Univerzitet u Banjoj Luci
Prirodno-matematički fakultet
Studijski program Matematika i informatika

Rješavanje problema minimalnog dominantnog skupa u usmjerenom težinskom grafu

Seminarski rad iz predmeta Operaciona istraživanja

Studenti:
Bojan Gavrić
Vladimir Mijić

Mentor:
doc. dr Marko Đukanović

Banja Luka, septembar, 2023.

Sadržaj

Uvod

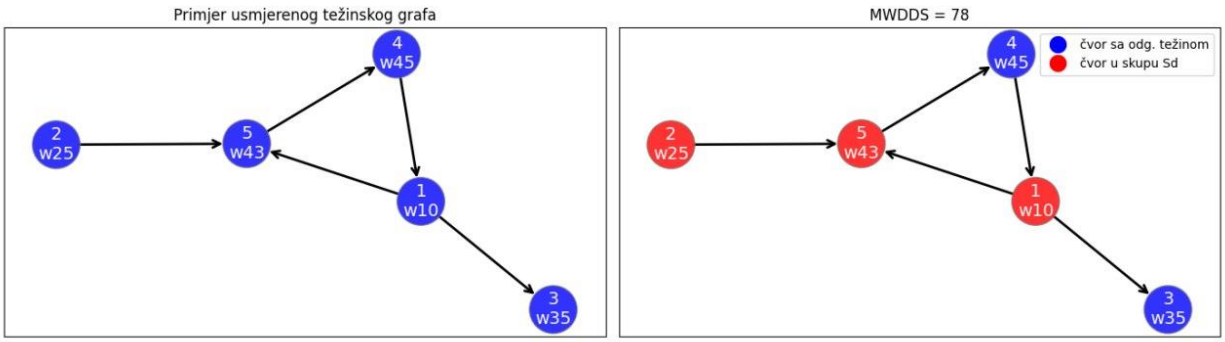
Cilj ovog seminarskog rada je testiranje i primjena tri različita pristupa za rješavanje problema minimalnog težinskog dominantnog skupa u usmjerenom grafu, problem će biti rješavan uz pomoć genetskog algoritma (eng. genetic algorithm - GA), pohlepnog algoritma (eng. greedy algorithm), kao i pomoću metode cjelobrojnog linearnog programiranja (eng. integer linear programming - ILP).

Napominjemo da cilj ovog rada nije postizanje boljih rezultata u odnosu na radove koji su do objavljeni do trenutka pisanja ovog rada.

Opis problema

Minimalni težinski usmjereni dominantni skup (eng. Minimum weighted directed dominating set - MWDDS) je optimizacioni problem koji je vezan za pronalaženje minimalnog težinskog dominantnog skupa u usmjerenom grafu. Za svaki usmjereni graf ***G = (V, E)***, sa skupom čvorova ***V*** i grana ***E***, usmjereni dominantni skup S_d je podskup skupa čvorova, tako da svaki čvor grafa ***G*** pripada S_d ili da može biti dosegnut iz najmanje jednog čvora koji je u S_d.

Navedeni MWDDS problem zahtjeva da takav podskup bude minimalan, u smislu da je sastavljen od čvorova sa najmanjom težinom, što znači da je svakom čvoru u grafu pridružena određena težina ***w*** i to dodatno komplikuje problem jer se u obzir, pored usmjerenih grana, mora uzeti i težina svakog čvora. U nastavku ćemo prikazati jedan primjer usmjerenog težinskog grafa i njegov dominantni skup, za ilustraciju su korišćeni Python paketi Matplotlib i NetworkX.



Slika 1. primjer usmjerenog težinskog grafa i njegovog dominantnog skupa

Na ilustraciji je prikazan graf G = (V, E), graf se sastoji od pet čvorova i 5 usmjerenih grana, gdje su čvorovi označeni brojevima i imaju pridružene težine koje su označene sa prefiksom „w“. Na primjer, čvor 1 ima težinu 10, čvor 2 ima težinu 25, i tako dalje. Strelice na granama prikazuju smjer, tako na primjer, postoji grana od čvora 1 do čvorova 3 i 5, od čvora 2 do 5, itd.

