus(trees)
}

def _plot(nx, ny, fig, i, tree, title):
    ax = fig.add_subplot(ny, nx, i)
    fig.subplots_adjust(wspace=0.5)
    Phylo.draw(tree, do_show=False, axes=ax)
    pyplot.axis('off')
```

```
23      pyplot.title(title)
24
25  # Shrani vhodna drevesa v datoteke
26  for i, tree in enumerate(trees):
27      Phylo.write(tree, 'eval/in_%d.tre' % i, 'newick')
28
29  # Shrani konsenzna drevesa v datoteke
30  for name, tree in amts.items() + other.items():
31      tree.ladderize()
32      Phylo.write(tree, 'eval/%s.tre' % name, 'newick')
33
34  # Izrisi vhodna drevesa v datoteko input_trees.pdf
35  input_trees = pyplot.figure(0)
36  x = len(trees)
37  for i, t in enumerate(trees):
38      _plot(x, 1, input_trees, i, t, 'Vhodno drevo %d' % i)
39  pyplot.savefig('input_trees.pdf')
40
41  # Izrisi konsenzna drevesa v datoteko consensus_trees.pdf
42  output_trees = pyplot.figure(1)
43  for i, kt in enumerate(amts.items() + other.items()):
44      k, t = kt
45      _plot(3, 2, output_trees, i, t, k)
46  pyplot.savefig('consensus_trees.pdf')
47
48  # Nalozi drevesa z DendroPy
49  d_amts = {
50      k: dendropy.Tree(stream=open('eval/%s.tre' % k), schema
          ↪ ='newick')
51      for k in amts.keys()
52  }
53  d_other = {
54      k: dendropy.Tree(stream=open('eval/%s.tre' % k), schema
          ↪ ='newick')
55      for k in other.keys()
56  }
57  d_input = [
58      dendropy.Tree(stream=open('eval/in_%d.tre' % i), schema
```

```
        ↪ ='newick')
59      for i in range(0, len(trees))
60  ]
61
62  tpl = '[{0:10}] Res = {1:4} RF_sum: {2:5} RFs: {3}'
63  for k in d_amts.keys() + d_other.keys():
64      if k in d_amts:
65          t = d_amts[k]
66      else:
67          t = d_other[k]
68
69      diffs = []
70      total_diff = 0
71      edges = len(t.get_edge_set())
72      t.is_rooted = False
73      t.update_splits()
74      for in_tree in d_input:
75          if in_tree.is_rooted:
76              in_tree.is_rooted = False
77              in_tree.update_splits()
78          d = in_tree.symmetric_difference(t)
79          diffs.append(str(d))
80          total_diff += d
81
82      print tpl.format(k, edges, ', '.join(diffs), total_diff
        ↪ )
```

Slika A.1: Testna programska koda za merjenje razrešenosti konsenznih dreves in Robinson-Fouldsove metrike. Koda naloži vhodno množico dreves in iz nje izračuna konsenzna drevesa s pomočjo šestih metod. Nato vhodna drevesa in izračunana drevesa izriše v PDF datoteki ter vsako drevo shrani v datoteko v formatu Newick. S pomočjo knjižnice DendroPy nato ta drevesa naloži in nad konsenznimi drevesi izračuna razrešenost ter Robinson-Fouldsovo metriko glede na vsako vhodno drevo.

```
from Bio.Phylo.BaseTree import Tree
from Bio.Phylo.Consensus import amt_consensus as amt
import time

current_time = lambda: int(round(time.time() * 1000))


def _test(method, trees):
    s = current_time()
    _ = amt(trees, method=method)
    return current_time() - s

N_TAXONS = range(10, 50)
N_TREES = 10
N_ITERS = 10
METHOD = 'no_approx'

for n in N_TAXONS:
    total = 0.0
    trees = [Tree.randomized(n) for i in range(0, N_TREES)]
    for _ in range(0, N_ITERS):
        total += _test(METHOD, trees)
    print '[%s, %d taxons, %d tress]: %fms' % (METHOD, n,
      ↪ N_TREES, total/float(N_ITERS))
```

Slika A.2: Testna programska koda za merjenje časa izvajanja. Spremenljivka *N_TREES* določa število dreves, ki bodo naključno generirana in predstavljala vhodno množico. Nad temi drevesi nato izračunamo asimetrično srednje drevo z metodo, ki jo določa spremenljivka *METHOD*.