Univerzitet u Beogradu
Matematički fakultet

Minimalna veličina ultrametrijskog stabla

Računarska inteligencija

Knežević Ana 356/2022

Ponjavić Pavle 162/2020

1. Uvod

Konstrukcija evolutivnih stabala iz matrica rastojanja je važan problem u biologiji i taksonomiji, ali većina optimizacionih problema vezanih za ovu konstrukciju su NP-teški. Kada se pretpostavi da je stopa evolucije konstantna, evolutivno stablo postaje ultrametrično stablo. Ulrametrično stablo je binarno stablo sa korenom, označenim listovima i težinskim granama, gde svaka unutrašnja tačka ima istu dužinu puta do svih listova u svom podstablu.

Problem konstrukcije ultrametričnog stabla iz nemetričke matrice rastojanja je NP-težak i ne može se aproksimirati u polinomijalnom vremenu. Problem ostaje NP-težak čak i kada je matrica rastojanja metrička, ali može se aproksimirati u razmeri 1.5(1 + [log n]).

Praktično, za manji broj vrsta moguće je izračunati optimalno stablo iscrpnom pretragom, ali zbog brzog rasta broja mogućih stabala, iscrpna pretraga postaje neizvodljiva čak i za umereno veliki broj vrsta. "Branch-and-bound" strategija pomaže da se izbegne iscrpna pretraga. Iako teoretski ne može garantovati polinomsko vreme u najgorem slučaju, uspešno je korišćena za rešavanje NP-teških problema i može pronaći optimalno ili skoro optimalno rešenje.

U ovom radu se prikazuju "branch-and-bound", genetski i algoritam iscrpne pretrage za konstrukciju minimalnog ultrametričkog stabla iz metričke ili nemetričke matrice rastojanja. Eksperimentalni rezultati pokazuju da algoritam može rešiti problem u razumnom vremenu za n ≤ 25 ako je ulaz metrička matrica, ili n ≤ 19 ako je ulaz nemetrička matrica.

2. Opis problema

Instanca: nxn matrica M sa pozitivnim celim brojevima.
Rešenje: Cilj je konstruisati podstablo sa težinskim granama T(V, E) sa n čvorova. Stablo treba da bude struktuirano tako da za bilo koja dva lista i i j, zbir težina na putanji između njih u stablu bude najmanje jednaka udaljenosti M[i,j].

Merilo: Veličina stabla, tj. $\sum_{e \epsilon E} w(e)$, gde je w(e) težina grane e.

Pronalaženje optimalnog rešenja za ovaj problem je računski zahtevno, ne može se aproksimirati unutar n^{ε} za neko $\varepsilon > 0$. U praktičnim primenama se mogu koristiti heurističke metode ili aproksimacije kako bi se došlo do skoro optimalnog rešenja s obzirom na teškoću dobijanja tačnog rešenja.

3. Rešavanje problema

Kao što je već rečeno, u ovom projektu ćemo koristiti algoritam iscrpne pretrage, genetski algoritam i “branch-and-bound”.

3.1. Algoritam iscrpne pretrage

Ceo kod možete videti ovde: brute_force.py

Ovde ćemo napraviti kratak pregled glavnih delova koda sa objašnjenjima.

Klasa *‘Tree’*:

- **‘__init__’** : Inicijalizuje prazno stablo koristeći dva rečnika: `tree` za čuvanje grana (dece) svakog čvora i `parents` za čuvanje roditelja svakog čvora.
- **‘add_edge’:** Dodaje granu između roditelja i deteta sa datom težinom. Takođe proverava da li grana već postoji.
- **‘calculate_total_weight’**: Izračunava ukupnu težinu stabla tako što sumira težine svih grana.
- **‘find_path_weight’**: Izračunava ukupnu težinu stabla tako što sumira težine svih grana.
- **‘generate_all_combinations’:** Generiše sva moguće podstabla kombinovanjem grana. Proverava validnost podstabla (da nema ciklusa) i dodaje validna podstabla u listu.
- ‘**has_cycle’:** Proverava da li postoji ciklus između roditelja i deteta.
- **‘__hash__’** i **‘__eq__’** : Ove metode omogućavaju da se stabla upoređuju i koriste u skupovima.
- **‘__repr__’:** Vraća reprezentaciju stabla u obliku stringa, korisno za ispis.

