SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2324

Orthobalancer: aplikacija za kreiranje skupova bioloških vrsta usporedive taksonomske širine

Ivan Slijepčević

Zagreb, lipanj 2012.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da biste uklonili ovu stranicu obrišite naredbu `\izvornik`.

zahvale

SADRŽAJ

POPIS SLIKA

POPIS ALGORITAMA

1. Uvod

Biološka sličnost među vrstama oduvijek je fascinirala čovječanstvo. Potpunije razumijevanje sličnosti na molekularnoj razini moglo bi imati posljedice velikog značenja. Na primjer, mnoge evolucijske veze među vrstama se mogu zaključiti na temelju sličnosti, odnosno razlike genetskog materijala među vrstama.

Orthobalancer je mrežna aplikacija kojoj je cilj za zadani skup paralognih proteina pronaći skup ortologni proteina, odnosno proteina koji obavljaju slične funkcije u drugim vrstama. Iako postoje mnoge metode za nalaženje ortologa bazirane samo na genetskim informacijama, odnosno na DNA ili na proteinskoj sekvenci, takve metode se mogu primijeniti samo na organizme čiji su cjelokupni genomi sekvencirani u bazama podataka. Orthobalancer ovom problemu pristupa na način da za svaki od ulaznih paralognih proteina prikupi skup vrsta koje sadrže proteine dovoljno slične ulaznome proteinu, a zatim spustom kroz taksonomsko stablo živog svijeta nalazi zajednički podskup vrsta kako bi odlučio za koje proteine se može zaključiti da su ortolozi.

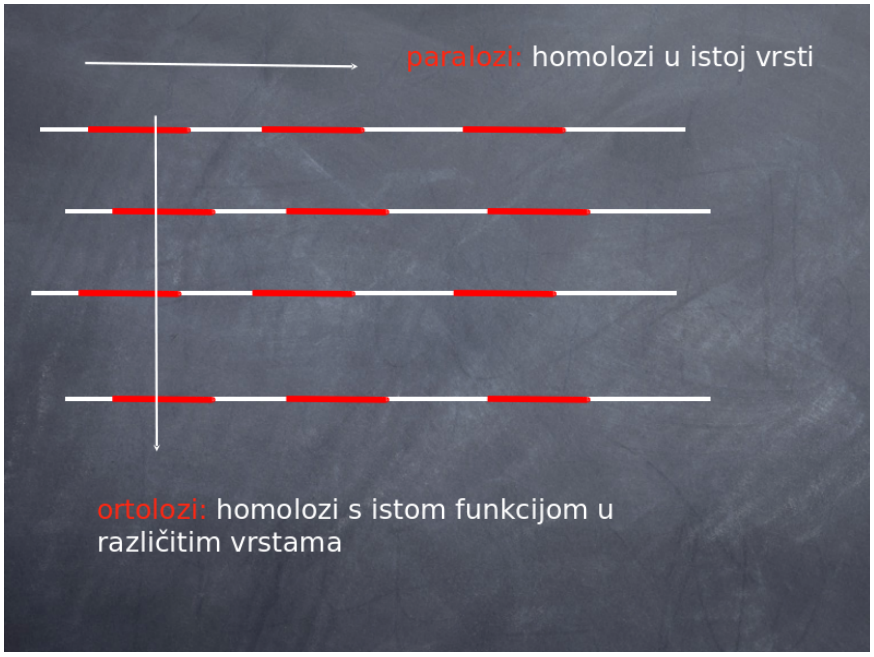
Dodatno, korisnik u aplikaciju može unijeti stupanj *zamjenjivosti* na način da određene čvorove taksonomskog stabla proglasi *zamjenjivima*. Tada, za dano podstablo, nije nužno da svi skupovi sadrže identičnu vrstu, već je dozvoljeno odstupanje pri kojem se bilo koja vrsta mogu *balansirati* kao reprezentativne vrste svog skupa za dano *zamjenjivo* podstablo.

Detaljniji biološki uvod dan je u poglavlju 2. Poglavlje 3 daje pregled o korištenim formatima podataka, bazama te ulaznim i izlaznim podacima. Kratki opis implementacije dan je u poglavlju 4. Središnji dio aplikacije — cjevovod — opisan je u poglavlju 5. Modul za nalaženje i balansiranje vrsta objašnjen je u poglavlju 6. Poglavlje 8 komentira dobivene rezultate. Zaključak je iznesen u poglavlju 9.

2. Teoretski uvod

2.1. Homologija proteina

Homologija u biološkom smislu predstavlja slične osobine među vrstama na različitim razinama organizacije života, poput organa, tkiva, stanice ili molekule. Homologne osobine uočene među jedinkama različitih vrsta obično upućuju na zajedničke pretke tih vrsta u evoluciji. Međutim, u molekularnoj biologiji termin homolog se često koristi i za naznačavanje sličnosti. bez obzira na genetsko srodstvo.[2]



Slika 2.1: Vizualni prikaz ortologni i paralognih gena na genetskim lancima raznih vrsta

Za homologne sekvence proteina kažemo da su ortologne kad su direktni potomci neke sekvence u zajedničkom pretku, bez da su prošle duplikaciju gena. Drugim riječima, ortologne sekvence se mogu naći u jedinkama različitih vrsta, a obavljaju istu funkciju u svim tim vrstama. Paralogne sekvence su homologne sekvence koje su nastale od dvije različite kopije nekog gena koji je prošao kroz proces duplikacije gena

u nekom zajedničkom evolucijskom pretku. Paralozi se mogu naći u jedinkama jedne ili više vrsta te ne obavljaju nužno identične funkcije. Slika 2.1 predočuje razliku paraloga i ortologa.

3. Podaci

Orthobalancer radi s primarnim proteinskim strukturama — sekvencama reziduuma, odnosno aminokiselina. Za zapis sekvenci koristi se standardizirani FASTA format. Orthobalancer za svoj rad koristi dvije NCBI-jeve[1] baza podataka od kojih jedna sadrži sekvence formatirane u FASTA formatu, a druga taksonomsko stablo živog svijeta.

3.1. FASTA format

FASTA je jedan od standardnih formata zapisa genetskih informacija na računalu. Format je tekstualnog oblika, a koristi se u NCBI-jevim alatima i bazama podataka, poput BLAST-a i neredundantne baze proteinskih sekvenci. Jedan zapis u datoteci s FASTA sekvencama se sastoji od zaglavlja i sekvence. Zaglavlje je jedan redak koji započinje s znakom '>' te nakon njega može imati razne informacije koje opisuju dan zapis. Zapisi proteinskih sekvenci u NCBI-jevoj neredundantnoj bazi obično imaju jedinstveni ključ sekvence, ime proteina, ime vrste u kojoj se protein nalazi, podatke o originalnoj bazi i slično. Nakon zaglavlja slijede retci koji sadrže zapisanu sekvencu gdje svaki znak predstavlja jedan reziduum aminokiseline u peptidnom lancu. Kad bi ti se retci slijepili zajedno, dobila bi se sekvenca u jednom nizu. Prazne linije nisu dopuštene.

3.2. Neredundantna baza

Neredundantna baza je baza podataka koju nudi NCBI, a sadrži prikupljene zapise iz nekoliko baza s raznih instituta u svijetu, poput GenPept, Swissprot, PIR, PDF, PDB i NCBI RefSeq. Za neredundantnu bazu se garantira da ne sadrži dvije jednake sekvence, već se one tada spajaju u jedan FASTA zapis s proširenim zaglavljem.

Neredundantnoj bazi se pristupa pomoću alata BLAST[1] i fastacmd, koji također dolazi u paketu s BLAST-om te se zato koristi oblik neredundantne baze preformatiran

[1] National Center for Biotechnology Information

u FASTA format. Veličina neredundantne baze jest 4.1GB.

3.3. Taksonomsko stablo živog svijeta

Taxonomy baza, odnosno baza taksonomskog stabla živog svijeta dostupna je u obliku nekoliko tekstualnih datoteka, od kojih svaka predstavlja ispis pojedine relacije iz baze podataka. Najbitnije, koje se koriste u Orthobalanceru su *nodes.dmp* i *names.dmp*. Za svaki čvor stabla, *nodes.dmp* sadrži identifikacijski broj čvora, identifikacijski broj roditelja te niz dodatnih informacija, poput ranga čvora u stablu (carstvo, rod, vrsta, ...). *names.dmp* za svaki čvor čuva niz raznih imena, od kojih je jedno jedinstveno, odnosno znanstveno, a ostala su prisutna za lakše raspoznavanje od strane čovjeka.

nodes.dmp sadrži nešto više od 900000 čvorova, dok *names.dmp* sadrži skoro 1300000 čvorova. Budući da se baze redovno ažuriraju, točni brojevi su podložni promjenama. *names.dmp* je veća jer se ondje nalaze i imena čvorova koji su izbrisani ili spojeni.

3.4. Ulaz

Aplikacija kao ulaz prima nekolicinu paralognih proteina u FASTA formatu. Ako korisnik posjeduje samo sekvencu proteina, može ju zadati bez FASTA zaglavlja, no u tome je slučaju dužan dati ime unesenoj sekvenci. Prije početka izvođenja svaki ulazni paralog mora imati upisano ime te to ime mora biti jedinstveno među svim paralozima.

Dodatno, korisnik može specificirati čvorove taksonomskog stabla za čija podstabla smatra da sadrže zamjenjive vrste. Ponuđen je i osnovni skup zamjenskih čvorova za koje se vjeruje da bi mogli biti od koristi korisniku.

