SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT - BIOINFORMATIKA

Algoritam "Neighbor joining"

Filip Beć
Zorana Ćurković
Goran Gašić
Melita Kokot
Dino Šantl
Igor Smolkovič
Mentor: *Dr. sc. Mirjana Domazet-Lošo*
Doc. dr. sc. Mile Šikić

Zagreb, siječanj 2014.

SADRŽAJ

1. Opis algoritma

Algoritam *Neighbor joining* služi za izgradnju filogenetskog stabla. Ulaz algoritma su evolucijske udaljenosti (matrica udaljenosti čvorova). Izlaz algoritma je stablo s težinskim bridovima. Cilj algoritma izgraditi je minimalno razapinjajuće filogenetsko stablo. Algoritam nužno ne pronalazi takvo stablo ali rješenja su često minimalna razapinjajuća filogenetska stabla ili blizu toga [1]. Glavni razlog je smanjenje vremenske složenosti, jer u praksi nije izvedivo ispitivanje svih mogućih stabala. Stablo se gradi od dna prema vrhu. Algoritam je pohlepan jer za jednom sparene čvorove ne ispituje točnost tog koraka.

Algoritam se sastoji od dva dijela: uparivanje čvorova i završni korak. Algoritam kreće od matrice udaljenosti nad kojom zaključuje koja dva čvora je potrebno upariti. Kada su poznata dva čvora koja se trebaju upariti, stvara se novi čvor koji se spaja s dva odabrana. Dva odabrana čvora se brišu iz matrice udaljenosti, a novi čvor se ubaci u matricu udaljenosti. Taj postupak se izvršava $N - 3$ puta, gdje je N početni broj čvorova. Završni korak uzima zadnja tri čvora koja su ostala u matrici udaljenosti, stvara novi čvor i spaja novi čvor s tri zadnja čvora u matrici udaljenosti. Formalno algoritam izgleda ovako:

1. **Ulaz**: matrica udaljenosti
2. Na temelju trenutne matrice udaljenosti izračunaj matricu **Q**
3. U matrici **Q** pronađi najmanju vrijednost $Q(i, j)$ i pripadajuće čvorove (i, j).
4. Stvori novi čvor w i dva nova brida: (w, i) i (w, j) te izračunaj udaljenosti $d(w, i)$ i $d(w, j)$ i zapiši ih na pripadajući brid.
5. Iz matrice udaljenosti izbriši čvorove i i j, te dodaj u matricu novi čvor w - potrebno je izračunati udaljenosti do novoga čvora w
6. Ako postoji više od 3 čvora u matrici udaljenosti skoči na korak 2.

7. Stvori novi čvor w i napravi 3 brida s preostalim čvorovima u matrici udaljenosti te izračunaj i pridruži vrijednost bridovima

1.1. Matrica Q

Matrica **Q** pomoćna je matrica iz koje se određuje par čvorova koji će biti susjedni. Matrica **Q** kao kriterij uzima osim udaljenosti čvorova i i j utjecaj njihovih susjeda. Dokaz da je najmanja vrijednost matrice **Q** pripada susjednim čvorovima dan je u [1]. Matrica **Q** izračunava se na sljedeći način:

$$Q(i,j) = (r-2)d(i,j) - \sum_{k=1}^{r} d(i,k) - \sum_{k=1}^{r} d(j,k) \tag{1.1}$$

, gdje su i, j indeksi čvorova, r je trenutni broj čvorova u matrici udaljenosti. Oznake $d(i,j)$, $d(i,k)$ i $d(j,k)$ predstavljaju vrijednosti u matrici udaljenosti. Čvorovi i i j moraju biti različiti.

1.2. Računanje vrijednosti bridova

Nakon što se stvore novi bridovi u stablu potrebno je odrediti njihovu vrijednost. U svakom koraku uparivanja stvore se dva nova brida. U završnom koraku stvore se tri nova brida.

Nakon što se čvorovi i i j proglase susjednima stvara se novi čvor w. Potrebno je odrediti udaljenosti $d(i,w)$ i $d(j,w)$. Udaljenosti se određuju prema formulama:

$$d(i,w) = \frac{1}{2}d(i,j) + \frac{1}{2(r-2)}\left[\sum_{k=1}^{r} d(i,k) - \sum_{k=1}^{r} d(j,k)\right] \tag{1.2}$$

, gdje je r trenutni broj čvorova u matrici udaljenosti.