Funkcija `is_ultrametric:`

- Proverava da li je dato stablo ultrametrično. To se postiže proverom da li je udaljenost između svaka dva lista veća ili jednaka odgovarajućoj vrednosti u matrici rastojanja M i da li su sve udaljenosti od korena do listova iste.

Funkcija `generate_unique_ultrametric_subtrees`

- Generiše sva jedinstvena ultrametrična podstabla iz datog početnog stabla. Koristi `is_ultrametric` da proveri da li je svako generisano podstablo ultrametrično.

Funkcija `brute_force_for_minimum_ultrametric_tree`

- Ova funkcija kreira početno stablo dodajući sve moguće grane na osnovu matrice rastojanja M, zatim generiše sva jedinstvena ultrametrična podstabla i pronalazi ono sa minimalnom težinom. Vraća optimalno stablo i njegovu ukupnu težinu.

Funkcija `load_distance_matrix` i `load_all_distance_matrices`

- **`load_distance_matrix`:** Učitava matricu rastojanja iz datoteke.
- **`load_all_distance_matrices`:** Učitava sve matrice rastojanja iz direktorijuma i skladišti ih u rečniku gde je ključ ime datoteke, a vrednost je matrica.

Glavna petlja

- U ovoj sekciji koda, sve matrice rastojanja iz zadatog direktorijuma se obrađuju pomoću funkcije `brute_force_for_minimum_ultrametric_tree`, a rezultati (minimalno ultrametrično stablo i njegova težina) se ispisuju.

3.2. Branch-and-bound

Kao i za prethodni algoritam, kod možete pronaći na sledećem linku:
branch_and_bound.py

Klasa 'Tree':

- **'__init__'** :Inicijalizuje stablo sa praznim rečnicima `tree` (koji čuva decu svakog čvora) i `parents` (koji čuva roditelja svakog čvora).
- **'add_edge':** Dodaje ivicu između roditelja i deteta sa težinom. Ažurira oba rečnika.
- **'calculate_total_weight'**: Izračunava ukupnu težinu svih ivica u stablu.
- **'find_path_weight'**: **i '_dfs'**: Pronalaženje težine puta između dva čvora pomoću pretrage u dubinu (DFS). Koristi se memorisanje rezultata putem `lru_cache` kako bi se ubrzalo pretraživanje.
- **Funkcija `maxmin_permutation`**: Generiše permutaciju čvorova distance matrice `M` tako da je minimizovano maksimalno rastojanje između uzastopnih čvorova. Ovo je početni korak u branch-and-bound algoritmu.

- **Funkcija `upgmm`**: Implementacija UPGMA algoritma za grupisanje, koji konstruiše dendrogram (stablo) na osnovu distance matrice `M`. Ovaj algoritam je modifikovan tako da kombinuje klastere sa minimalnim maksimalnim rastojanjem i koristi se kao gornja granica u branch-and-bound algoritmu.

- **Funkcija `compute_lower_bound`**: Računa donju granicu ukupne težine stabla koristeći minimalne težine između već umetnutih listova u stablo.

- **Funkcija `is_ultrametric`**:Proverava da li je stablo ultrametrično, što znači da za svaka tri lista u stablu važi ultrametrična nejednakost.

- **Funkcija `branch_and_bound`**: Implementacija branch-and-bound algoritma za pronalaženje optimalnog ultrametričnog stabla na osnovu distance matrice `M`. Koristi `maxmin_permutation` za inicijalizaciju, `upgmm` za postavljanje gornje granice i rekurzivno istražuje moguća stabla kako bi pronašla ono sa minimalnom težinom.

