Optimizacija problema maksimalnog zajedničkog podstabla

Mladen Dilparić

September 29, 2023

1 Uvod

U ovom radu se bavimo optimizacionim tehnikama (tehnika simuliranog kaljenja) na problemu maksimalnog zajedničkog stabla, gde će nam mera kvaliteta biti broj čvorova nadjenog maksimalnog stabla, uporedjena sa rezultatom izvršavanja algoritma grube sile.

2 Opis problema

2.1 Maksimalno zajedničko stablo

Definicija problema: Za ulaz C = (T1, T2, .. Tn) korenskih stabala naći stablo T' takvo da je izomorfno svim stablima u kolekciji, čija je veličina (broj čvorova) maksimalna.

Problem maksimalnog zajedničkog stabla svoju primenu nalazi u bioinformatici kao i biohemiji. Varijacije ovog problema svoju primenu nalaze i u dizajnu kompajlera, detekciji plagijarizma i slično.
Problem nalaženja maksimalnog zajedničkog stabla u kolekciji stabala je do dana pisanja ovog seminarskog rada ostao nedovoljno istražen, sa tek par naučnih radova koji direktno adresiraju ovaj problem, dok na temu optimizacije nije nadjena odgovarajuća literatura za ovaj problem. Pripada klasi NP teških problema tj. problema za koji algoritam koji polinomijalno po veličini ulaza može da da odgovor nije nadjen.
U radu su razmatrana stabla bez oznaka čvorova, sa listom dece koja je neuredjena, tj. poredak dece čvorova nije bitan. Kao mera kvaliteta za ovaj problem uzet je broj čvorova nadjenog podstabla za manje test slučajeve, dok je za veće test slučajeve, zbog nemogućnosti izvršavanja algoritma grube sile na njima uzet odnos broja čvorova stabla u odnosu na minimalno stablo kolekcije. Za test kolekcije se pokazalo da rešenje uglavnom ima odnos izmedju 0.5 i 0.7 za nasumično kreirano drveće koje ima neku uslovljenjost u pogledu veličine dece i maksimalne visine, sem za kolekcije raznorodnog drveća gde je taj odnos mnogo manji 0.3-0.5.

2.2 Algoritam grube sile

Ukratko algoritam grube sile će za kolekciju datih stabala naći najmanje stablo kolekcije po broju čvorova,a potom će za za minimalno stablo kolekcije naći sva različita podstabla. Kako je nalaženje svih podstabala generalno eksponencijalno, provera da li je neko stablo podstablo drugog stabla ima polinomijalnu složenost(za ovu klasu drveća moguće je ovaj problem svesti na problem maksimalnog uparivanja u bipartitnom grafu, koji je uradjen preko Hopkroft-Karp algoritma), jasno je da bi ovakav pristup zbog kombinatorijalne eksplozije kod stabala sa većom količinom čvorova kao i dubinom stabla bio praktično nemoguć, algoritam grube sile u slučajevima malog broja čvorova daje zadovoljavajuće rezultate. Generisanje podstabala jednog stabla ćemo raditi rekurzivno za njegovu decu a potom na osnovu generisanih podstabala dece naći sve moguće kombinacije sa korenom stabla. Nakon toga prolazimo kroz listu podstabala uredjenu po broju čvorova opadajuće i proveravamo da li je podstablo svih stabala u kolekciji.