3.5. Izlaz

Web aplikacija nakon izvođenja prikazuje tablicu balansiranih vrsta. Stupci tablice su imenovani po paralozima s ulaza. Retci su grupirani u zamjenske čvorove. Svaki redak predstavlja jedan balansirani skup vrsta. U stupcu pod pojedinim paralogom nalazi se ortologna vrsta, a lijevo od svih vrsta je zapisan čvor na kojem su vrste tog retka balansirane. Primjer tablice se može vidjeti na slici 3.1.

Također, završna stranica sadrži poveznice za preuzimanje generiranih datoteka tijekom izvođenja. Datoteke su opisane u nastavku.

Exchangeable nodes	Balanced node	CAL1	COF1
Mammalia	Ornithorhynchus anatinus	Ornithorhynchus anatinus	Ornithorhynchus anatinus
	Monodelphis domestica	Monodelphis domestica	Monodelphis domestica
	Equus caballus	Equus caballus	Equus caballus
	Canis lupus familiaris	Canis lupus familiaris	Canis lupus familiaris
	Ailuropoda melanoleuca	Ailuropoda melanoleuca	Ailuropoda melanoleuca
	Ovis aries	Ovis aries	Ovis aries
	Bos taurus	Bos taurus	Bos taurus
	Sus scrofa	Sus scrofa	Sus scrofa
	Callithrix jacchus	Callithrix jacchus	Callithrix jacchus
	Macaca fascicularis	Macaca fascicularis	Macaca fascicularis
	Macaca mulatta	Macaca mulatta	Macaca mulatta
	Nomascus leucogenys	Nomascus leucogenys	Nomascus leucogenys
	Pan troglodytes	Pan troglodytes	Pan troglodytes
	Homo sapiens	Homo sapiens	Homo sapiens
	Pongo abelii	Pongo abelii	Pongo abelii
	Primates	Otolemur garnettii	Papio anubis
	Oryctolagus cuniculus	Oryctolagus cuniculus	Oryctolagus cuniculus
	Mus musculus	Mus musculus	Mus musculus
	Rattus norvegicus	Rattus norvegicus	Rattus norvegicus
	Eutheria	Mustela putorius furo	Mus spretus
Sauropsida	Gallus gallus	Gallus gallus	Gallus gallus
	Taeniopygia guttata	Taeniopygia guttata	Taeniopygia guttata
	Squamata	Anolis carolinensis	Gekko japonicus
Actinopterygii	Salmo salar	Salmo salar	Salmo salar
	Osmerus mordax	Osmerus mordax	Osmerus mordax
	Esox lucius	Esox lucius	Esox lucius
	Epinephelus coioides	Epinephelus coioides	Epinephelus coioides
	Tetraodon nigroviridis	Tetraodon nigroviridis	Tetraodon nigroviridis
	Percomorpha	Sparus aurata	Anoplopoma fimbria
	Danio rerio	Danio rerio	Danio rerio
	Clupeocephala	Ictalurus punctatus	Oncorhynchus mykiss
	Ancestral node	**unbalanced**	
	Percomorpha	Dicentrarchus labrax	
	Clupeocephala	Ictalurus furcatus	
Amphibia	Xenopus (Silurana) tropicalis	Xenopus (Silurana) tropicalis	Xenopus (Silurana) tropicalis
	Xenopus laevis	Xenopus laevis	Xenopus laevis
unclassifiable homologues		synthetic construct	
		Saccoglossus kowalevskii	
		Trichoplax adhaerens	

Slika 3.1: Izlazna tablica s web stranice Orthobalancera

Najvažnije izlazne datoteke su one koje sadrže poravnate sekvence ortologa iz pojedine grupe ortologa. Za svaki ulazni paralog postoji datoteka imena <*ime-proteina*>.*afa*, gdje nastavak *afa* ima konotaciju poravnata FASTA (engl. *aligned fasta*). Datoteke

ovog tipa se mogu direktno preuzeti s poveznice na završnoj stranici.

Nadalje, sa završne stranice se može preuzeti i *zip* arhiva poravnatih sekvenci, čije je ime oblika <*PAR*>*_OB_fasta_*<*D*>*_*<*M*>*.zip*. <*PAR*> je ime prvog paraloga kojeg je korisnik unio, <*D*> je dan, a <*M*> je mjesec kad je upit bio pokrenut. Budući da su zaglavlja izlaznih FASTA sekvenci generirana tijekom rada Orthobalancera, ova *zip* arhiva sadrži i datoteke s dodatnim informacijama za svaku ortolognu sekvencu, imenovane <*ime-paraloga*>*_orthologues.dict*. Svaka ortologna sekvenca će imati jednu liniju u ovoj opisnoj datoteci, sa sljedećim informacijama:

- *species name* ime vrste u kojoj se ova ortologna sekvenca nalazi. Može se reći da ova sekvenca predstavlja navedenu vrstu jer između svih pronađenih sekvenci okje pripadaju navedenoj vrsti, ova sekvenca ima najveći rezultat sličnosti ulaznome paralogu.
- *score* rezultat sličnosti ove sekvence prema ulaznome paralogu, dobiven iz izlaza alata BLAST.
- *gid* jedan ili više identifikatora, odnosno FASTA ključeva ove sekvence. Ispisani su ključevi koji identificiraju samo one proteine koji se mogu naći u vrsti navedenoj u prvome stupcu.

Dodatno, ova *zip* arhiva sadrži i datoteku *similar_species.txt*. Ovdje je u tekstualnom obliku opisan sadržaj tablice na završnoj stranici. Na početku su navedena imena paraloga. Nadalje, za svaki zamjenski čvor navedene su grupe balansiranih vrsta, popraćene s čvorom taksonomskog stabla na kojem su vrste balansirane. Ukoliko se u nekoj balansiranoj grupi za neki paralog pronašlo više ortologa, jedan se slučajnim odabirom prikazuje kao balansirani, a ostali se prikazuju kao nebalancirani (engl. *unbalanced*) na kraju ispisa pripadajućeg zamjenskog čvora. Na kraju datoteke je grupa vrsta koje su pronađene za sve paraloge, ali se ne nalaze niti ispod jednog zamjenskog čvora.

Na poslijetku, sa završne stranice se može preuzeti i *zip* arhiva s cjelokupnim sadržajem informacija prikupljenim tijekom pojedinog upita. Imenovana je <*PAR*>*_OB_*<*D*>*_*<*M*>*.zip*, gdje je <*PAR*> ime prvog paraloga kojeg je korisnik unio, <*D*> dan, a <*M*> mjesec pokretanja upita. Osim dosad spomenutih datoteka, ova arhiva sadrži još sljedeće:

- <*ime-paraloga*>*.blast* izlaz alata BLAST za pojedini ulazni paralog.
- <*ime-paraloga*>*.fasta* sekvenca pojedinog paraloga u jednom rektu bez zaglavlja
- <*ime-paraloga*>*_fastacmd.fasta* izlaz alata *fastacmd*

– <*ime-paraloga*>*_orthologues.fasta* FASTA sekvence svih ortologa pojedinog paraloga, neporavnate. Zaglavlje je generirano tijekom rada Orthobalancera.

4. Implementacija

Aplikacija je pisana u programskom jeziku Python verzije 2.7. Aplikacija se dijeli u nekoliko zasebnih cjelina. U središtu aplikacije nalazi se cjovovod koji poziva alate poput BLAST-a [1] i fastacmd-a za komuniciranje s NCBI-jevom neredundantnom bazom, zatim dio aplikacije za odabir i balansiranje vrsta na taksonomskom stablu te alat mafft [6] za poravnanje sekvenci. Rad cjevovoda detaljno je opisan u poglavlju 5, a modul za odabir i balansiranje vrsta u poglavlju 6. Pored cjevovoda implementirana je mrežna aplikacija kao korisničko sučelje za cijeli program. Mrežna aplikacija je implementirana koristeći Flask microframework verzije 0.7, dok su operacije na klijentskoj strani implementirane u javascriptu uz korištenje biblioteke jQuery. Opis rada mrežne aplikacije dan je u poglavlju 7.

