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

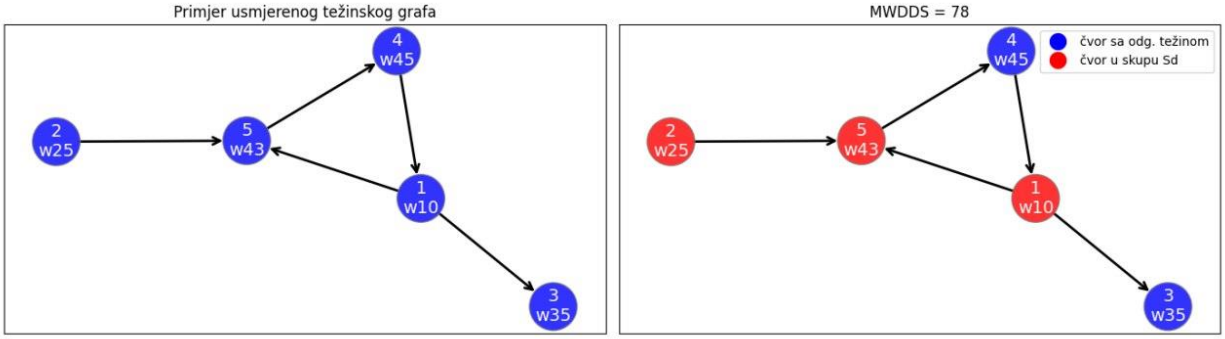
Cilj ovog seminarskog rada je testiranje i primjena tri različita pristupa za rješavanje problema minimalnog težinskog dominantnog skupa u usmjerenom grafu, problem će biti rješavan uz pomoć genetskog algoritma (eng. genetic algorithm - GA), pohlepnog algoritma (eng. greedy algorithm), kao i pomoću metode 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, pod pretpostavkom da je P ≠ NP.

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, uz pomoć heuristika 1, 2 i 3 koje su opisane u radu (Nakkala 2019) [1], uz popravljanje rješenja, u zavisnosti od vjerovatnoće, tako da dobijemo skup koji je dominantan.

2. **Elitizam –** u novu populaciju prebacujemo *k* najboljih jedinki (zadana vrijednost za *k* je 2)

3. **Selekcija** – nakon inicijalizacije algoritam prelazi na proces selekcije kako bi odabrao podskup jedinki za reprodukciju. U našoj implementaciji smo koristili turnirsku selekciju, zadana veličina grupe za odabir je 4, iz inicijalne populacije prvo odaberemo jedinke nad kojima će se izvršavati 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.

4. **Ukrštanje** – selektovane jedinke prolaze kroz proces ukrštanja sa određenom vjerovatnoćom (inicijalno 0.95), koristili smo jednopoziciono i dvopoziciono ukrštanje (zadana vjerovatnoća za dvopoziciono ukrštanje je 0.9), a koristeći metodu slučajnog odabira dolazimo do pozicije presjeka.

5. **Mutacija** – poslije procesa ukrštanja, novonastale jedinke sa određenom vjerovatnoćom (inicijalno je to 0.05) prolazi kroz proces mutacije gdje se sa određenom vjerovatnoćom (0.5 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.

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

7. **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 binarna 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 ukupnu težinu dominantnog skupa, ovo se postiže definisanjem ciljane funkcije kao sume težina čvorova koji pripadaju dominantnom skupu:

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

   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 v_i u grafu ***G=(V, E)*** treba da bude dio usmjerenog dominantnog skupa ili treba da ima barem jednog ulaznog susjeda (roditelja) koji je dio dominantnog skupa. To se definiše pomoću ove nejednakosti:

$$\boldsymbol{x_i} + \sum\nolimits_{\boldsymbol{v_j} \in \boldsymbol{N^-_{G\,(vi)}}} \boldsymbol{x_j} \geq \boldsymbol{1} \;\; \forall \boldsymbol{v_i} \in \boldsymbol{V}$$

$$\boldsymbol{N^-_{G\,(vi)}} \; - \; \text{označava ulazni stepen (indegree) čvora } \boldsymbol{v_i}$$

   Ovu formulaciju pojačavamo na sljedeći način, po uzoru na [5], za svaki čvor v_i neka:

$$z(v_i) = 1 + |N + G(v_i)| > 0$$

   bude maksimalan broj čvorova koji mogu biti pokriveni čvorom v_i. Pošto je potrebno pokriti sve čvorove, definišemo i sljedeće ograničenje:

$$\sum_{\boldsymbol{v_i} \in \boldsymbol{V}} \boldsymbol{z}\,(\boldsymbol{v_i})\boldsymbol{x_i} \geq \boldsymbol{n}$$

