Optimizacija - Minimum Equivalent Digraph

Lazar Stanojević

27. Jun 2023.

Sadržaj

1 Uvod

U ovom radu bavićemo se optimizacijom Minimum Equivalent Digraph (skraćeno MED) problema koristeći VNS (Variable Neighborhood Search) algoritam.

VNS je metaheuristička metoda za rešavanje problema kombinatorne optimizacije. Algoritam istražuje okolinu tekućeg rešenja, uz pomoć lokalne pretrage, kako bi se približio lokalnom optimumu u toj okolini, i prelazi na sledeće rešenje samo ukoliko je našao bolje.

U daljem radu, opisaćemo implementaciju ključnih delova VNS algoritma, kao i pohlepan, i algoritam grube sile, koji će nam koristiti za poređenje rezultata.

2 Opis problema

U ovom delu, biće prezentovan opis problema koji rešavamo, kao i tehnike korišćene za njegovo rešavanje.

2.1 MED

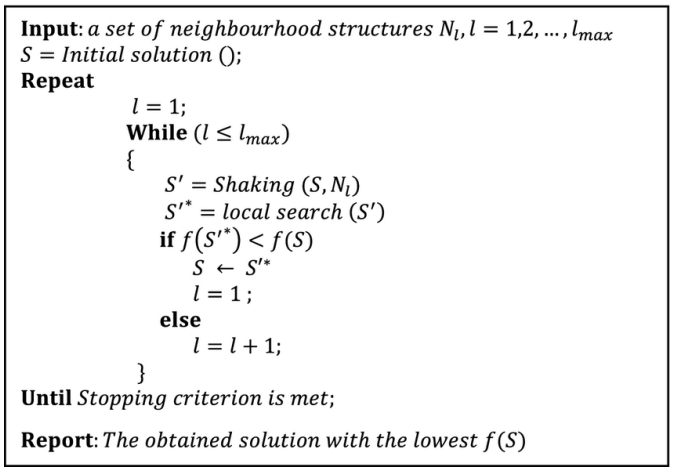
MED problem spada u grupu NP teških problema. Cilj nam je naći ekvivalentan usmereni graf G'(V, E'), za usmereni graf G(V, E') koji se daje kao ulaz. V predstavlja skup čvorova, dok su E i E' skupovi grana, takvi da važi: E' je podskup od E. Kada se kaže ekvivalentan, misli se na to da su svi čvorovi do kojih je moglo da se dođe nekim putem u grafu G, ostali dostupni u grafu G'. Na primer, ukoliko je postojao usmereni put od čvora u do čvora v u grafu G, moraće da postoji i u grafu G', pri čemu težimo da minimizujemo broj grana koji ne menja dostupnost čvorova.

Neke od primena ovog problema u praksi su u okviru mreža, za analizu promena struktura mreže, ili na primer u okviru sinhronizacije baza podataka.

Kako na ovu temu u kontekstu optimizacije nema literature, ili bar ne lako dostupne, jedino merilo i kriterijum za upoređivanje biće nam naše različite varijante metoda za rešavanje ovog problema.

2.2 VNS

Na slici 1 prikazano je kako bi otprilike mogao da izgleda pseudokod za VNS. Na osnovu ovoga možemo da zaključimo da nam je od najvćeg značaja može biti kako implementiramo funkcije inicijalizacije rešenja, lokalne pretrage, shakinga, zatim kako računamo vrednost rešenja, odnosno šta nam je fitness funkcija f.



Slika 1: VNS pseudokod

3 Implementacija algoritama

Slede opisi kako su korišćene metode realizovane.

3.1 Gruba sila i pohlepni algoritam

Algoritam grube sile, kao što mu i samo ime kaže, ne radi ništa preterano pametno. Kako nam je cilj da nađemo graf G', čiji je skup grana podskup skupa grana grafa E, jedini način da budemo sigurni da smo u tome uspeli, jeste da prođemo kroz sve podskupove skupa grana E, ili skoro sve, ukoliko bi koristili neka odsecanja, i proverimo da li dobijeni graf ima traženo svojstvo.
Ovde je algoritam grube sile implementiran bez odsecanja, tako što prolazi kroz sve podskupove grana grafa G, i proverava, uz pomoć Flojd Varšalovog algoritma, da li se dostupnost čvorova ne menja.
Moguća modifikacija bi bila, da ukoliko za neki podskup grana zaključimo da dobijeni graf ima traženo svojstvo, njegove nadskupove ne proveravamo.
Kako je Flojd Varšalov algoritam kubne složenosti po broju čvorova, a mi ga primenjujemo za svaki podskup grana, kojih ima eksponencijalno u odnosu na broj grana, zaključujemo da je složenost grube sile $O(V^3 \cdot 2^E)$, na osnovu čega nam je jasno da je algoritam veoma osetljiv na povećanje broja grana, pa samim tim i neće moći da završi u razumnom vremenu čak i za grafove sa svega nekoliko desetina grana.