5. Cjevovod

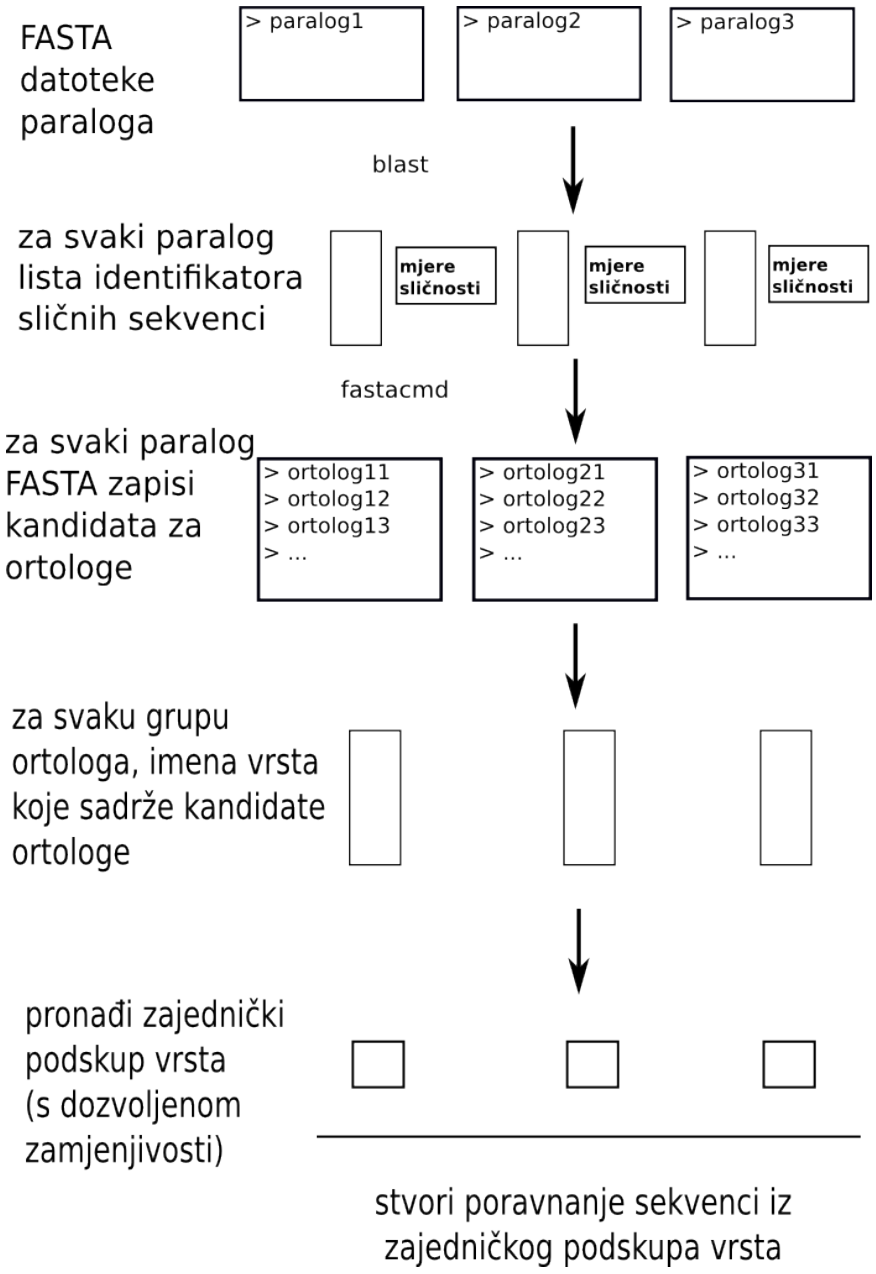
Cjevovod je arhitektonski programski obrazac u kojem podaci prolaze kroz filtre koji su postavljeni jedan za drugim. Time se simulira jedan tok koji iz ulaznih podataka transformacijom kroz filtre generira izlazne podatke. U ovome projektu cjevovodna arhitektura je samo logički kostur koji se enkapsulira unutar razreda *Pipeline*. Iako je u začetku razvoja aplikacije svaki filter bio zaseban proces, vrlo ubrzo je ustanovljeno kako većina filtera generira podatke koji su potrebni na raznim mjestima u cijeloj aplikaciji te se činilo lakše imati sve podatke u memoriji pojedinog cjevovoda. To je omogućilo da razred *Pipeline* naslijedi razred *Thread* iz modula *threading* te se može pozivati kao zasebna dretva.

Tok cjevovoda se može vidjeti na slici 5.1. Ulaz u cjevovod predstavljaju paralogni proteini u FASTA formatu koje zadaje korisnik. Ti se podaci zadaju pri stvaranju objekta *Pipeline* kako bi se mogli zapisati na disk u direktorij vezan za instancu *Pipeline-a*. Stvarni objekt kojeg prima konstruktor *Pipeline-a* je rječnik prilagođen uporabi mrežna aplikacije, što je detaljnije opisano u poglavlju 7.

Pri pokretanju cjevovoda za svaku se od unesenih sekvenci stvara objekt razreda *ProteinHolder* prilikom čega se obavljaju pozivi filtara nezavisnih za svaku pojedinu sekvencu. Prvi filter koji se koristi je alat *BLAST*[1] te je izveden kao poziv zasebnog izvršnog programa *blastall* na sljedeći način:

```
blastall -p blastp -i <ulaz-FASTA> -d <nr-baza> -m 8 -a <broj-dretvi>
```

Argument *p* s parametrom *blastp* označava programu da se koristi algoritam za uspoređivanje jedne ulazne sekvence amino kiselina s bazom proteinskih sekvenci. S argumentom *i* se zadaje put do ulazne datoteke s FASTA sekvencom. Nadalje, argument *d* prima put na disku do NCBI-jeve nereduntantne baze sekvenci koja je prethodno formatirana za pretraživanje sekvenci u FASTA formatu. Argument *m* određuje format ispisa koji generira *blastall*, a parametar 8 označava tabularni ispis bez dodatnih komentara koji je pogodan za parsiranje. Konačno, argument *a* upućuje *blastall* na broj dretvi koji treba koristiti kako bi se ubrzalo njegovo izvođenje. Broj dretvi se u Orthobalanceru može postaviti prilikom instalacije.



Slika 5.1: Protok podataka kroz cjevovod. Radije se prikazuju podaci nego filtri zbog boljeg uvida u rad cjevovoda

Izlaz koji generira *BLAST* predstavlja informacije o sekvencama sličnim ulaznoj sekvenci, odnosno informacije o potencijalnim ortolozima za ulazni paralog. Svaki redak, između ostalih, sadrži dvije bitne informacije: jedinstveni ključ sekvence u neredundantnoj bazi — gi-broj — te ocjenu sličnosti ulaznome paralogu. Nakon parsiranja tog izlaza ocjene sličnosti se spremaju za kasniju upotrebu, a gi-brojevi se zapisuju u privremenu datoteku za sljedeći korak u cjevovodu.

Sljedeći filter je izvršni program *fastacmd* koji za svaki gi-broj pronalazi i ispi-

suje sekvencu u FASTA formatu, također koristeći NCBI-jevu neredundantnu bazu. Program se poziva ovako:

```
fastacmd -i <ulaz> -d <nr-baza>
```

Budući da program *fastacmd* dolazi u paketu zajedno s *BLAST* alatima, argumenti imaju sličnu konotaciju kao i za *blastall*: s argumentom *i* se zadaje ulazna datoteka, a s *d* put do neredundantne baze.

Izlaz *fastacmd*-a se parsira pomoću razreda FastaRecord. Svaka dobivena sekvenca kao kandidat za ortologa dobiva instancu razreda FastaRecord u kojoj je sadržana sekvenca te sve bitne informacije iz zaglavlja pojedinog FASTA zapisa. Dodatno, instanci FastaRecord se pridružuje ocjena sličnosti sekvence koju opisuje prema ulaznom paralogu. Kako bi bilo lakše objasniti što je preuzeto iz zaglavlja pojedinog FASTA zapisa, bitno je imati na umu strukturu neredundantne baze što je objašnjenu u poglavlju 3. Naime, ako je za nekoliko proteina zabilježeno da imaju identičnu sekvencu, tada će zaglavlje takve FASTA sekvence u neredundantnoj bazi sadržavati spojene podatke o navedenim proteinima. Zato objekt razreda FastaRecord sadrži listu elemenata zaglavlja, gdje se svaki od tih elemenata opisuje n-torkom (gi-broj elementa, ime vrste, ime proteina, zastavica: najbolja sekvenca za ovaj gi-broj). Iako je poznato da neredundantna baza sadrži sekvence sakupljene iz raznih baza što implicira činjenicu da zaglavlja pojedinih sekvenci ne moraju imati identičan oblik, uočeno je određeno pravilo po kojem *fastacmd* ispisuje sekvence te se ono koristi kao heuristika ugrađena u parser. Pretpostavljeni oblik pojedinog elementa zaglavlja je sljedeći:

```
>gi|"gi-broj"|nebitne-informacije IME PROTEINA
[IME VRSTE]nebitna-informacija.
```

Na primjer:

```
>gi|344243907|gb|EGW00011.1| Cofilin-1 [Cricetulus griseus]
```

Na kraju obrade podataka za pojedini paralog s ulaza prolazi se listom svih objekata FastaRecord te se za svaku pronađenu vrstu odabire sekvenca s najboljom ocjenom sličnosti. Time nastaje jedan podskup sekvenci za koje možemo reći da predstavljaju grupu ortologa za dani paralog.

Nakon što je svaki paralog opisan jednim objektom razreda *ProteinHolder* potrebno je odrediti postoji li koji gi-broj koji se može pronaći u grupama ortologa različitih paraloga. Donesena je odluka kako to svojstvo nije poželjno jer jedan te isti protein ne želimo imati kao predstavnika neke vrste u različitim grupama ortologa. Za potrebe ove funkcionalnosti dodani su razredi *BestScore* i *BestScoreCollection*. Razred *BestScore* sadrži informaciju koja upućuje u kojoj se grupi ortologa, na kojem FASTA zapisu te s kojim elementom zaglavlja FASTA zapisa nalazi najbolje ocijenjena sekvenca za dani gi-broj identifikator. Razred *BestScoreCollection* služi kao sučelje za korištenje rječnika najboljih sekvenci, a nudi metode za ažuriranje rječnika kandidatima za najbolje sekvence te metodu za dohvat trenutno postavljene najbolje instance razreda *BestScore*. Skica algoritma za pronalazak najbolje ocijenjenih dana je algoritmom 1. Nakon provedbe algoritma nepoželjni proteini imaju spuštenu zastavicu najbolje sekvence za svoj gi-broj te se ne razmatraju u nastavku programa.