Na desnoj ilustraciji, crvenom bojom su prikazani oni čvorovi koji pripadaju dominantnom skupu S_d. Za dati graf, dominantni su čvorovi 1, 2 i 5, odnosno oni formiraju MWDDS težine 78, čvor 1 se nalazi u ovom skupu jer je to čvor sa najmanjom težinom koji pokriva čvor 3 (u suprotnom bi čvor 3 morao biti uključen u S_d što nije optimalno jer je njegova težina 38), čvor 2 je uključen u S_d jer je on izvorni čvor (source node) odnosno čvor koji ne može biti dosegnut iz nekog drugog čvora i zbog toga mora biti u ovom skupu, u suprotnom skup ne bi bio dominantan, a čvor 5 je u S_d jer ima manju težinu u odnosu na čvor 4. Pored ovog rješenja, za dati graf postoje i neka druga rješenja ali predstavljeno je ono sa najmanjom težinom.

Ovaj problem pripada klasi NP-teških optimizacionih problem, što znači da nije poznat efikasan algoritam za pronalaženje tačnog rješenja za sve vrste grafova.

MWDDS se koristi u oblastima koje uključuju usmjerene veze (rutiranje mreža, rudarenje podataka, itd.).

Genetski algoritam

Genetski algoritam (GA) je heuristički pristup za rješavanje različitih optimizacionih problema, ovaj algoritam spada u klasu evolutivnih algoritama i često se koriste za aproksimaciju i traženje optimalnih rješenja u prostorima velikih dimenzija, pa se zbog toga može koristiti za rješavanje MWDDS problema.

U nastavku će biti opisan pristup koji smo mi koristili:

1. **Inicijalizacija populacije** – GA počinje generisanjem inicijalne populacije jedinki koje predstavljaju potencijalna rješenja problema. Jedinka, u ovom slučaju predstavlja skup čvorova koji formiraju dominantni skup, i definisana je na način da su pozicije onih čvorova koji su uključeni u dominantni skup označene sa 1, u suprotnom je na njihovoj poziciji 0.
   Inicijalna populacija sadrži:
   - 30% od ukupnog broja jedinki, sadrži izvorne čvorove (čvorovi koji ne mogu biti dosegnuti iz drugih čvorova) grafa, dok za ostale čvorove u tom rješenju postoji jednaka vjerovatnoća da budu dio tog rješenja
   - ostatak jedinki se generiše na slučajan način tako da ne sadrže izvorne čvorove

   Generisane jedinke se ocjenjuju uz pomoć funkcije cilja (fitness function), koja svakoj jedinki dodjeljuje cijenu, odnosno u našem slučaju je to zbir svih težina uključenih čvorova ukoliko je potencijalno rješenje validno, u suprotnom mu se dodjeljuje zbir svih težina čvorova grafa. Validnost rješenja tj. jedinke, se provjerava pomoću bojenja čvorova, slično načinu koji je opisan u radu (Nakkala 2019) [1], uz popravljanje rješenja, u zavisnosti od vjerovatnoće, tako da dobijemo skup koji je dominantan.

2. **Selekcija** – nakon inicijalizacije algoritam prelazi na proces selekcije kako bi odabrao podskup jedinki za reprodukciju. U našoj implementaciji smo koristili jednostavnu turnirsku selekciju, iz inicijalne populacije prvo, na slučajan način, odaberemo 50% jedinki nad kojima se izvršavaju ostale operacije. Napominjemo da smo u ranijim fazama testiranja koristili i metod ruleta (roulette wheel) ali smo se na kraju zbog dobijenih rezultata odlučili za korišćenje samo turnirske selekcije.

3. **Ukrštanje** – selektovane jedinke prolaze kroz proces ukrštanja sa određenom vjerovatnoćom, koristili smo jednopoziciono ukrštanje (single-point crossover) i koristeći metodu slučajnog odabira dolazimo do pozicije presjeka.