Sa druge strane, pohlepni algoritam je najbrži od punuđenih, ali kao što mu i ime kaže, pohlepan je, pa postoje slučajevi u kojima se on baš zaglavljuje u nekom lokalnom optimumu.
Ovde je pohlepni algoritam implementiran tako da prolazi kroz skup grana grafa G, i za svaku provera da li se njenim uklanjanjem gubi svojstvo dostupnosti čvorova, i ako ne, uklanja je.
Jasno je, da kada je poredak izbacivanja grana bitan, a algoritam ne pogodi taj poredak, neće moći da nađe dobro rešenje, ili ne bar onoliko dobro koliko očekujemo da će naći VNS. Najjednostavniji primer koji oslikava manu pohlepnog algoritma, je taj kada imamo dva čvora, i dva puta između ta dva čvora, a prva grana na koju algoritam naiđe je grana kraćeg puta. Tada će algoritam eliminisati kraći put, i zaista iako će dobijeni graf imati traženo svojstvo, bolje rešenje bi bila eliminacija grana dužeg puta.

3.2 VNS

U ovom delu biće predstavljeni opisi implementacija najvažnijih delova algoritma, koji su navedeni ranije u tekstu.

Inicijalizacija

Kako je graf predstavljen kao matrica susedstva, za inicijalizaciju je odlučeno da svaka grana sa verovatnoćom 0.8 može biti deo novokreiranog grafa, odnosno na mestu G'[i][j] će se nalaziti vrednost 1 sa verovatnoćom 0.8, ukoliko je (i, j) grana grafa G. Izbor baš ovakve konstante za verovatnoću je više heurističke prirode, pošto lokalne pretrage teže više izbacivanju grana nego dodavanju, počećemo sa većim brojem grana, jer je veća šansa da će graf odma na početku biti ekvivalentan polaznom.

Fitnes funkcija

Fitnes, odnosno vrednost grafa koji predstavlja trenutno rešenje računata je po formuli $f(G') = \frac{|E'|}{|E|^2} + \alpha \cdot R$, pri čemu je sa R označen broj puteva koji ne postoji

u grafu G' između neka dva čvora, a postojao je u G. Parametar α uzima vrednosti između 0 i 1, i služi za menjanje značaja koji R ima na vrednost rešenja, kako bi bilo moguće balansirati između onoga što želimo da više utiče na rad algoritma, broj grana ili faktor R.

Shaking

Shaking kao parametar prima k, i pomera rešenje u njegovu k-tu okolinu, koju treba da definišemo. U ovoj implementaciji je uzeto da se k-ta okolina rešenja dobija kada se proizvoljnih k grana iz polaznog grafa G, koje se ne nalaze u trenutnom rešenju, doda tom rešenju. Ovakav izbor je načinjen zbog prirode lokalnih pretraga, i njihove težnje za izbacivanjem grana, pa se ovim omogućava otkrivanje nekih alternativnih puteva, koji mogu biti kraći, i pomoći u minimizaciji broja grana u rezultujućem grafu. Shaking nam donekle omogućava da se ne zaglavimo u nekom lokalnom optimumu, odnosno obezbeđuje diverzifikaciju.

Lokalna pretraga

Lokalna pretraga nam služi za eksploataciju trenutnog rešenja. Implementirane su tri vrste lokalne pretrage, i biće testirano ponašanje algoritma sa svakom od njih.

Ideja u prvoj lokalnoj pretrazi je jednostavna, imamo zadat broj iteracija, i u svakom koraku pravimo mali pomeraj u odnosu na prethodno rešenje, i ukoliko smo dobili bolju vrednost novog rešenja, pretragu nastavljamo odatle, u suprotnom ne menjamo ništa, i nadamo se da će nas naredna iteracija pomeriti u pravom smeru. Mala promena rešenja sastoji se od invertovanja neke nasumične grane početnog grafa, odnosno ukoliko se nalazi u našem grafu, uklanjamo je, a ako ne, dodaćemo je.

Pošto tražimo rešenje sa što manjim brojem grana, to je bila motivacija za drugi tip lokalne pretrage, u kojoj dok god je moguće, odnosno dok dobijamo bolju vred-

nost rešenja, izbacujemo grane iz tekućeg rešenja.