Ovdje bi se još mogla dodati funkcionalnost kojom bi se za vrstu kojoj je protein odbačen pronašao sljedeći najbolji nezauzeti protein, no to nije razmatrano. Za takve slučajeve ne može se sa sigurnošću

Algorithm 1 Nalaženje najbolje ocijenjenih proteina

```
1: function ProteinHolder::izračunajNajOcjene
2:     ocjene ← BestScoreCollection()
3:     for all fastaRecord ∈ this.records do
4:         for all element ∈ fastaRecord.elementiZaglavlja do
5:             ocjena ← BestScore(element)
6:             ocjene.auriraj(ocjena, fastaRecord.ocjena)
7:         end for
8:     end for
9:     return ocjene
10: end function
11: function ProteinHolder::propagiraj(najOcjene)
12:     for all fastaRecord ∈ this.records do
13:         for all element ∈ fastaRecord.elementiZaglavlja do
14:             najOcjena ← najOcjene.dohvati(element['gid'])
15:             if ¬najOcjena.provjeriElement(element) then
16:                 element['zastavicaNajbolji'] ← false
17:             end if
18:         end for
19:     end for
20: end function
21: function pronađiNajOcjene(proteinHolders)        ▷ funkcija je isječak iz cjevovoda
22:     najOcjene ← newBestScoreCollection()
23:     for all protein ∈ proteinHolders do
24:         ocjene ← protein.izraunajNajOcjene()
25:         najOcjene.aurirajKolekcijom(ocjene)
26:     end for
27:     for all protein ∈ proteinHolders do        ▷ propagacija najboljih ocjena
28:         protein.propagiraj(najOcjene)
29:     end for
30: end function
```

reći jesu li kodirajući geni doista ortologni ili se u nekoj roditeljskoj vrsti gen duplicirao te se s takvim nejednoznačnostima aplikacija ne bavi.

Najbitniji korak u cjevovodu je pronalaženje i balansiranje zajedničkih vrsta na stablu taksonomije

živog svijeta. Ovaj korak je detaljno opisan u poglavlju 6. Ovom se filtru kao ulaz predaju kompletni opisnici proteina, odnosno objekti razreda *ProteinHolder* iz kojih sam filter povlači sve potrebne informacije. Nakon što završi obrada, izlaz filtra je predočen višeslojnim rječnikom koji razdvaja podatke po sljedećim slojevima: grupe ortologa, grupe zamjenskih čvorova u taksonomskom stablu, grupe balansiranih čvorova u taksonomskom stablu te su zadnje vrijednosti balansirane vrste.

Zadnji korak u cjevovodu prije zapisivanja svih skupljenih podataka na disk je poravnanje sekvenci dobivenih kao izlaz programa *fastacmd*. Za poravnanje se koristi alat *mafft*[6] te mu je ulaz datoteka sa zapisanim sekvencama u FASTA formatu, a izlaz datoteka s istim, ali poravnatim sekvencama. *mafft* se poziva na sljedeći način:

```
mafft --auto <ulaz> > <izlaz>
```

Argument –auto upućuje *mafft* da automatski odabere najbolju strategiju poravnavanja sekvenci, uzimajući u obzir veličinu podataka. Izlaz se direktno preusmjerava u novu datoteku na disk jer poravnate sekvence nisu potrebne u memoriji.

6. Nalaženje zajedničkih vrsta

Pronalazak zajedničkih vrsta najbitniji je modul Orthobalancera. U suštini, ovaj modul nije logički vezan za nalaženje ortologa, no za potrebe Ortobalancera je svojim sučeljima prilagođen njegovom cjevovodu. Nalaženje zajedničkih vrsta među skupovima ulaznih vrsta predstavlja jednu novu dimenziju u pronalasku ortolognih proteina. Konvencionalno traženja ortologa [3, 8] jest kvalitetnije jer se pretraživanje odvija na razini sekvence, ali je moguće samo za genome koji su u potpunosti istraženi i zapisani u baze podataka.[5] Podizanje potrage za ortolozima na razinu vrsta omogućava da se ortolozi pronađu i među vrstama čiji genomi nisu još zabilježeni.

Ovaj modul za svoj rad koristi podatke iz NCBI-jeve *Taxonomy* baze. Koristi se čvorovi taksonomskog stabla živog svijeta i znanstvena imena pridijeljena čvorovima. Čvorovi su identificirani svojim jedinstvenim identifikatorima *tax_id*, a svaki čvor ima pridruženo jedno ili više korištenih imena. Samo jedno od tih imena je označeno kao znanstveno i također je jedinstveno za svaki čvor, uz neke iznimke poput sintetički stvorenih vrsta koje dijele znanstveno ime *Synthetic construct*. Datoteke koje sadrže podatke iz ovih baza su prilično velike te njihovo učitavanje, prilagođavanje i korištenje najviše utječe na trajanje izvođenja programa.

Na početku rada modula svi podaci se prilagođavaju za potrebe modula. Iz cjevovoda se od svake instance *ProteinHolder*-a uzimaju zabilježene vrste, a iz popisa zamjenjivih čvorova koje zadaje korisnik se prikupljaju svi zadani čvorovi. Svim prikupljenim imenima se tada pridružuje *tax_id* iz baze.

Nakon toga se gradi taksonomsko stablo u memoriji. Stablo je izvedeno kao veliki rječnik kojem za ključeve koristi *tax_id*, a svaki čvor je objekt razreda *Node*. Razred *Node* sadrži sljedeće bitne podatke: *tax_id* roditeljskog čvora, listu djece, brojač za svaku grupu ortologa, ukupni brojač grupa, listu zamjenskih roditeljskih čvorova te zastavicu je li balansiran. Brojači se inicijaliziraju na nulu, a kasnije tijekom algoritma će koristiti kao broj vrsta pojedine grupe ortologa koje se nalaze pod danim čvorom. Ukupni brojač grupa govori koliko ortolognih grupa ima svoga predstavnika pod nekim čvorom. Lista zamjenskih roditeljskih čvorova se inicijalizira na praznu listu, a bit će popunjena svim čvorovima koji su od korisnika označeni kao zamjenski te se nalaze iznad trenutno razmatranog čvora. Zastavica za balansiranje koristi se kasnije tijekom postupka balansiranja podstabala ispod zamjenskih čvorova. Algoritam je opisan u nastavku, a dan je i njegov pseudokod 2.

Sljedeći koraci predstavljaju inicijalizaciju stabla. Najprije se za svaku ortolognu grupu prolazi kroz sve vrste. Za svaku vrstu pronalazi se njen čvor, odnosno list u stablu. Brojač tog lista za trenutnu ortolognu grupu se inicijalizira na 1. Istovremeno se ukupni brojač grupa u tome listu postavi na 1 te se pozove metoda koja propagira ukupni brojač grupa sve do korijena stabla, pazeći da ukupni brojač niti u jednom čvoru ne bude više od jednom povećan za neku grupu ortologa.

Nakon toga treba inicijalizirati zamjenske čvorove i pripadna podstabla. Za svaki zamjenski čvor poziva se rekurzivna funkcija koja se spušta do listova i svakome čvoru u njegovu listu zamjenskih

roditeljskih čvorova dodaje *tax_id* zamjenskog čvora nad kojim je rekurzija pozvana. Time svaki čvor sadrži informaciju kojim zamjenskim podstablima pripada.

Algorithm 2 Obilazak stabla — balansiranje vrsta

algoritam obilaska počinje funkcijom $obidiStablo()$
$ULAZ$: korijenski čvor stabla; n skupova vrsta
$IZLAZ$: višeslojni rječnik $izlaz$ koji za svaki balansirani čvor sadrži listu vrsta predstavnica iz svake grupe g
function BALANSIRAJ(cvor, grupe)
    $izlaz \leftarrow$ new $izlaz$
    for all $dijete \in cvor.djeca$ **do**
        $izlaz \leftarrow izlaz \cup balansiraj(dijete, grupe)$
        if $\neg dijete.balansiran$ **then**
            for all $g \in grupe$ **do**
                $cvor.brojac[g] \leftarrow cvor.brojac[g] + dijete.brojac[g]$
            end for
        end if
    end for
    if $cvor.brojac[g] > 0 \; \forall g \in grupe$ **then**
        $cvor.balansiran \leftarrow \top$
        if $\exists k \forall g \in grupe(cvor.brojac[g] = k)$ **then**
            $A \leftarrow$ skup svih pripadajućih vrsta pod cvorom $cvor$
            $izlaz.pridruzi(A)$
        else
            $n \leftarrow \min\{brojac[g] \in cvor\}$
            $R \leftarrow \forall g \in grupe$ odaberi n slučajnih vrsta pod cvorom $cvor$
            $izlaz.pridruzi(R)$
        end if
    end if
    return $izlaz$
end function