4. **Mutacija** – poslije procesa ukrštanja, novonastale jedinke sa određenom vjerovatnoćom (inicijalno je to 5% ali zavisi i od dimenzije problema) prolazi kroz proces mutacije gdje se sa određenom vjerovatnoćom (50% u našem slučaju) mijenjaju pojedini geni u hromozomu jedinke. Treba napomenuti da vjerovatnoću mutacije jedinke povećavamo za određeni faktor u slučaju kada nema poboljšavanja ciljane vrijednosti (fitness value) kroz određeni broj generacija (ovi parametri se podešavaju – vjerovatnoća mutacije jedinke, faktor za povećanje vjerovatnoće, broj generacija koji se prati), dok je vjerovatnoća mutacije pojedinog gena fiksirana.

5. **Ciljana vrijednost** – nakon reprodukcije i mutacije, svaka jedinka se ocjenjuje uz pomoć funkcije cilja, onako kako je opisano na početku.

6. **Kriterijum zaustavljanja** – algoritam se izvršava kroz više generacija, u zavisnosti od skupa problema, i ima nekoliko kriterijuma zaustavljanja – maksimalni broj generacija, vremensko ograničenje na 10 minuta, a ostavljena je mogućnost prekidanja algoritma nakon određenog broja generacija bez poboljšanja ciljane vrijednosti.

Pohlepni algoritam

Pohlepni algoritam je heuristički pristup za rješavanje problema koji se zasniva na trenutno optimalnom izboru u svakom koraku, bez razmatranja šire slike problema koji se rješava, pa zbog toga ne možemo očekivati optimalno rješenje.

Pohlepni algoritam koji smo mi koristili prilikom rješavanja MWDDS problema se zasniva na algoritmu za aproksimaciju koji je predstavljen u knjizi „Approximation Algorithms“ [2].

Postupak:

1. **Inicijalizacija varijabli** – na početku inicijalizujemo prazan skup *domination_set* koji će sadržati čvorove dominirajućeg skupa i skup *remaining_vertices* koji inicijalno sadrži sve čvorove iz grafa, iz ovog skupa ćemo postepeno izbacivati čvorove prilikom formiranja dominantnog skupa.

2. **Glavna petlja** - algoritam ulazi u glavnu petlju koja se izvršava dok ima preostalih čvorova u *remaining_vertices* skupu.

3. **Izbor sljedećeg čvora** - algoritam bira čvor koji će dodati u dominantni skup na osnovu pohlepne strategije. Za svaki preostali čvor *v*, algoritam računa cijenu (cost) koji zavisi od težine čvora *v* i broja njegovih nasljednika (oni čvorovi do kojih postoji direktna grana iz čvora *v*), na taj način favorizujemo čvorove sa manjom težinom u odnosu na broj nasljednika. Čvor sa najmanjom cijenom se dodaje u skup *domination_set*.

4. **Uklanjanje čvora i njegovih nasljednika iz preostalih čvorova** – po dodavanju čvora u dominantni skup, moramo osigurati da on i njegovi nasljednici neće biti predmet razmatranja u budućim iteracijama, zbog toga ih uklanjamo iz skupa *remaining_vertices*.

5. **Računanje rješenja** – po izlasku iz glavne petlje, računamo ukupnu težinu rezultujućeg dominantnog skupa tako što sumiramo sve težine čvorova koji su uključeni u *domination_set*

Cjelobrojno linearno programiranje

Cjelobrojno linearno programiranje (ILP) je matematička metoda za rješavanje optimizacionih problema koji se modeliraju pomoću linearne funkcije i linearnih ograničenja. Za rješavanje MWDDS problema smo koristili LP modeler PuLP.

Opis model koji smo koristili:

1. **Definisanje varijabli** – varijable koje predstavljaju čvorove grafa ***G=(V, E)*** smo označili sa ***x,*** i svaka varijabla x_i ima vrijednost 0 ili 1, u zavisnosti da li je ***i***-ti čvor uključen u dominantni skup S_d.