Kako vrijedi $d(i,j) = d(i,w) + d(j,w)$, iz toga sljedi:

$$d(j,w) = d(i,j) - d(i,w) \tag{1.3}$$

1.3. Računanje udaljenosti do novog čvora

Pri ubacivanju novog čvora u matricu udaljenosti potrebno je izračunati udaljenost od svih starih čvorova do novog čvora w. Udaljenost se računa prema formuli:

$$d(k, w) = \frac{1}{2}\left[d(i, k) + d(j, k) - d(i, j)\right] \tag{1.4}$$

, gdje je k bilo koji stari čvor u matrici udaljenosti. Čvorovi i i j su upravo spojeni čvorovi.

1.4. Složenost algoritma

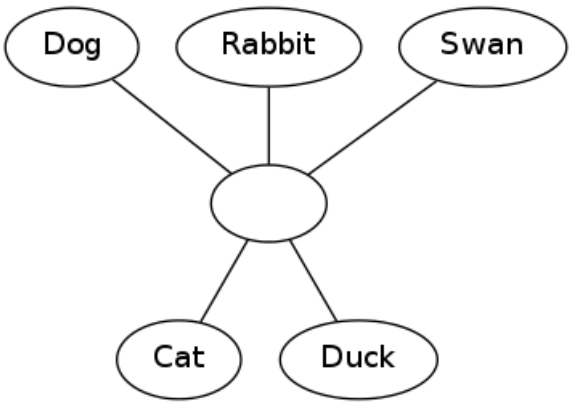
Algoritam mora izvršiti korak uparivanja $N - 3$ puta, gdje je N broj čvorova u početnoj matrici udaljenosti. U svakom koraku potrebno je izračunati matricu **Q** koja je dimenzije $r \times r$, gdje je r trenutni broj čvorova. Ako se predprocesiraju sume u prije navedenim formulama u $O(N)$ vremenu, tada za računanje matrice **Q** potrebno $O(N^2)$ vremena. Zbog toga je ukupna vremenska složenost algritma $O(N^3)$. Memorijska složenost je $O(N^2)$ jer se pamti matrica udaljenosti.

2. Primjer izvođenja

Raspolažemo skupom od 5 taksona $(\text{Dog}, \text{Cat}, \text{Rabbit}, \text{Monkey}, \text{Cow})$ te pridanim udaljenostima među njima definiranim matricom udaljenosti:

	Dog	Cat	Rabbit	Duck	Swan
Dog	0	5	17	15	13
Cat	5	0	9	19	14
Rabbit	17	9	0	20	16
Duck	15	19	20	0	12
Swan	13	14	16	12	0

Algoritam će biti proveden u $N - 3 = 2$ koraka.



Slika 2.1: Početno stablo

2.1. 1. korak

Izračunamo matricu Q te tražimo par (i, j) koji ima najmanju vrijednost:

	Dog	Cat	Rabbit	Duck	Swan
Dog	0	-82	-61	-71	-66
Cat	-82	0	-82	-56	-60
Rabbit	-61	-82	0	-68	-69
Duck	-71	-56	-68	0	**-85**
Swan	-66	-60	-69	-85	0

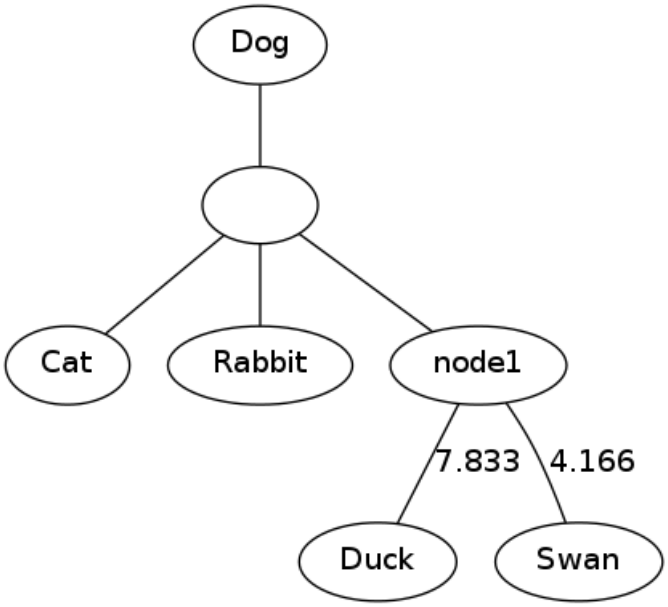
$i =$ Duck, $j =$ Swan, $Q_{min} = -85$. Taksone Duck i Swan spajamo u novi čvor node1 te računamo udaljenosti:

$d(\text{Duck}, \text{node1}) = 7.833$,

$d(\text{Swan}, \text{node1}) = 4.166$

Preostale udaljenosti $d(k, \text{node1})$ definirane su novom matricom udaljenosti :