Slijedi algoritam 2. Algoritam rekurzivno prolazi stablom, polazeći od listova. U zamjenskim podstablima balansira grupe ortologa na sljedeći način. Nakon što se obiđu listovi, svaki čvor, za svaku od grupa ortologa, postavlja svoj brojač za tu grupu na vrijednost zbroja brojača svoje djece, pri čemu se od djece ne razmatraju već balansirani čvorovi. U slučaju da nijedan brojač u čvoru ne ostane na vrijednosti 0, čvor se proglašava balansiran. Drugim riječima, u svakoj grupi ortologa postoji barem jedna vrsta ispod toga čvora te se one mogu ispisati kao uparene ili zamjenjive vrste. U idealnom, ali i najčešćem

```
27: function OBIDISTABLO(cvor, grupe)
28:     if cvor.zamjenski() then        ▷ vraća ⊤ ako je čvor u zamjenskom podstablu
29:         return balansiraj(cvor, grupe)
30:     else
31:         if cvor.list() then                          ▷ vraća ⊤ ako je čvor list
32:             for all g ∈ grupe do
33:                 izlaz[g]['nezamjenski'] ← cvor.ime
34:             end for
35:         else
36:             izlaz ← new izlaz
37:             for all dijete ∈ cvor.djeca do
38:                 if dijete.ukupniBrojac = cvor.ukupniBrojac then
39:                     izlaz ← izlaz ∪ obidiStablo(dijete, grupe)
40:                 end if
41:             end for
42:         end if
43:     end if
44:     return izlaz
45: end function
```

slučaju, desit će se da je vrsta prisutna u svim grupama ortologa te će tada sam čvor promatrane vrste biti proglašen balansiranim i roditelji ga zbog oga neće razmatrati. Za primjenu nalaženja ortolognih proteina to bi značilo da neka vrsta, na primjer Homo sapiens, sadrži ortologni protein svakome od ulaznih paraloga. Algoritam je tako osmišljen kako bi se balansirale što srodnije vrste. Što se tiče dijelova stabla koji nisu pod niti jednim zamjenskim čvorom, algoritam ne provodi postupak balansiranja te se u izlaz dodaju jedino one vrste koje su prisutne u svim grupama ortologa. Izlaz ovog algoritma vraća se pozivatelju, odnosno cjevovodu.

7. Mrežna aplikacija

Korisničko sučelje Orthobalancera ostvareno je kao mrežna aplikacija implementirana također u Pythonu koristeći Flask microframework. Flask je izgrađen na Werkzeug WSGI[1] mrežnom alatu i Jinja2 sustavu predložaka. Flask je zvan *microframeworkom* jer svoju jezgru drži jednostavnom, ali proširivom. Flask također nudi određene konvencije koje pojednostavljuju izgradnju mrežne aplikacije. Osim Flaska, za razvoj aplikacije korišten je javascript uz većinu koda napisanu pomoću biblioteke jQuery. Osim za asinkronu komunikaciju s poslužiteljem pomoću AJAX-a[2], javascript se najviše koristio za postizanje dinamike te robusnosti provjere i sinkronizacije ulaznih podataka.

HTML[3] stranice koje aplikacija generira bazirane su na Jinja2 predlošcima. Jedno od njihovih pogodnih svojstava je nasljeđivanje predložaka. Time se omogućuje da postoji jedan temeljni predložak koji sadrži osnovne elemente svake stranice, a ostali samo nasljeđuju te elemente i popunjavaju potrebni sadržaj. HTML predlošci Orthobalancera su sljedeći:

- *layout.html* definira raspored svih elemenata te ga svi ostali predlošci nasljeđuju.
- *input.html* prihvaća ulazne podatke od korisnika. Detaljniji opis mogućnosti koje su ostvarene za ulazne podatke se nalazi u odjeljku. 7.1
- *manipulate_exchangeable.html* je stranica koja se prikazuje samo kada korisnik želi unijeti vlastite zamjenske čvorove taksonomskog stabla. Stranica se otvara s ponuđenim pretpostavljenim skupom zamjenskih čvorova.
- *execution.html* prikazuje trenutno stanje izvođenja, asinkrono komunicirajući s poslužiteljem, kako bi korisnik imao jasniju predodžbu o zadacima koje poslužitelj obavlja te o trajanju izvođenja. Jednom kada stranica za ispis primi poruku o kraju izvođenja, automatski se preusmjerava na URL **'/output'**.
- *output.html* prikazuje izlaz aplikacije korisniku. Prikazani izlaz je tablica na slici 3.1. Osim tablice, nude se poveznice na niz datoteka za preuzimanje. Izlazne datoteke su opisane u odjeljku 3.5.
- *error.html* se prikazuje ukoliko na poslužitelju dođe do pogreške.

Poslužitelj Orthobalancera se može zamisliti kao stroj stanja kojemu su stanja predočena URL-om[4]. Flask radi tako da veže URL na jednu funkciju Pythona[5] koristeći mehanizam dekoriranja [7] funkcije

[1] Web Server Gateway Interface [4]
[2] Asynchronous JavaScript and XML
[3] HyperText Markup Language
[4] unified resource locator
[5] *view* funkcija

kojeg nudi Python. Na taj se način uz svaki URL veže određeni posao koji treba obaviti. Bitniji URL-ovi su sljedeći:

- **'/'** ulazna točka za pokretanje novog upita. Upit dobiva jedinstveni identifikacijski broj te se za njega inicijaliziraju određeni podaci na poslužitelju kao sjednički podaci. Automatski se preusmjerava na **/start**. Od ove točke na dalje svaki URL ima identifikacijski broj upita u svome parametru, što nije posebno naznačeno. Takav je način prenošenja identifikacijskog broja odabran kako bi korisnik samo pomoću URL-a mogao dohvatiti svoje rezultate.
- **'/start'** pozvan za inicijalizirani upit. Generira i vraća HTML stranicu baziranu na predlošku *input.html*.
- **'/finish_input'** poziva se tek u trenutku kad su na klijentskoj strani svi ulazni podaci pravilno uneseni. Podaci su tada već spremljeni u sjedničkim varijablama te se može stvoriti instanca razreda *Pipeline*, odnosno nova dretva cjevovoda. Ovisno o korisnikovom odabiru na stranici *input.html*, sljedeće stanje izvođenja može biti na URL-u **'/get_exchangeable'** ako korisnik želi zadati svoje zamjenske čvorove, odnosno na URL-u **'/preparations'** za pokretanje cjevovoda s predloženim zamjenskim čvorovima.
- **'/preparations'** čita sadržaj datoteke s predloženim zamjenskim čvorovima, sprema ih u argumente sjednice te se preusmjerava na URL **'/executing'**.
- **'/get_exchangeable'** generira stranicu baziranu na predlošku *manipulate_exchangeable.html* s podacima iz datoteke s predloženim zamjenskim čvorovima.
- **'/save_exchangeable'** sprema korisnikove zamjenske čvorove u argumente sjednice te se preusmjerava na URL **'/executing'**.
- **'/executing'** pokreće rad dretve cjevovoda te generira stranicu baziranu na predlošku *execution.html*.
- **'/output'** generira stranicu iz predloška *output.html* predajući joj sve potrebne informacije dohvaćene iz datoteka koje je stvorio izlaz cjevovoda.

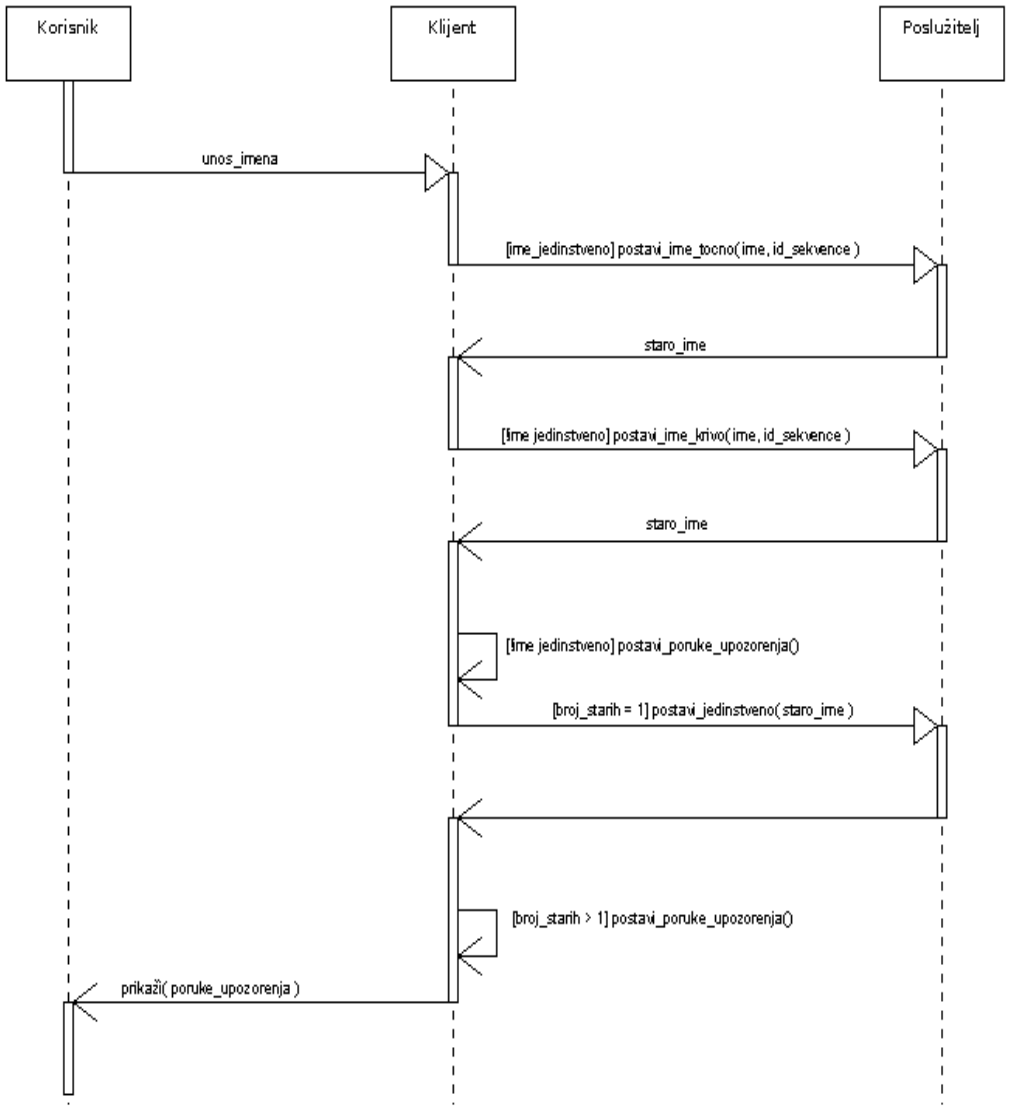
Vrijeme izvođenja posla u cjevovodu može potrajati nekoliko minuta te je zato bilo potrebno osmisliti način kako predočiti korisniku napredak izvođenja u nekom trenutku. Rješenje je osmišljeno na način da cjevovod svoj napredak zapisuje u *log* datoteke, a klijent asinkrono traži od poslužitelja sadržaj tih datoteka svakih nekoliko sekundi. Za odvajanje funkcionalnosti ispisa stanja cjevovoda u *log* datoteke iskorišten je Pythonov mehanizam dekoriranja funkcija. Implementirani su razredi *beforeMessage* i *afterMessage* koji imaju funkcionalnost dekoratora. Oba dekoratora imaju argument — ključ rječnika koji sadrži konkretan tekst za ispis u *log* datoteke. *beforeMessage* dekorira funkciju tako da obavlja ispis prije poziva dekorirane funkcije, a *afterMessage* nakon izvršavanja dekorirane funkcije. Oba razreda dekoratora nasljeđuju temeljni dekorator *messagingDecorators* koji sadrži implementaciju samog zapisivanja statusa u *log* datoteke, a izvedeni razredi se samo brinu o načinu dekoriranja.

