Maximum Subforest problem

Filip Gajin

Uvod

Maximum Subforest problem predstavlja NP-težak problem koji se tiče grafova (konkretnije, stabala – povezanih necikličnih grafova). Definicija problema je sledeća:

- INSTANCA: Drvo T = (V, E) i skup podstabala H.
- REŠENJE: Podstablo drveta T takvo da ono ne sadrži nijedno podstablo izomorfno bilo kojem elemntu skupa H.
- MERA: Kardinalnost našeg izabranog podstabla, to jest broj grana.

Drugim rečima, dato nam je neko drvo T i skup „zabranjenih podstabala“ H. Naše rešenje ne samo da ne sme sadržati nijedno od tih podstabala, već i ne sme sadrži stablo koje je izomorfno nekom iz tih skupova. To znači da ako, na primer, u skupu H ima stablo sa granama (1, 2) i (2, 4), rešenje ne sme biti nijedno stablo koje u sebi sadrži takvu strukturu, odnosno ne sme imati dve spojene grane; u ovom slučaju, najveće podstablo koje možemo uzeti za rešenje jeste samo jedna grana bilo koja dva čvora. Jasno je da za svakog kandidata za rešenje moramo proveravati sve njegove podgrafove i da li su oni izomorfni sa bilo kojim od stabala iz skupa H – takođe, sa većim dimenzijama drveta T, računica postaje sve veća i veća. Otuda i nastaje problem.

Pristup problemu

Problem je rešavan u Jupyter Notebook-u. Za rad sa grafovima je korišćena biblioteka networkx, koja nudi mnogo pogodnosti i funkcija koje su bile potrebne.

Pokušano je rešavanje problema sa tri metode:

1. Brute-force algoritmom
2. Pohlepnim algoritmom
3. Genetskim algoritmom

Očekivalo se da će sva tri algoritma moći da reše problem nad malim instancama, i to u „dobrom vremenu“, a da će se sa povećavanjem instanci povećati i efikasnost algoritama, pre svega Brute-force algoritma. Za pohlepni algoritam se nije moglo garantovati da će naći optimalno rešenje, o čemu će kasnije biti više reči. Od genetskog algoritma se očekivalo da će nuditi najefikasniji pristup nad velikim instancama.

Imali smo zaseban program (na slici ispod) koji je proveravao efikasnost algoritama. U njemu smo i generisali instance problema, to jest stablo T i elemente skupa H. Za skup H je postavljeno da sadrži 5 elemenata, dok se za dimenzije stabla T nudio niz dimenzija, koje su bile prilagođene svakom od algoritama (na primer, za Brute-force se nije mogla ispitati i dimenzija 200 – mnogo bi se čekalo – dok za genetski jeste).

```
def generate_instance(N, num_of_subtrees):
    T = nx.random_tree(N)
    H = []
    nodes = list(T.nodes)

    for i in range(num_of_subtrees):
        subtree_nodes = random.sample(nodes, 10)
        subtree = T.subgraph(subtree_nodes).copy()
        if nx.is_connected(subtree):
            H.append(subtree)
    return T, H

def merenje_vremena(maximum_subforest_function, tree_sizes, num_of_subtrees):
    times = []
    i = 1
    for N in tree_sizes:
        T, H = generate_instance(N, num_of_subtrees)
        start = time.time()
        subforest = maximum_subforest_function(T, H)
        print("zavrseno: {i}")
        i = i+1
        end = time.time()

        time_elapsed = end - start
        times.append(time_elapsed)

    return times
```