   Neka ***Z*** bude skup svih različitih parova $z(v_i)$, prethodna ograničenja možemo pojačati sa:

$$\sum_{\boldsymbol{v_i} \in \boldsymbol{V}} \left\lceil \frac{\boldsymbol{z_{(vi)}}}{\boldsymbol{q}} \right\rceil \boldsymbol{x_i} \geq \left\lceil \frac{\boldsymbol{n}}{\boldsymbol{q}} \right\rceil \quad \forall \boldsymbol{q} \in \boldsymbol{Z}$$

   Ovim dodatnim ograničenjima pojačavamo formulaciju problema kako bi se poboljšala efikasnost ILP rješavača pri rješavanju različitih instanci problema. 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
V5E6	78	78	0	0.1066	80	0	78	0.0064
V5E8	114	114	0	0.0839	212	0	114	0.0055
V5E10	79	79	0	0.0798	115	0	79	0.007
V9E12	244	244	0	0.1287	361	0.0002	244	0.0089
V5E13	103	103	0	0.0851	177	0	103	0.0069

Tabela 1 – rezultati za testne instance

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
V50E50	814	814	0	3.5716	1124	0.027	814	0.0133
V50E100	668.4	672	1.2	2.9699	1084	0.0362	668	0.0177
V50E250	357.4	358	0.489898	2.4415	613	0.0402	357	0.042
V150E150	2295	2316	10.8259	21.7394	3322	1.5211	2285	0.073
V150E250	1924.8	1975	29.11288	20.0036	2782	1.8463	1890	0.1067
V150E500	1308.7	1375	35.87213	17.1767	1952	2.5383	1244	0.3301
V250E250	3870.9	3910	21.85155	68.8508	5557	13.0398	3846	0.2667
V250E500	3001.2	3094	48.70483	60.3758	4675	16.5351	2952	0.4971
V250E750	2411.8	2474	27.69043	54.1388	4182	21.1484	2350	0.7888

Tabela 2 – rezultati za male instance

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
V50E500	161	161	0	21.5356	342	0.0519	161	0.0642
V50E750	131	131	0	21.2449	291	0.0476	131	0.0605
V50E1000	104	104	0	19.0988	214	0.0385	104	0.0802
V150E750	881	893	11.19821	121.959	1508	2.6159	866	0.3358
V150E1000	736.5	738	1.5	114.4981	1246	2.6573	726	1.1241
V150E2000	450.3	460	11.74777	102.4953	850	2.5234	428	0.9351
V250E1000	1958.3	1961	0.9	414.4264	3497	20.4099	1902	0.9947
V250E2000	1181.5	1185	2.291288	314.4318	2314	21.5392	1150	1.9046
V250E3000	865.8	871	9.610411	293.0337	1562	21.3569	824	4.2341

Tabela 3 – rezultati za srednje instance

instance	ga_mean	ga_max	ga_std	ga_time	greedy	greedy_time	ilp	ilp_time
V500E1000	5717.7	5721	4.22019	615.4095	8429	231.2315	5678	3.7483
V500E2000	3675	3699	24.32014	614.5641	6160	292.5303	3606	7.5413
V500E5000	1825.7	1833	5.330103	605.6584	3582	342.6431	1786	22.2794
V800E1000	10918.4	11411	193.103	657.0487	16064	1245.0842	10864	9.5204
V800E2000	7911	7927	8	627.7354	12284	1736.105	7880	19.5504
V800E5000	4535.1	5434	389.6384	627.9002	7394	2233.996	4412	48.7607
V1000E1000	14667	14667	0	754.7649	20994	2845.0072	14660	16.3556
V1000E5000	6792.3	7671	249.749	680.4351	10724	6192.2993	6512	77.6117
V1000E10000	4124	4149	12.5	665.7487	7324	6154.1804	3924	174.1637