7.1. Sinkronizacija ulaza

Jedan od nužnih preduvjeta za uspješan rad aplikacije jest pažljivo preuzimanje ulaznih podataka uz robusne provjere. S druge strane bitno je omogućiti raznolike načine unošenja podataka za lakše rukovanje aplikacijom. Orthobalancer na ulazu očekuje dvije ili više proteinskih sekvenci od kojih će svaka

sadržavati različito ime. Elemente za unos podataka pojedinog proteina na stranici moguće je dodavati dinamički. Sam FASTA zapis se može izravno upisati u površinu za unos teksta, ili se može predati datoteka sa sadržajem FASTA zapisa.

Inicijalna ideja bila je sve provjere implementirati na strani klijenta pomoću javascripta, no budući da javascript nije pogodan za otvaranje i čitanje sadržaja datoteka odlučeno je da se podaci čuvaju na poslužitelju. Zbog toga je potrebno prilikom svake promjene sadržaja bilo kojeg elementa unosa na stranici sinkronizirati podatke sačuvane u sjedničkim varijablama na poslužitelju s trenutnim stanjem na stranici koju gleda korisnik. Sinkronizacije se događaju u pozadini asinkronim komuniciranjem klijenta i poslužitelja. Asinkrona komunikacija ostvarena je AJAX tehnologijom, a klijent i poslužitelj prenose podatke u JSON[6] formatu, za što Flask ima kvalitetnu podršku.



Slika 7.1: UML sekvencijski dijagram koji prikazuje slijed operacija prilikom unosa imena

[6]JavaScript Object Notation

Obrada i spremanje podataka prilikom unosa imena sekvence prikazani su UML[7] sekvencijskim dijagramom na slici 7.1. Obrada počinje automatski u pozadini netom nakon što se ime unese. Klijent najprije pretražuje imena svih vidljivih sekvenci. Ukoliko je postavljeno ime različito od svih drugih unesenih imena, AJAX pozivom se dojavljuje poslužitelju da postavi uneseno ime za tu sekvencu. Nakon što postavi ime, poslužitelj provjerava validnost svih podataka za danu sekvencu te vraća staro ime koje je bilo pridruženo sekvenci. Klijent briše eventualne poruke upozorenja za danu sekvencu jer je ime jedinstveno. U slučaju da uneseno ime nije jedinstveno, AJAX-om se dojavljuje poslužitelju uneseno ime uz naznaku da je krivo. Svim sekvencama s unesenim imenom poslužitelj miče zastavicu da su jedinstvena te pokreće validaciju njihovih podataka. Poslužitelj opet vraća staro ime sekvence s promijenjenim imenom. Također, u slučaju nejedinstvenog unesenog imena, svim HTML elementima s istim imenom se pridružuje prikladna poruka upozorenja. Nakon postavljanja imena na poslužitelju, klijent je dužan provjeriti jesu li neka od jednakih imena na stranici trenutnim unosom razriješena. Klijent dohvaća sve elemente s imenom jednakim starim imenom koje je poslužitelj vratio. Ukoliko takvih nema, ništa se ne događa jer problema niti nije bilo. Ako ih je više, na primjer 2, to znači da ih je prethodno bilo 3, ali ta se imena i dalje trebaju razriješiti tako da im se poruke upozorenja ne miču. Na poslijetku, ako je samo jedno ime pronađeno, njemu se brišu poruke upozorenja te se AJAX-om dojavljuje poslužitelju da je ono od sada jedinstveno. Poslužitelj označava ime jedinstvenim te pokreće validaciju nad tom sekvencom.

Unos sekvence korisnik može obaviti na dva načina: može napisati ili zalijepiti FASTA sekvencu u područje za unos teksta ili može *uploadati* FASTA datoteku. Ako je odabran tekstualni način, jednom kada se sekvenca upiše u površinu za unos aktivira se logika koja predaje sekvencu poslužitelju na obradu AJAX pozivom. Za slučaj kada je odabran *upload* FASTA datoteke ne može se koristiti AJAX. AJAX tehnologija je u mogućnosti slati samo čisti tekst, a javascript nije u mogućnosti otvarati datoteke odabrane za *upload*. Iz tih razloga se ovdje koristi pomalo zastarjelo i ne sasvim pouzdano, ali jedino preostalo rješenje, a to je simuliranje *submita* HTML forme i korištenje HTML *iframe* taga. *iframe* tag u svojoj osnovnoj funkcionalnosti omogućava prikaz bilo koje druge stranice unutar osnovne stranice koju je korisnik zatražio, drugim riječima *iframe* element je kao prozor koji prikazuje neku drugu stranicu. No ako *form* tagu koji sadrži sve elemente ulaznih podataka za *target* tag postavimo ime *iframea* na istoj stranici, sadržaj trenutne stranice ostati će nepromijenjen, što uključuje i sve podatke u *input* elementima koje je korisnik unio. U ovome slučaju, gdje je potrebno asinkrono prenijeti datoteku do poslužitelja, dodane su sljedeće akcije. Prilikom odabira datoteke za *upload* automatski se poziva javascript fukcija koja simulira *submit* glavne forme. Poslužitelj potom preuzima datoteku, sprema ju, poziva obradu sekvence i vraća tijelo javascript poziva kojim se označava kraj slanja datoteke poslužitelju. Minimalni uvjet koji sama datoteka mora zadovoljiti jest da ima nastavak *.fasta* ili *.txt*.

Obrada sekvence na poslužitelju prvenstveno provjerava jesu li predani podaci doista u FASTA formatu. Zapis ne mora nužno sadržavati FASTA zaglavlje jer korisnik može upisati ime koje želi, a ime je jedino što bi se uzimalo iz zaglavlja ulaznih sekvenci. Za sekvencu je bitno da ne sadrži nedozvoljene znakove koji ne predstavljaju niti jednu aminokiselinu. Sekvenca se za daljnju obradu sprema u varijable sjednice kao jedan znakovni niz velikih slova bez bjelina. Na kraju obrade pokreće se validacija nad tom sekvencom.

Sekvence na stranici moguće je dinamički dodavati i brisati. Dodavanje nove sekvence samo umeće nove HTML elemente na stranicu, a podaci za tu sekvencu se inicijaliziraju na poslužitelju prilikom pr-

[7] Unified Modeling Language

vog AJAX poziva koji sadrži tu sekvencu. Brisanje sekvence zahtijeva određene akcije. Klijent najprije sakrije obrisanu sekvencu, iako ona još uvijek posoji. Zatim pretraži i dohvati sve elemente s imenom jednakim imenu obrisane sekvence. Nakon toga, klijent AJAX pozivom dojavljuje poslužitelju da je određena sekvenca obrisana, prosljeđujući mu istovremeno niz istoimenih sekvenci. Poslužitelj označava obrisanu sekvencu neaktivnom i pokreće validaciju nad njome. Na kraju, ako je duljina niza istoimenih elemenata jednaka 2, to znači da od trenutka brisanja drugi element ima jedinstveno ime. Poslužitelj označava ime druge sekvence jedinstvenim te pokreće validaciju nad njome. Po povratku na klijenta, ako je duljina niza bila jednaka 2, miču se poruke upozorenja druge sekvence.