- **Funkcija `insert_leaf`**:Umeće novi list u stablo ako je umetanje validno (ne krši strukturu stabla i ultrametričnost).

- **Funkcije za učitavanje matrica**: `load_distance_matrix` i `load_all_distance_matrices` učitavaju distance matrice iz fajlova u direktorijumu.

- **Glavni deo koda**:Učitava sve distance matrice iz datog direktorijuma i za svaku matricu koristi `branch_and_bound` da pronađe optimalno stablo i njegovu težinu, koja se zatim ispisuje.

3.3. Genetski algoritam

Kod za genetski algoritam mozete videti ovde.

- Klasa `Tree` predstavlja drvo sa `n` čvorova. Čuvaju se ivice (`edges`) kao parovi čvorova sa težinama, dok se `fitness` koristi za ocenjivanje kvaliteta drveta.

- Metod `add_edge` dodaje ivicu između čvorova `u` i `v` sa težinom `weight`. Metod `calculate_total_weight` računa ukupnu težinu svih ivica u drvetu.

- Metod `is_ultrametric` proverava da li je drvo ultrametrično na osnovu uporedbe težina ivica sa odgovarajućim vrednostima u matrici rastojanja.

- Metod `calculate_balance_penalty` računa kaznu za balansiranje drveta na osnovu razlika u rastojanjima između listova.

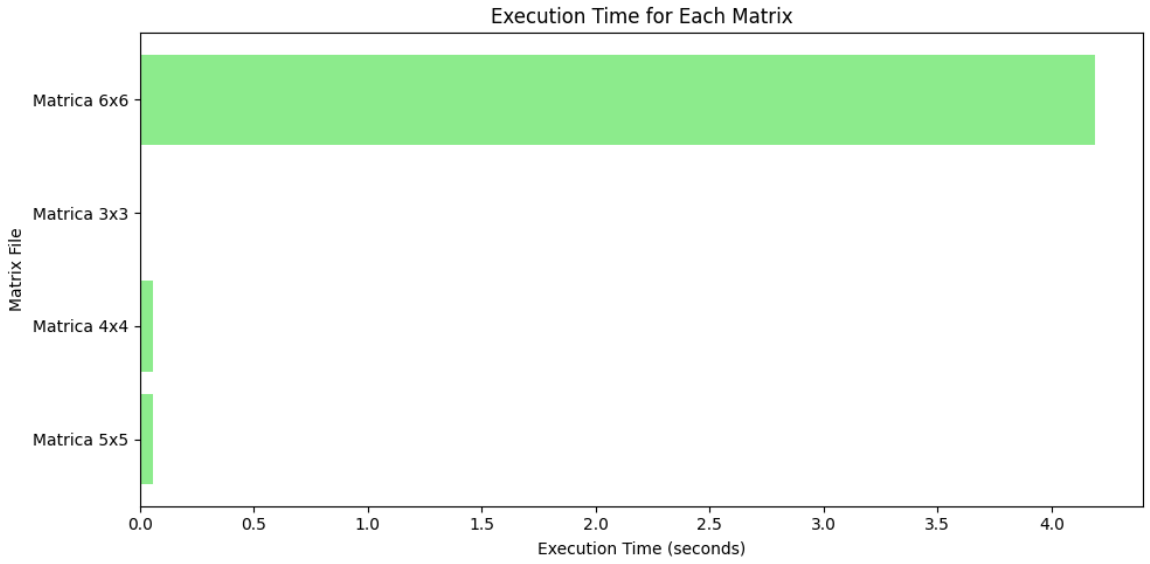
- Metod `calc_fitness` računa funkciju `fitness` za drvo, koja je zasnovana na težini drveta, kazni za broj ivica, i kazni za balansiranje.