2. **Funkcija cilja** – cilj je minimizovati težinu dominantnog skupa, ovo se postiže definisanjem ciljane funkcije kao sume težina čvorova koji pripadaju dominantnom skupu:

$$\boldsymbol{min}\left\{\sum_{i}^{V} x_i * w_i\right\}$$

   x_i **–** ***i***-ti čvor i pripadajuća težina w_i

3. **Ograničenja** - da bi se osiguralo da je formirani skup dominantan, potrebno je postaviti odgovarajuće ograničenje, za svaki čvor ***i*** u grafu ***G=(V, E)*** je definisano ograničenje koje garantuje da ako je čvor ***i*** uključen u S_d ($x_i = 1$), barem jedan od njegovih ulaznih susjeda mora biti uključen u S_d. Ovo se postiže formulisanjem ograničenja za svaki čvor:

$$\forall i, j \in V : \sum_{(i,j)\in E} x_j > x_i$$

   Nakon definisanja varijabli, ciljane funkcije i ograničenja, koristili smo PuLP-ov omotač (wrapper) za **CBC** (COIN Branch and Cut) rješavač sa vremenskim ograničenjem od 10 minuta.

Eksperimentalni rezultati

Za testiranje predstavljenih pristupa za rješavanje MWDDS problema koristili smo testne instance koje smo mi generisali, a pored toga i male, srednje i velike instance koje su iz rada (Jovanovic, Tuba and Simian 2010) [3] ali su od njih formirani usmjereni grafovi (na način da umjesto grana *(u, v)* i *(v, u)* u skup grana *E* dodamo samo jednu od njih). Za svaku veličinu problema su odabrane neke instance sa određenim brojem čvorova i grana, težine za čvorove su iz opsega [20, 70] (osim za testne instance).

Sva testiranja su izvršena pod Linux operativnim sistemom sa sljedećim specifikacijama:

- Intel Core i5-8250U na radnom taktu od 2.6GHz
- 36GB RAM

Vremena izvršavanja su izražena u sekundama.

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
test_5_6	78	78	0	0	80	0	78	0.007
test_5_8	114	114	0	0	212	0	114	0.005
test_5_10	79	79	0	0	115	0	79	0.011
test_9_12	263	291	17.1	0.001	361	0	244	0.008
test_5_13	103	103	0	0.001	177	0	103	0.007

Tabela 1 – rezultati za testne instance

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
V50E50	990.2	1040	37.38395	1.6488	1124	0.026	814	0.009
V50E100	813.2	887	37.53079	1.6866	1084	0.052	668	0.011
V50E250	459.6	548	41.30908	1.5857	613	0.074	357	0.032
V150E150	3231.7	3387	118.6171	6.1681	3322	1.626	2285	0.015
V150E250	3038.5	3174	110.3442	6.6249	2782	2.17	1890	0.017
V150E500	2277	2556	161.3536	6.919	1952	2.562	1244	0.061
V250E250	6023.7	6232	170.4752	13.4663	5557	12.748	3846	0.02
V250E500	5329.6	5838	244.2602	14.222	4675	17.93	2952	0.026
V250E750	4767	4953	119.1327	14.3403	4182	19.241	2350	0.029

Tabela 2 – rezultati za male instance

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
V50E500	162.8	168	2.315167	7.3596	342	0.05	161	0.022
V50E750	132.4	138	2.8	7.3847	291	0.045	131	0.014
V50E1000	104.2	106	0.6	6.8597	214	0.038	104	0.014
V150E750	1132.1	1235	56.37987	34.1636	1508	2.625	866	0.042
V150E1000	931.2	1016	57.52878	33.4007	1246	2.707	726	0.569
V150E2000	534.1	593	33.4229	40.3381	850	2.627	428	0.207
V250E1000	2736.3	3028	171.6852	110.8634	3497	20.719	1902	0.058
V250E2000	1726.2	2016	120.6713	105.5266	2314	22.126	1150	0.106
V250E3000	1413.2	1548	72.20083	106.5668	1562	22.03	824	1.004