Tabela 4 – rezultati za velike instance

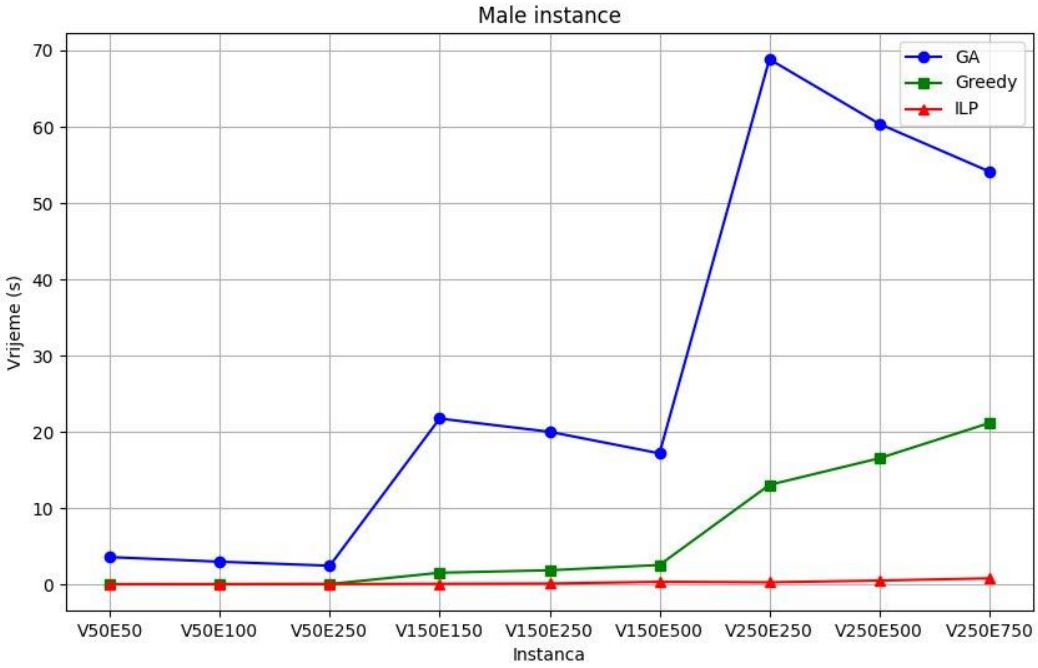
Odlučili smo se za generisanje testnih grafova sa malim brojem čvorova i grana kako bismo uradili provjeru korektnosti rezultata, na osnovu rezultata koji su predstavljeni u tabeli 1, kada posmatramo rješenja koja su dobijena pomoću PuLP-a možemo vidjeti da implementirani GA daje optimalna rješenja za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

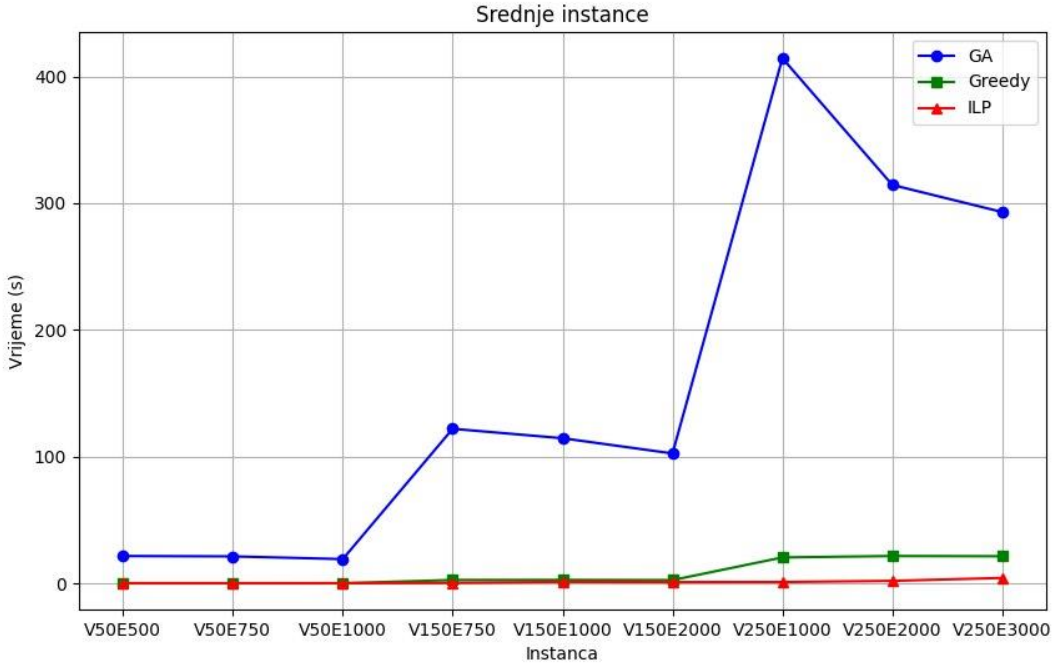
PuLP rješavač je bio ograničen na 600 sekundi, dok je rješenje sa GA traženo u 10 iteracija sa ograničenjem od 600 sekundi po svakoj instanci, sa rastom dimenzije problema smo podešavali i određene parametre kao što su: broj generacija, veličina populacije, broj elitnih jedinki, veličina turnira, vjerovatnoća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