Brute-force algoritam

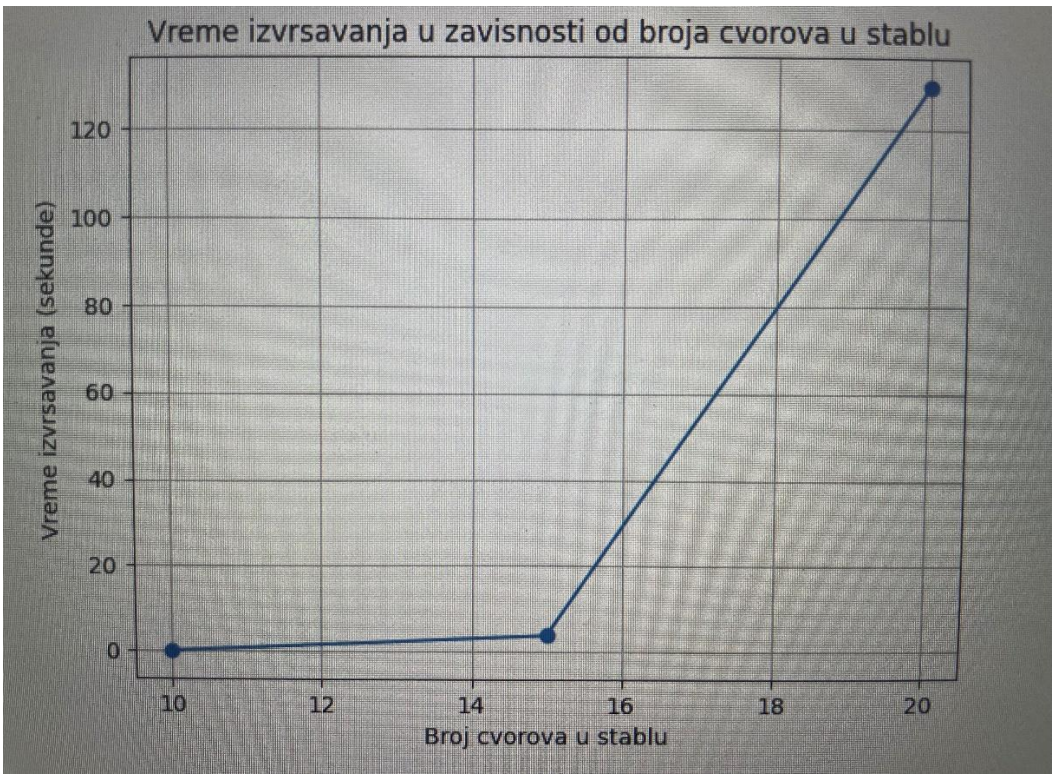
Gruba sila je rešavala problem tako što je pravila sve podgrafove stabla T, i za svako proveravalo da li ispunjava uslove (dakle, i njemu pravimo podgrafove i za svaki proveravamo da li je izomorfan sa nekim elementom skupa H). Takođe, bilo je bitno proveravati i da li je dobijeni podgraf povezan, i nepovezani su se isključivali iz rešenja.

Ako nađemo neki podgraf koji ispunjava uslove, računamo njegovu kardinalnost i upoređujemo je sa do tada najboljim nađenim rešenjem.

Ispostavilo se da ovaj algoritam dobro radi nad instancama malih dimenzija, ali vreme izvršavanja mu se eksponencijalno povećava s povećavanjem tih dimenzija.

U proveri vremena izvršavanja, data su mu nasumično generisana stabla T sa dimenzijama, redom: 10 čvorova, 15 čvorova, 20 čvorova.

Već s dolaskom do 20 čvorova, kao što se može videti sa slike, vreme postaje izuzetno veliko, zbog čega ovaj algoritam nije preporučljiv za reševanje ovog problema.



Pohlepni algoritam

Pohlepni algoritam je rešavao problem tako što počinje od jednog čvora kao rešenja, i redom pokušavao da dodaje granu u graf, nakon čega bi usledila provera da li se dodatkom narušio neki od uslova (sadrži podstablo izomorfno stablu iz H, ili da je nepovezan); ako bi se narušio, čvor bi se izbacio i nastavilo bi se sa dodavanjem sledećih čvorova, to jest grana.

Pre toga, ipak, lista grana nad kojom se pokušava dodavanje je sortirana opadajuće po stepenu čvorova koje čine granu. Stepen označava koliko suseda neki čvor ima.

Ovaj algoritam nad malim dimenzijama može dati i netačno rešenje, ali je uvek blizu tačnog. Nad većim dimenzijama, kako se vidi na slici ispod, radi sa relativno dobrim vremenom. Ipak, ne preporučuje se za rešavanje ovog problema upravo zbog svog negarantovanja tačnog rešenja. Stvar je u tome što on postepeno gradi neki graf i čim dođe do strukture koja je izomorfna nekom podstablu iz H, on će se „vratiti unazad“, a može biti da, ako bi nastavio dalje, došao bi do strukture koja predstavlja najbolje rešenje (to jest, umesto da izbaci granu koja je trenutno narušila uslov, ako bi izbacio neku raniju granu a preko ove trenutne nastavio građenje, došao bi do tačnog rešenja).