- Klasa `Individual` predstavlja pojedinca u okviru genetskog algoritma. Sadrži drvo (`tree`) i njegovu `fitness` vrednost. Drvo može biti izgrađeno na osnovu matrice rastojanja.

- `GeneticAlgorithm` implementira genetski algoritam za optimizaciju drveta. Algoritam koristi selekciju, ukrštanje, i mutaciju da iterativno poboljšava populaciju stabala.

4. Analiza rezultata

4.1. Brute-force



Slika 1. Grafik vremena izvršavanja za svaku matricu - brute force

Na x-osi je prikazano vreme izvršenja u sekundama, dok su na y-osi nazivi matrica.

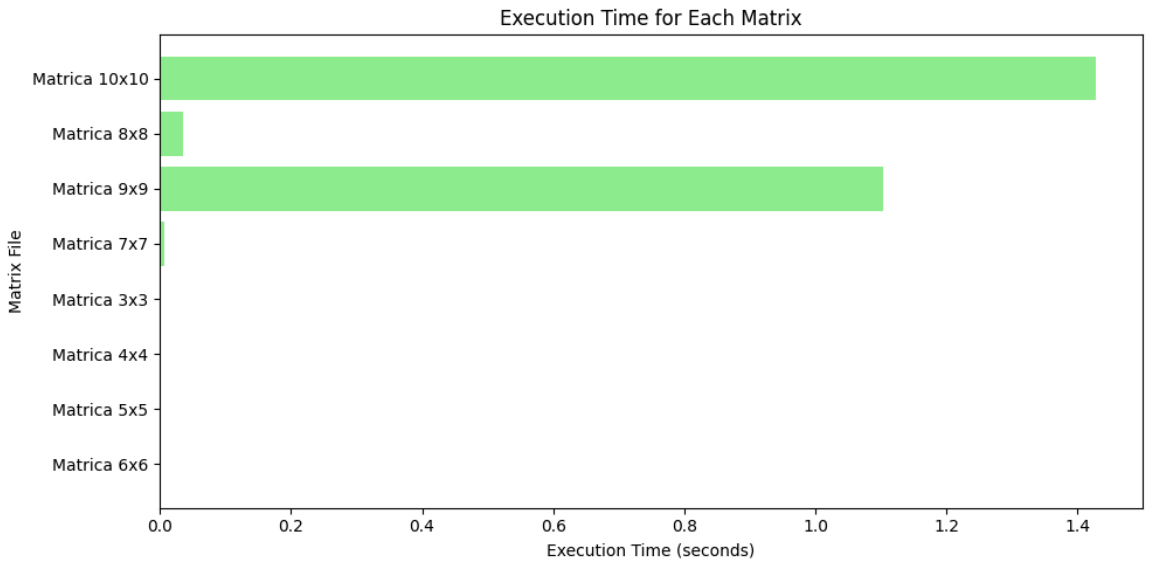
Matrica 6x6: Ima najduže vreme izvršenja, koje prelazi 4 sekunde. Ovo ukazuje na značajno složeniju obradu u poređenju sa ostalim matricama.

Matrice 4x4 i 5x5: Vremena izvršenja su mnogo kraća, ali još uvek primetno veća nego kod najmanjih matrica. Ukazuje na proporcionalan rast složenosti sa veličinom matrice.

Matrix 3x3:Ima najkraće vreme izvršenja. Ovo sugeriše da se algoritam brzo izvršava kada je matrica manja i kada ima manje mogućih kombinacija.

Ovaj grafik nam daje uvid u to kako složenost i veličina matrice direktno utiču na vreme koje je potrebno algoritmu da pronađe optimalno rešenje. Matrice sa većim dimenzijama zahtevaju znatno više vremena za obradu, što je očekivano zbog prirode problema koji se rešava.