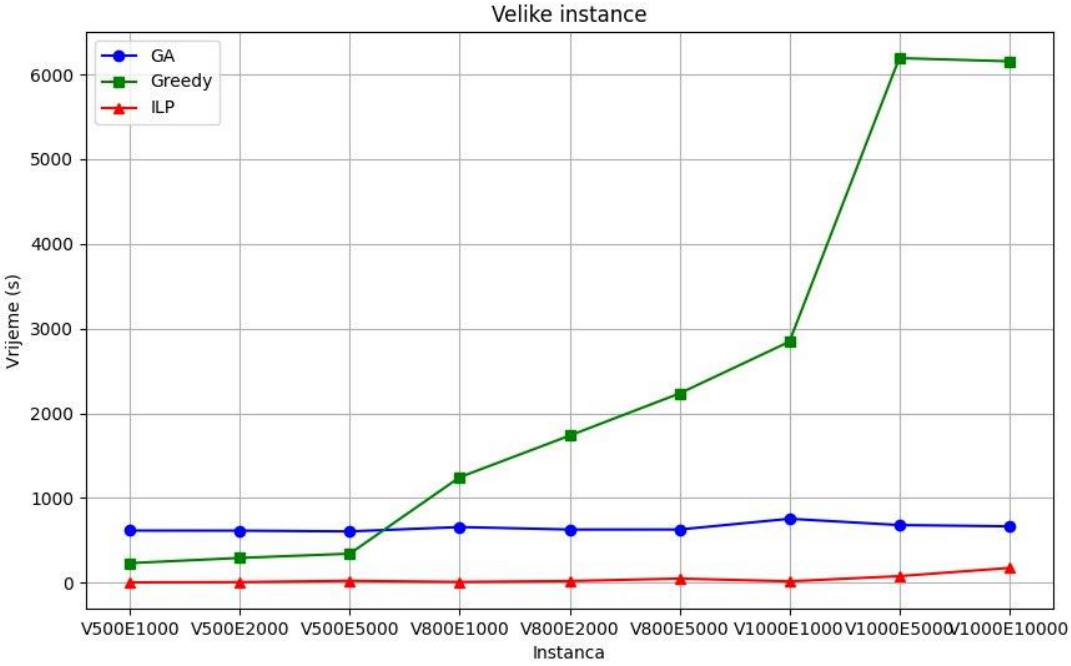
Na osnovu prikazanih rezultata, zaključujemo da se PuLP rješavač dobro pokazao za sve dimenzije problema u pogledu pronalaska rješenja. Takođe, vrijeme izvršavanja PuLP rješavača je relativno konstantno, osim kod većih dimenzija problema. Slična situacija je i sa genetskim algoritmom u pogledu vremena izvršavanja, ali se isto tako veoma dobro pokazao za sve dimenzije problema. Veća odstupanja između rezultata se javljaju kod 3 velike instance, u tim slučajevima GA nije uspio da se značajnije približi

optimalnim rješenjima. Metode ILP i GA su se pokazale efikasnijim od pohlepnog algoritma, što je i očekivano.

Na narednim slikama se može vidjeti uporedni prikaz zavisnosti vremena izvršavanja od broja čvorova, tj. grana.




Srednje instance
GA
Greedy
ILP
Vrijeme (s)
0
100
200
300
400
V50E500
V50E750
V50E1000
V150E750
V150E1000
V150E2000
V250E1000
V250E2000
V250E3000
Instanca


Velike instance
GA
Greedy
ILP
Vrijeme (s)
0
1000
2000
3000
4000
5000
6000
V500E1000
V500E2000
V500E5000
V800E1000
V800E2000
V800E5000
V1000E1000
V1000E5000
V1000E10000
Instanca

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. Pokazali smo da se efikasnost genetskog algoritma može poboljšati uz korišćenje odgovarajućih heuristika prilikom generisanja inicijalne populacije, a i kasnije prilikom stvaranja novih populacija. Vjerujemo da se efikasnost predloženog hibridnog genetskog algoritma može dodatno poboljšati naprednim i sistematičnim podešavanjem ulaznih parametara, kao i paralelizacijom predloženog GA kako bi se izvršavao na multiprocesorskom računaru ali to ostavljamo za neki budući rad.

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.
5. Nakkala, M.R., Singh, A., Rossi, A. (2022). Swarm intelligence, exact and matheuristic approaches for minimum weight directed dominating set problem. Engineering Applications of Artificial Intelligence, Volume 109, March 2022, 104647.