Možda bi se taj nedostatak mogao rešiti sa nekim vidom backtracking-a. Takođe, mogao bi se unaprediti algoritam dodavanjem nekih boljih heuristika od one koja je iskorišćena ovde (sortiranje liste grana po stepenima čvorova).

U proveri vremena izvršavanja, date su mu dimenzije stabla T, redom: 10, 40, 70, 100.

Genetski algoritam

Genetski algoritam je bio najkomplikovaniji za implementaciju; zauzvrat se pokazao najboljim.

Početnu populaciju su činili random generisani, povezani podgrafovi stabla T.

Fitnes funkcija je vraćala 0 ako jedinka nije povezana, ako je prazna i ako narušava uslov iz definicije problema (sadrži podstablo izomorfno nekom elementu iz H). U suprotnom, vraćala je broj grana u jedinci, to jest njenu kardinalnost.

Za selekciju se koristio turnir (turnir_size je bio 5).

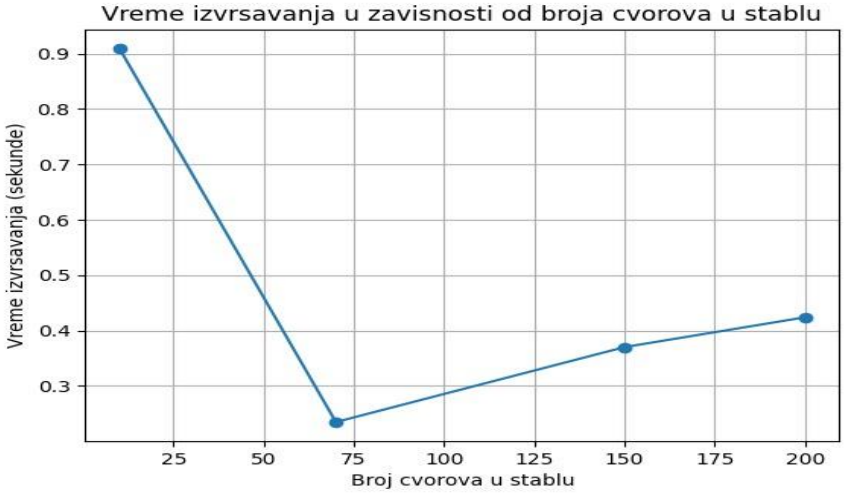
Ukrštanje je od dva roditelja pravilo dete tako što je naizmenično uzimalo grane iz prvog i drugog roditelja sa verovatnom 50% i dodavalo ga detetu (koje je na početku bilo prazno) ako već ne sadrži tu granu (možda ga je već uzelo iz prvog roditelja). Na kraju se morao odraditi i presek deteta sa stablom T da bi se osiguralo da je dobijeni graf ovim ukrštanjem zaista podstablo stabla T.

Mutacija je sa verovatnoćom od 50% dodavala rendom grane iz stabla T u jedinku; u suprotnom je izbacivala rendom granu iz nje. Mutation_rate je bio 0.1.

Population_size i generations su iznosili, redom: 50 i 100.

U proveri su mu date dimenzije stabla T: 10, 70, 150, 200. Za dimenziju 10, radio je najsporije! Za ostale je bio brz, ali je vreme ipak raslo sa povećavanjem dimenzija.

Ovaj algoritam se preporučuje za rad sa većim dimenzijama instanci ovog problema.



Zaključak

Kao što smo videli, Brute-force je najgori za rad sa instancama većih dimenzija ovog problema, ali na malim dimenzijama radi prilično dobro. On nam može korisiti za razumevanje prirode problema, na malim primerima.

Pohlepni algoritam je nebezbedan zbog svoje nepredvidivosti i negarantovanja najboljih rešenja. Ipak, ako nam nije potrebno nužno najbolje rešenje, već nam je od većeg značaja vreme izvršavanja sa dovoljno dobrim rešenjem, može se i on koristiti.

Genetski algoritam, za razliku od prethodna dva, je komplikovan za implementaciju, ali daje najbolje rezultate u dovoljno dobrom vremenu.

Nadalje se može eksperimentisati sa dimenzijama skupa H (ovde je ta dimenzija bila fiksna) – međutim, očekuje se slično ponašanje kao i do sad.