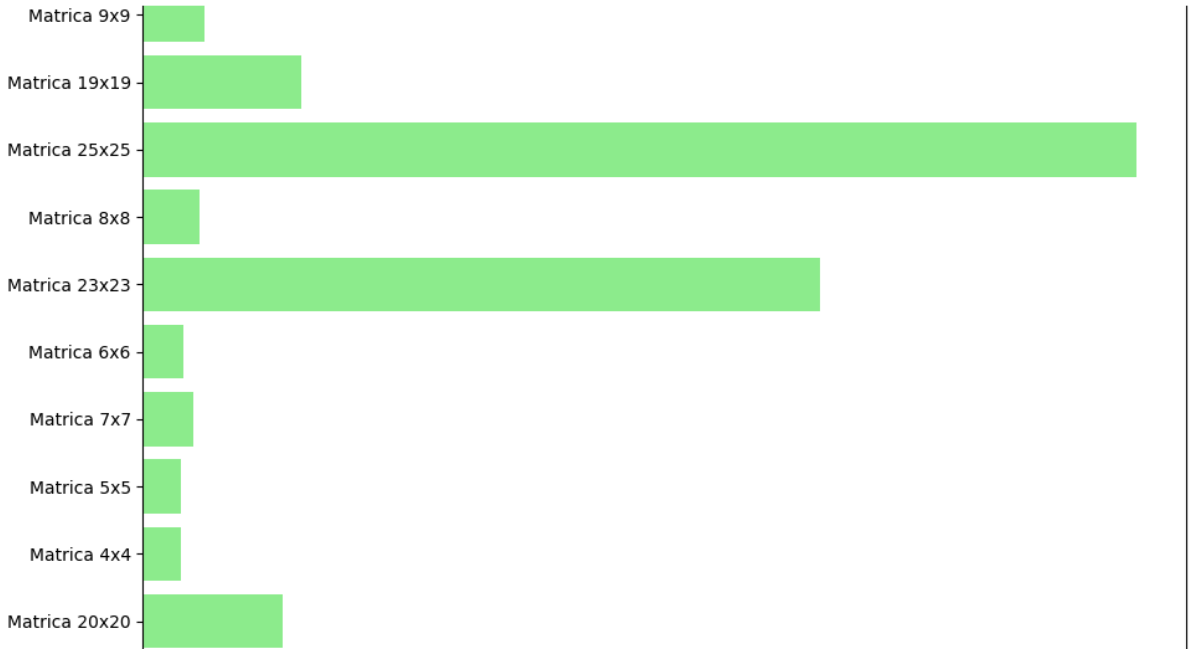
4.2. Branch-and-bound



Slika 2. Grafik vremena izvršsavanja za svaku matricu - branch-and-bound

Grafik jasno pokazuje kako vreme izvršenja raste sa veličinom matrice. Matrice sa većim dimenzijama (10x10, 9x9) zahtevaju više vremena za obradu, dok manje matrice (7x7 i manje) imaju veoma kratko vreme izvršenja. Ovi rezultati su u skladu sa očekivanjem da kompleksnost problema raste sa povećanjem dimenzija matrice.

4.3. Genetski algoritam



Slika 3. Grafik vremena izvršavanja za svaku matricu - genetic algorithm

Matrica 25x25:Ima najduže vreme izvršenja među svim prikazanim matricama.Ovo je očekivano, jer veće dimenzije matrice obično zahtevaju više vremena za obradu zbog složenosti računanja.

Matrica 20x20 i 19x19:Imaju srednje vreme izvršenja u poređenju sa ostalim matricama.Iako su ove matrice velike, njihovo vreme izvršenja je kraće od matrica 23x23 i 25x25, ali duže od manjih matrica.

Manje matrice (npr. 6x6, 7x7, 8x8, itd..): Sve manje matrice imaju znatno kraće vreme izvršenja.To je zbog njihove male dimenzije, što omogućava bržu obradu algoritma.