Validacija pojedine sekvence na poslužitelju postoji kako bi se svi podaci dane sekvence provjerili jesu li uneseni i jesu li u dozvoljenom obliku kako bi se jednostavnom provjerom zastavice 'OK' moglo ustvrditi može li dana sekvenca biti proslijeđena u cjevovod. Prilikom validacije događaju se sljedeće provjere:

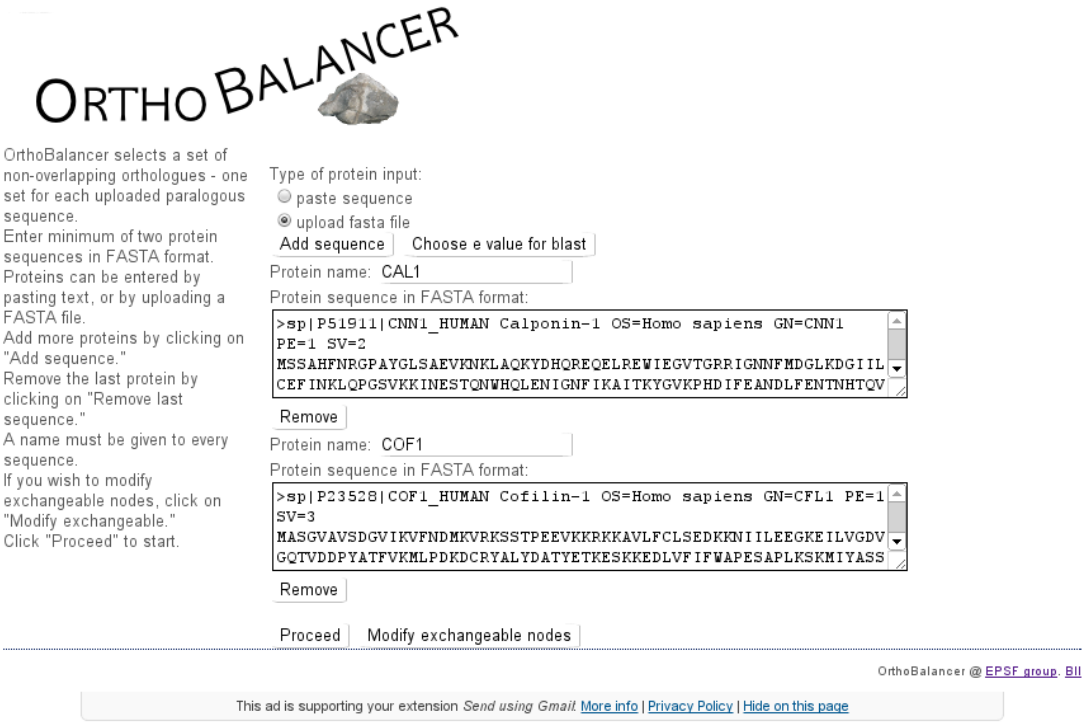
- je li sekvenca još aktivna?
- je li uneseno ime jedinstveno? Ako nije, iznimno dozvoli za slučaj u kojem ime nije uneseno, a postoji ime u FASTA zaglavlju unesene sekvence.
- je li unesena sekvenca?

Jednom kad korisnik pritisne bilo koju od tipki za nastavak, bilo da je to *Proceed* ili *Modify exchangeable nodes*, pretpostavlja se da su svi podaci već spremljeni na poslužitelju. Pritiskom na tipku pozvat će se *submit* alternativne forme koja je vezana za provjeru validnosti podataka. Forma je također vezana za svoj *iframe* za slučaj da svi podaci nisu spremni za pokretanje cjevovoda. *iframe* je ovdje potreban kako se ne bi izgubili svi podaci koje je korisnik unio. Poslužitelj ovdje samo provjerava izračunatu validnost podataka te vraća tijelo javascript poziva kojim se ispisuje ispravna poruka o grešci ili kojim se izvodi *submit* konačne forme za nastavak izvođenja.

8. Izvođenje i rezultati

Orthobalancer je javno dostupan na URL-u `http://orthobalancer.bmad.bii-sg.org`.

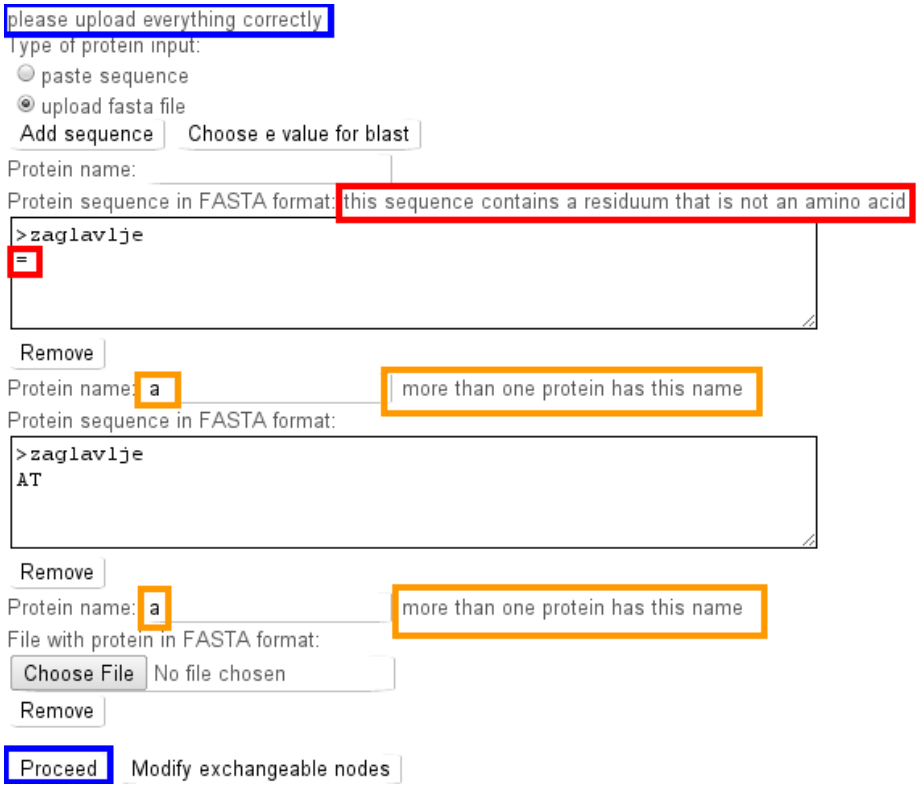
Početna stranica je prikzana na slici 8.1. Pri vrhu su smješteni HTML elementi kojima korisnik može urpavljati aplikacijom. Za odabir načina unosa FASTA sekvence pojedinog paraloga postoje radio gumbi (engl. *radio buttons*) za ručni upis, odnosno za zalijepiti sekvencu (engl. *paste sequence*) te za *upload* datoteke, a sama se sekvenca dodaje pritiskom na gumb *Add sequence*. Svaku sekvencu je moguće i naknadno ukloniti pritiskom na gumb *Remove* uz pojedinu sekvencu. S lijeve strane se nalaze jednostavne upute za korištenje.



Slika 8.1: Izgled početne stranice Orthobalancera

Sukladno s ograničenjima opisanim u odjeljku 7.1, aplikacija je sposobna dojaviti razne greške. Dojave grešaka označene su na slici 8.2. Narančastim okvirima označena je greška jedinstvenih imena. Pogrešan unos FASTA sekvence označen je crvenim okvirima. Ukoliko se pokuša pokrenuti pretraga dok su podaci pogrešni, pojavit će se upozorenje na vrhu što je označeno plavim okvirima.

Ukoliko korisnik želi modificirati skup zamjenskih čvorova, umjesto pritiska na gumb *Proceed*, što odmah započinje pretragu, korisnik može pritisnuti gumb *Modify exchangeable nodes* na početnoj stra-