2.3 Algoritam simuliranog kaljenja

Algoritam simuliranog kaljenja svoje ime, kao i inspiraciju uzima iz metalurškog procesa kaljenja čelika.
Kako je pri početku procesa kaljenja inicijalna temperatura čelika velika, čelik se lakše kali tj. oblikuje. Kako prolazi vreme čelik se sve više hladi, i sam proces kaljenja postaje teži sa opadanjem temperature.
Postepeno hladjenje se u algoritmu simuliranog kaljenja može protumačiti kao postepeno smanjivanje verovatnoće da se loše rešenje problema prihvati, kako se prostor rešenja istražuje. Neki generalni pseudokod ovog algoritma izgleda ovako:
(1) Generisati početno rešenje s
(2) Inicijalizovati vrednost (fitnes funkcija)najboljeg resenja f* = f(s)
(3) Dok nije ispunjen uslov zaustavljanja(maksimalan broj iteracija, ili drugi parametri), ponavljaj:
    (3a) Izabrati proizvoljno resenje s' u okolini resenja s
    (3b) Ako je fitness(s') ¡ fitness(s), onda je s = s'
    (3c) Ako je fitness(s') ¿= fitness(s), onda:
        (3c1) Ažurirati vrednost opadajuce funkcije p
        (3c2) Izabrati proizvoljno q iz intervala [0,1]
        (3c3) Ako je p ¿ q, onda s = s'
    (3d) Ako je f(s') ¡ f*, onda je f* = f(s')
(4) Ispisati vrednost resenja f*

Dakle, u algoritmu simuliranog kaljenja najveću ulogu u usmeravanju ka dobrom rešenju u prostoru pretrage igraju dobri odabiri funkcija inicijalizacije rešenja, fitnes funkcije, kao i načina generisanja okoline rešenja.

Generacija inicijalnog rešenja za ovaj problem je probana sa 3 različite verzije:
-vraćamo samo stablo sa jednim čvorom

-vraćamo minimalno stablo u kolekciji po broju
-vraćamo nasumično generisano podstablo minimalnog stabla kolekcije gde sa nekom verovatnoćom uključujemo čvorove minimalnog stabla po broju kolekcije

Fitnes funkcija je uzeta kao broj čvorova stabla ako je stablo podstablo kolekcije, a 0 ako nije.

Način generisanja okoline rešenja zavisi od same generacije inicijalnog početnog rešenja. Ako je uzeto da je minimalno stablo inicijalno rešenje, isprva se pomeramo u okolinu rešenja tako što imamo veću šansu za brisanjem nekog nasumičnog čvora, koje postepeno opada sa iteracijama, u korist dodavanja čvora. Suprotno radimo za stablo sa jednim čvorom gde inicijalno mnogo više dodajemo čvorove nego što ih brišemo. Za nasimično stablo možemo dodavati ili brisati čvorove sa nekom verovatnoćom.

3 Eksperimentalna analiza

3.1 Kolekcija test podataka

Kako je ovaj problem nedovoljno istražen na polju računarske inteligencije, test kolekcije za ovaj problem predstavljaju kolekcije nasumičnog drveća predstavljenog kao niske. Drveće se može predstavljati sa svojom Knutovom reprezentacijom gde je čvor predstavljen sa () , sva deca čvora potom se smeštaju rekurzivno unutar zagrada čvora roditelja, tako da se čvor sa 3 deteta može predstaviti kao (()()()) tj ako sa 0 označimo otvorenu a sa 1 zatvorenu zagradu kao 00101011.

3.2 Test uslovi

Svi test primeri su izvedeni na procesoru AMD® Fx(tm)-6200 sa 6 jezgara, 8 Gb rama na OS Ubuntu 20.04.6 LTS 64-bit.

3.3 Eksperimentalni rezultati

Pri rešavanju problema probani su različiti parametri kao deo rešenja. Zbog broja kombinacija koje treba istražiti najbolje rezultate za veće kolekcije će dati i veći broj iteracija u samom algoritmu kaljenja.Za veće kolekcije većih stabala veći broj iteracija vodi ka ispitivanju više kombinacija pa samim tim i boljem kvalitetu rešenja.
Kako se pokazalo najbolje rezultate u pogledu parametra generisanja susednih rešenja je pokazalo generisanje nasumičnog podstabla minimalnog stabla kolekcije, gde je svaki čvor počevši od korenog imao neku verovatnoću P da se nadje u inicijalnom rešenju. Jasno je da ako izaberemo premalo ili preveliko stablo to utiče na tačnost rešenja. Delom uslovljeno samim test podacima, ukoliko imamo retko zajedničko drvo(u pogledu čvorova) bolje nam je da uzmemo manje podstablo, analogno tome za drugi slučaj. Na kraju se eksperimentalno došlo do zaključka da najbolje rezulate daje P od 0.7 do 0.85, tj da više negativno utiče odabir manje veličine nego veće na tačnost(veličinu) rešenja.