Konačno, pošto je jasno da lokalno najbolje rešenje u nekim slučajevima ne možemo dobiti samo izbacivanjem grana, razlog za to je sličan kao kod pohlepnog algoritma, ideja za treći tip pretrage zasniva se na tome da u svakom koraku, osim što izbacujemo neku granu, i dodamo neku granu početnog grafa G koja nije u skupu grana tekućeg rešenja, sa nekom verovatnoćom, recimo 0.5. Ovako omogućavamo otkrivanje alternativnih puteva između nekih čvorova, koji mogu biti kraći, pa nas to može odvesti boljem rešenju.

4 Rezultati

V	E	Brute Force	Greedy	VNS1	VNS2	VNS3	VNS vreme
3	3	2	2	2	2	2	0.05 min
6	10	5	5	5	5	5	0.05 min
7	27	-1	9	7	7	7	0.1 min
15	53	-1	18	15	16	15	3 min
30	89	-1	37	32	34	31	5 min
29	44	-1	43	43	43	43	1 min
45	918	-1	110	119	-1	-1	5 min
64	704	-1	200	206	-1	-1	10 min
55	356	-1	89	88	-1	90	15 min
72	1012	-1	102	121	-1	97	15 min

Tabela 1: Rezultati

U tabeli 1 vidimo rezultate naših algoritama na različitim grafovima. Kolone V i E označavaju broj čvorova, odnosno grana grafa, respektivno. Za njim sledi po kolona za svaki od algoritama, kao i vreme dato VNS algoritmima za izvršavanje, u poslednjoj koloni. Ako se u koloni nekog algoritma nalazi vrednost -1, to označava da nije uspeo da iskonvergira ka ekvivalentnom grafu, u smislu dostupnosti čvorova, za dodeljeno vreme, ili u slučaju grube sile, nije uspela da završi u nekom razumnom vremenu.

5 Zaključak

Na osnovu dobijenih rezultata iz poglavlja 4, možemo da izvedemo neke zaključke o algoritmima.

Zbog eksponencijalne složenosti algoritma grube sile, vidimo da on ne uspeva da završi rad u nekom realnom vremenu, za grafove sa svega par desetina grana. Tako da je gruba sila poslužila kao merilo samo na jako malim grafovima, na kojima smo videli da su se sva 3 VNS algoritma, kao i pohlepna pretraga dobro snašli, i pronašli optimalna rešenja.

Kako povećavamo dimenzije grafa, priča se komplikuje, i složenost raste. Pohlepni algoritam je pogodan u smislu njegove brzine, jer je najmanje složenosti od svih ponuđenih, i u opštem slučaju eliminiše dosta grana. Njegov problem nastupa kada u grafu ima dosta ciklusa, i alternativnih puteva, pa samo puko izbacivanje grana redom, može da napravi problem, odnosno da ne nađe dovoljno dobro rešenje. To se vidi na rezultatima iz srednjeg dela tabele, na grafovima srednje veličine, kada VNS algoritmi uvek nađu bolja rešenja od pohlepnog algoritma.

Na većim grafovima (većim u skladu za složenošću algoritama), situacija se malo menja, i VNS algoritmima, iako nisu eksponencijalni, zbog njihove složenosti može trebati relativno dosta vremena da iskonvergiraju ka ekvivalentnom grafu, a samim tim i ka boljem rešenju, pa pohlepni algoritam i u tim situacijama može biti bolji izbor. Ipak, u zavisnosti od prirode samog grafa, čak i tada, za neko relativno malo vreme, VNS uspeva da pronađe zadovoljiva rešenja, gde se opet ispoljava problem pohlepnog algoritma sa ciklusima i alternativnim putevima.

Za kraj, možemo i da primetimo i razlike u radu VNS algoritama, u zavinosti od korišćene lokalne pretrage. Naime, najbrža konvergencija se dostiže korišćenjem prvog tipa lokalne pretrage, dok uglavnom, treća lokalna pretraga daje najbolje rezultate, ukoliko joj se da dovoljno vremena. Kako deluje na ovim primerima, najgore se pokazala lokalna pretraga broj 2, što je donekle i očekivano, s obzirom

na to da u sebi ima neke pohlepne elemente, odnosno samo izbacuje grane, pa se u nekoj meri ispoljavaju iste mane kao i kod pohlepnog algoritma.

Literatura

[1] https://en.wikipedia.org/wiki/Variable_neighborhood_search

[2] https://www.researchgate.net/

[3] https://github.com/MATF-RI/Materijali-sa-vezbi

[4] https://networkrepository.com/networks.php