Tabela 3 – rezultati za srednje instance

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
V500E1000	9654.7	10145	218.7515	265.1614	8429	238.267	5678	0.056
V500E2000	6837.1	7446	319.5198	268.3955	6160	304.073	3606	0.21
V500E5000	4087.7	4362	131.6177	279.8065	3582	348.533	1786	3.718
V800E1000	27033.4	27741	2122.8	360.785	16064	1250.566	10864	0.079
V800E2000	21952.1	27741	5798.128	361.098	12284	1691.297	7880	0.122
V800E5000	10649.1	11790	574.1966	361.1503	7394	2163.48	4412	0.702
V1000E1000	34260	34260	0	361.7012	20994	2746.765	14660	0.199
V1000E5000	15961.3	17143	547.6389	362.335	10724	5425.016	6512	1.089
V1000E10000	11953.9	12856	357.4629	361.1336	7324	6346.143	3924	28.95

Tabela 4 – rezultati za velike instance

Odlučili smo se za generisanje testnih grafova sa malim brojem čvorova i grana kako bismo uradili provjeru korektnosti rezultata, na osnovu rezultata koji su predstavljeni u tabeli 1, kada posmatramo rješenja koja su dobijena pomoću PuLP-a možemo vidjeti da implementirani GA daje optimalna rješenja za probleme manjih dimenzija, dok pohlepni algoritam ne daje najoptimalnije rezultate.

PuLP rješavač je bio ograničen na 600 sekundi, dok je rješenje sa GA traženo u 10 iteracija sa ograničenjem od 360 sekundi po svakoj instanci, sa rastom dimenzije problema smo podešavali i određene parametre kao što su: broj generacija, veličina populacije, vjerovatnoća mutacije, faktor za povećanje vjerovatnoće mutacije, korak za povećanje mutacije, kao i maksimalan broj generacija bez poboljšanja ciljane funkcije.

Dakle, iz predstavljenih rezultata zaključujemo da se PuLP rješavač pokazao dosta dobro kod svih slučajeva, dok se implementirana verzija GA pokazala dobrom za veoma male instance problema, a pohlepni algoritam je po rezultatima između njih.

Zaključak i budući rad

U ovom radu opisali smo pristup rješavanju problema pronalaženja minimalnog dominantnog skupa u usmjerenom težinskom grafu, uz pomoć genetskog algoritma, cjelobrojnog linearnog programiranja i pohlepnog algoritma. Izbor pristupa za rješavanje ovog problema treba da zavisi od veličine problema i ograničenja u pogledu dostupnih resursa i vremena. Mi nismo uspjeli implementirati GA koji će za kratko vremensko ograničenje, kao i za dati broj generacija i veličinu populacije, brzo konvergirati ka optimalnom rješenju problema. Sigurni smo da se efikasnost GA za ovaj problem može poboljšati drugačijim pristupom za generisanje inicijalne populacije (npr. korišćenjem različitih heuristika) i navođenjem na optimalno rješenje, kao i naprednim i sistematičnim podešavanjem ulaznih parametara.

Izvorni kod, instance problema, eksperimentalni rezultati, različite vizualizacije i drugi resursi su dostupni na GitHub repozitorijumu - https://github.com/neuralmaticv/operations-research-project_mwdds/

Reference i literatura

1. Nakkala, M.R., Singh, A. 2019. “Heuristics for Minimum Weight Directed Dominating Set Problem.” In *Futuristic Trends in Networks and Computing Technologies*, 494-507. Chandigarh: Springer .
2. Vazirani, Vijay V. 2001. *Approximation Algorithms.* Springer.
3. Jovanovic, R, M Tuba, and D. Simian. 2010. “Ant colony optimization applied to minimum weight dominating set problem.” *International Conference on Automatic Control, Modelling and Simulation.* World Scientific and Engineering Academy and Society. 322-326.
4. Đukanović, M, and Matić D. 2022. *Uvod u operaciona istraživanja.* Prirodno-matematički fakultet, Univerzitet u Banjoj Luci.