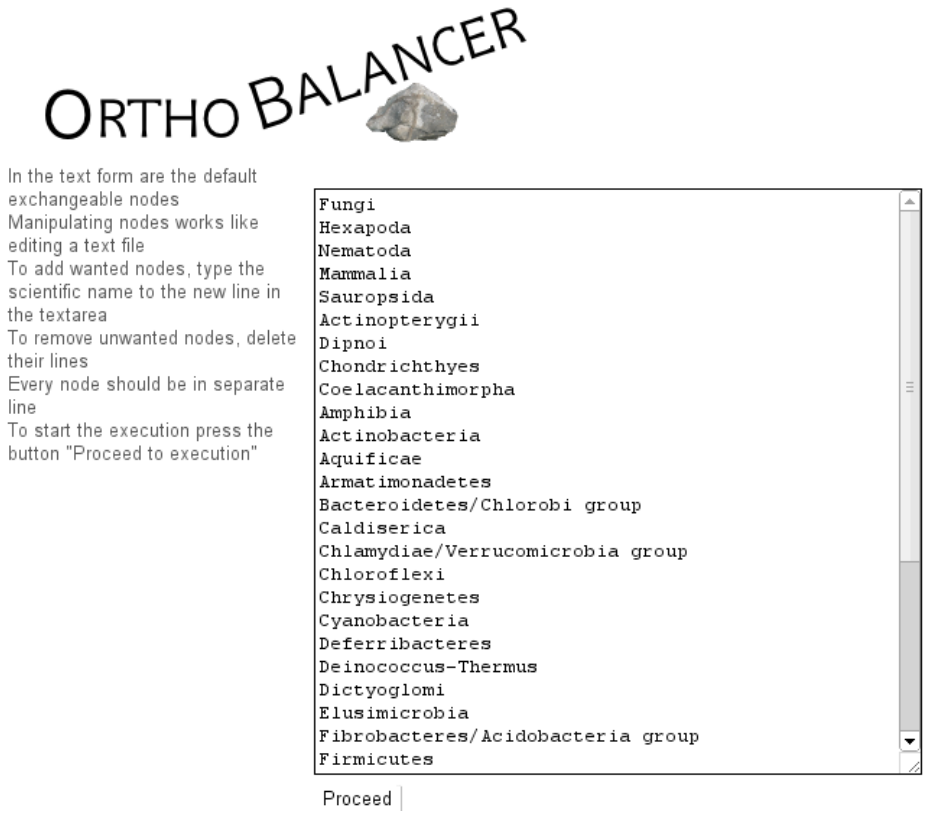


Slika 8.2: Ponašanje početne stranice Orthobalancera

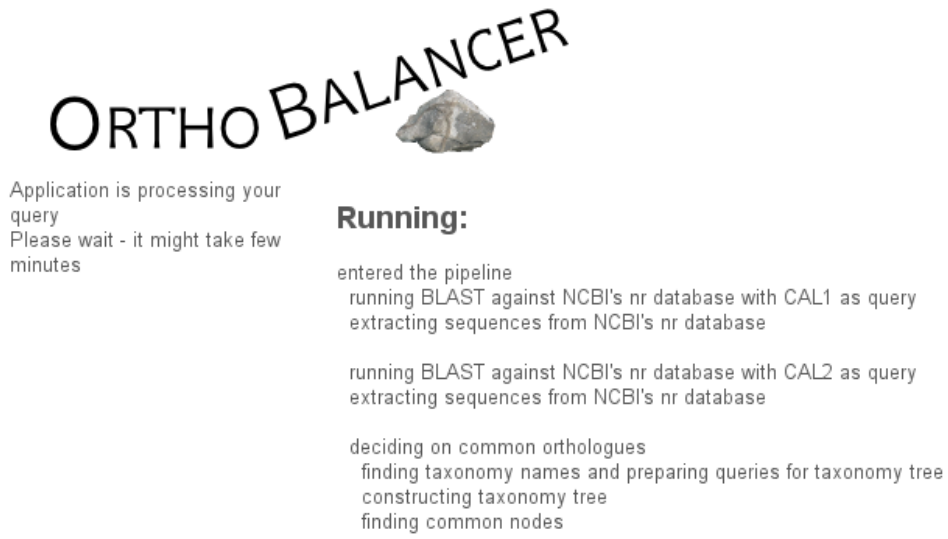
nici. Time se otvara stranica za modificiranje zamjenskih čvorova, prikazana na slici 8.3. U području za unos teksta svaki redak sadrži pojedini zamjenski čvor. Prilikom otvaranja navedene stranice u području se nalazi predloženi skup čvorova koji se koristi i u pretrazi bez korisnikovog modificiranja. Za korištenje funkcionalnosti podržane ovom stranicom preporučuje se bolje poznavanje NCBI-jeve taksonomske baze.

Orthobalancer za vrijeme svog rada komunicira s prilično velikim bazama te vrijeme izvođenja može potrajati nekoliko minuta. Kako je takvo ponašanje očekivano, status izvođenja se prikazuje na stranici prikazanoj na slici 8.4.

Završna stranica prikazana je na slici 8.5. Na vrhu su poveznice za preuzimanje izlaznih datoteka, a ispod njih se nalazi tablica balansiranih vrsta. Tablica i datoteke su detaljnije objašnjeni u odjeljku 3.5.



Slika 8.3: Stranica za modificiranje zamjenskih čvorova



Slika 8.4: Stranica za modificiranje zamjenskih čvorova

Output data

CAL1 orthologues fasta file
CAL2 orthologues fasta file
zipped output: orthologues fasta files
all files gathered during execution (including alignments; zipped)

The results will be availible to download on this link:
http://orthobalancer.bmad.bii.a-star.edu.sg/output/2012_06_27_03_02_04_638907

Exchangeable nodes	Balanced node	CAL1	CAL2
Mammalia	Monodelphis domestica	Monodelphis domestica	Monodelphis domestica
	Canis lupus familiaris	Canis lupus familiaris	Canis lupus familiaris
	Ailuropoda melanoleuca	Ailuropoda melanoleuca	Ailuropoda melanoleuca
	Bos taurus	Bos taurus	Bos taurus
	Sus scrofa	Sus scrofa	Sus scrofa
	Callithrix jacchus	Callithrix jacchus	Callithrix jacchus
	Macaca mulatta	Macaca mulatta	Macaca mulatta
	Pan troglodytes	Pan troglodytes	Pan troglodytes
	Homo sapiens	Homo sapiens	Homo sapiens
	Pongo abelii	Pongo abelii	Pongo abelii
	Mus musculus	Mus musculus	Mus musculus
Sauropsida	Gallus gallus	Gallus gallus	Gallus gallus
	Phasianidae	Coturnix coturnix	Meleagris gallopavo
	Taeniopygia guttata	Taeniopygia guttata	Taeniopygia guttata
Actinopterygii	Salmo salar	Salmo salar	Salmo salar
	Percomorpha	Sparus aurata	Tetraodon nigroviridis
	Danio rerio	Danio rerio	Danio rerio
	Ancestral node	**unbalanced**	
	Percomorpha	Epinephelus coioides	
	Percomorpha	Dicentrarchus labrax	
Amphibia	Xenopus (Silurana) tropicalis	Xenopus (Silurana) tropicalis	Xenopus (Silurana) tropicalis
	Xenopus laevis	Xenopus laevis	Xenopus laevis
unclassifiable homologues		synthetic construct	
		Schistosoma japonicum	
		Schistosoma mansoni	
		Daphnia pulex	
		Branchiostoma belcheri	

Slika 8.5: Stranica za modificiranje zamjenskih čvorova

9. Zaključak

Orthobalancer nudi jedinstven i izravan način odabira skupova ortolognih proteinskih sekvenci iz bliskih taksonomskih grana. U suštini, aplikacija radi na razini vrsta i taksonomske hijerarhije, rađe nego na razini proteinskih sekvenci. To međutim ima posljedicu da se ne može dokazati da aplikacija nalazi prave ortologe. Ipak, pristup koji koristi Orthobalancer je inovativan zbog činjenice da se na taj način pokriva znatno širi skup vrsta za nalaženje ortologa.

Za daljnje istraživanje, ideja zajedničkog podskupa vrsta među vrstama usporedive taksonomske širine bi se mogla probati primijeniti za drugačije namjene.

LITERATURA

[1] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, i D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[2] A. D. Baxevanis i B. F. F. Ouellette. *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*. John Wiley & Sons, Inc., 2002. ISBN 9780471223924.

[3] K. Bharatham, Z.H. Zhang, i I. Mihalek. Determinants, discriminants, conserved residues-a heuristic approach to detection of functional divergence in protein families. *PLoS ONE*, 6(9):e24382, 2011.

[4] P. J. Eby. Pep333 – python web server gateway interface v1.0, Prosinac 2003. URL `http://www.python.org/dev/peps/pep-0333/`.

[5] P. Flicek, M.R. Amode, D. Barrell, K. Beal, S. Brent, Y. Chen, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, et al. Ensembl 2011. *Nucleic Acids Res.*, 39:D800–806, 2011.

[6] K. Katoh, K. Kuma, H. Toh, i T. Miyata. Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res.*, 33:511–518, 2005.

[7] K. D. Smith. Pep318 – decorators for functions and methods, Lipanj 2003. URL `http://www.python.org/dev/peps/pep-0318/`.

[8] R.L. Tatusov, E.V. Koonin, i D.J. Lipman. A genomic perspective on protein families. *Science*, 278 (5338):631, 1997.

Orthobalancer: aplikacija za kreiranje skupova bioloških vrsta usporedive taksonomske širine

Sažetak

OrthoBalancer mrežna aplikacija za ulaz uzima skup od n *bona fide* paralognih — sličnih, ali kodiranih različitim genom — proteinskih sekvenci iz iste vrste. Za svaki paralog, aplikacija vraća skup najsličnijih proteina iz drugih vrsta prisutnih u ishodišnoj bazi podataka. U jezgri aplikacije je modul koji koristi taksonomsko stablo živog svijeta kako bi pronašao zajednički podskup n skupova vrsta, ali ipak dozvoljavajući razinu neizrazitosti ili "zamjenjivosti" srodnih vrsta. Stupanj zamjenjivosti, odnosno razina u taksonomskom stablu ispod koje bilo koja vrsta može poslužiti kao reprezentant pripadajućeg podstabla, se može podesiti od strane korisnika.

Ključne riječi: homologija, taksonomsko stablo živog svijeta, FASTA, neredundantna baza, mrežna aplikacija, Python, jQuery, AJAX

Orthobalancer: web application for creation of taxonomically balanced sets of orthologous protein sequences

Abstract

OrthoBalancer web application takes as an input a set of n *bona fide* paralogous — similar, but encoded by a different gene — protein sequences from the same species. For each of the paralogues, the application returns the set of the most similar proteins from other species present in the underlying database. At its core is an module that uses taxonomic tree to find a common subset of n sets of species, while allowing some fuzziness or "exchangeability" of related species. The degree of exchangeability, that is, the level in the taxonomic tree below which any species can serve as a representative of the branch, can be modified by the user.

Keywords: homology, taxonomy tree, FASTA, nonredundant database, web application, Python, jQuery, AJAX