Test primer	Broj stabala, veličina minimalnog	BF algoritam	SK algoritam	BF vreme	SK vre
test1	9	20	20	1.025s	1.161s
test2	10	25	22	1min	3s
test12	14	18	18	2s	3s
test3	18	30	30	11s	11s
test19 10 4	19	25	25	13s	14s
test4	20	28	27	30min	8s
test40	40 : 120	-	70	-	204s
multi	15	13	15	250s	16s
mnogo	20	14	14	126s	3s
test49 8 5	49	28	25	400s	27s
bigCollection	20	15	15	21s	2s
bla1	20	31	31	3365s	48s

Table 1: Eksperimentalni podaci

Za manje ulazne veličine se pokazalo da algoritam grube sile daje dobre rezultate za isto ili bolje vreme. Najviše uticaja na njegov rad pre svega ima dubina minimalnog kao i broj čvorova stabla. Simulirano kaljenje pak, za veće ulaze daje rezultate koji nisu po broju čvorova mnogo manji nego nadjena maksimalna veličina BF algoritma, za daleko kraće vreme. Kvalitet rešenja za veće slučajeve je donekle i uslovljen nasumičnim drvetom sa kojim počnemo, ali generalno, pokretanje algoritma više puta daje dovoljno dobre aproksimacije. Zbog mogućnosti da na izlazu dobijemo rešenje koje uopšte nije zajedničko podstablo(nasumično stablo nije pogodno, i nikad ne nadjemo bilo koje zajedničko podstablo), restartovanje procesa je moguće. Ako nam je bitno da dobijemo što bolje rešenje možemo ponovo restartovati ceo proces, a ako želimo garanciju da ćemo dobiti neko rešenje možemo pokrenuti proces sa inicijalnim rešenjem sa jednim čvorom.
U jednom od test primera(multi.txt) algoritmom simuliranog kaljenja je dobijeno i veće rešenje, pa je potrebno pojasniti da se to dobija zbog toga što je dobijeno drvo algoritmom simuliranog kaljenja podstablo u smislu povezanog podgrafa, dok algoritam grube sile koristi top-down definiciju postabla, koja je restrikcija na izomorfizam podstabla u pogledu toga što se traži da mapiranje sadrži oba korena drveta a ne samo koren podstabla.

4 Zaključak

Može se zaključiti da u slučajevima kad je minimalno stablo date kolekcije malo, po broju čvorova ili dubini, ili retko (sa manjim brojem dece po čvoru) algoritam grube sile će nam dati tačne i brze rezultate. Medjutim, kako raste broj kombinacija podstabala minimalnog stabla koje treba uporediti, jasno je da vreme izvšavanja ovog algoritma pogotovo za velike slučajeve biti ogromno(neupotrebljivo). Rešenja algoritma simuliranog kaljenja nisu u većini slučajeva odstupala mnogo od rešenja algoritma grube sile, uglavnom dajući rešenja koja su od optimalnih

odstupala za oko 10 posto u najgorem slučaju. Ono što mu je glavna prednost jeste brzina nalaženja rešenja u slučajevima kad je stablo velike dubine sa većim brojem dece.

Za eventualne pravce daljeg razvoja moguće je da generišemo zajedničko podstablo minimalnog i drugog najmanjeg kao inicijalno rešenje , što je polinomijalan proces. Preprocesiranje kolekcije stabala i prepoznavanje odredjenih karakteristika u test podacima(prosečan broj dece po nivoima stabla npr.) nam takodje može biti dalji pravac razvoja. Dodatno razmatranje parametara kao i načina generisanja samog rešenja može biti predmet daljeg razmatranja.

5 Literatura

1.Beleske sa predavanja iz Racunarske Inteligencije
2.Knjiga Algorithms on Trees and Graphs by Gabriel Valiente
3.Akutsu, T., and Halldórsson, M. (1994), "On the approximation of largest common point sets and largest common subtrees"