	Dog	Cat	Rabbit	node1
Dog	0	5	17	8
Cat	5	0	9	10.5
Rabbit	17	9	0	12
node1	8	10.5	12	0



Slika 2.2: Trenutno stablo

2.2. 2. korak

Izračunamo matricu Q te tražimo par (i, j) koji ima najmanju vrijednost:

	Dog	Cat	Rabbit	node1
Dog	0	**-44.5**	-34	-44.5
Cat	-44.5	0	-44.5	-34
Rabbit	-34	-44.5	0	-44.5
node1	-44.5	-34	-44.5	0

$i = \text{Dog}, j = \text{Cat}, Q_{min} = -44.5$. Taksone Dog i Cat spajamo u novi čvor node2 te računamo udaljenosti:

$d(\text{Dog}, \text{node2}) = 3.875$,

$d(\text{Cat}, \text{node2}) = 1.125$

Preostale udaljenosti $d(k, \text{node2})$ definirane su novom matricom udaljenosti :

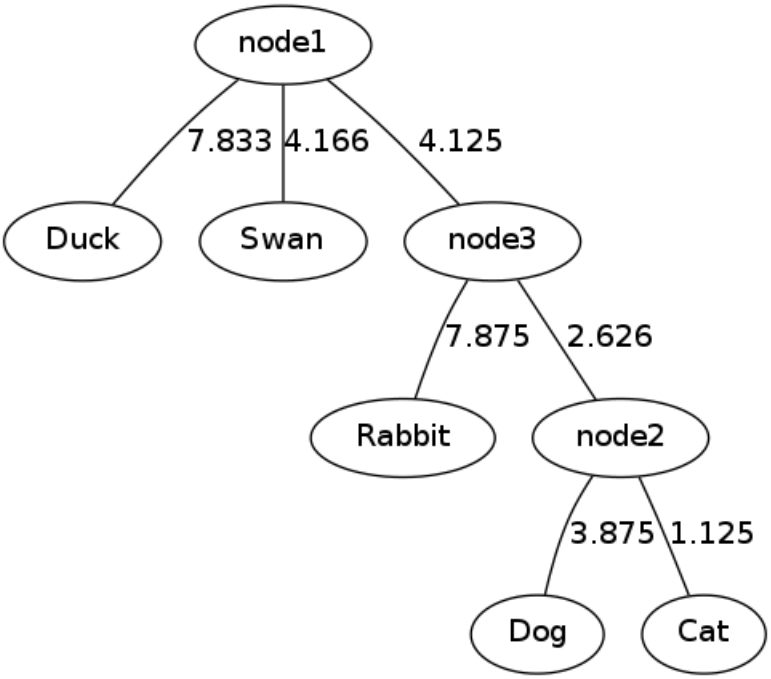
	node2	Rabbit	node1
node2	0	10.5	6.75
Rabbit	10.5	0	12
node1	6.75	12	0

Potrebno je još spojiti preostala 3 čvora. U svrhu spajanja stvaramo novi čvor node3. Poznate su nam udaljenosti $d(\text{node2}, \text{Rabbit})$, $d(\text{node2}, \text{node1})$ i $d(\text{Rabbit}, \text{node1})$. Temeljem tih udaljenosti možemo izračunati posljednja tri luka.

$d(\text{node3}, \text{node2}) = 0.5 * (d(\text{node2}, \text{node1}) + d(\text{node2}, \text{Rabbit}) - d(\text{node1}, \text{Rabbit}))$
$= 2.625$

$d(\text{node3}, \text{ node1}) = 0.5 * (d(\text{node1}, \text{Rabbit}) + d(\text{node2}, \text{Rabbit}) - d(\text{node1}, \text{node2}))$
$= 4.125$

$d(\text{node3}, \text{Rabbit}) = 0.5 * (d(\text{node1}, \text{Rabbit}) + d(\text{node1}, \text{node2}) - d(\text{node2}, \text{Rabbit}))$
= 7.875



Slika 2.3: Konačno stablo

3. Testiranje i usporedbe

4. Zaključak

5. Literatura

[1] S. N. and N. M., "The neighbor-joining method: a new method for reconstructing phylogenetic trees.," *Molecular Biology and Evolution 4*, pp. 406–425, 1987.

[2] "Phylogeny programs." `http://evolution.genetics.washington.edu/phylip/software.html`.

[3] "Neighbor joining." `http://en.wikipedia.org/wiki/Neighbor_joining`.

[4] J. Felsenstein, "Phylogeny methods, part 3 (distance methods)." `http://evolution.gs.washington.edu/gs541/2002/lecture3.pdf`, 2002.