UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Branko Grbić

PROBLEM MAKSIMIZACIJE DUŽINE ŽIVOTNOG VEKA BEŽIČNIH SENZORSKIH MREŽA I METODE ZA NJEGOVO REŠAVANJE

master rad

Beograd, 2023.

Mentor:

dr Zorica Stanimirović, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Zorica Stanimirović, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Zoran Stanić, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Zorica Dražić, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: 04.07.2023.

Naslov master rada: Problem maksimizacije dužine životnog veka bežičnih senzorskih mreža i metode za njegovo rešavanje

Rezime: Jedan od često izučavanih koncepata u teoriji grafova su dominantni skupovi. Dominantan skup predstavlja podskup skupa čvorova grafa za koji važi sledeće: svaki čvor grafa se nalazi u tom dominantom skupu ili je sused sa nekim čvorom iz tog dominantnog skupa. Dominantni skupovi imaju značajnu primenu pri rešavanju problema produžavanja životnog veka bežičnih senzorskih mreža. Naime, pridruživanjem senzorskih čvorova disjunktnim dominantnim skupovima je jedan od načina produžavanja životnog veka bežičnih senzorskih mreža. Nakon pridruživanja, dominantni skupovi se sekvencijalno aktiviraju mehanizmom cikličnog paljenja i gašenja u cilju sačuvanja energije. U ovom radu je razmatran problem produžavanja životnog veka bežičnih senzorskih mreža (engl. *Maximization lifetime of wireless sensor networks - MWDDS*). Predložene su dve nove modifikacije postojeće pohlepne heuristike iz literature za rešavanje problema MWDDS. U cilju evaluacije novih, modifikovanih pohlepnih heuristika izvršena su opsežna testiranja na skupu od 640 test instanci. Takođe, izvršeni su preliminarni eksperimenti u cilju određivanja adekvatnih vrednosti parametara heuristika. Dobijeni eksperimentalni rezultati su pokazali da su dve novopredložene pohlepne heuristike dale bolje rezultate u odnosu na rezultate postojećih metoda iz literature za rešavanje ovog problema.

Ključne reči: maksimizacija životnog veka; bežične senzorske mreže; pohlepne heuristike; disjunktni dominantni skupovi, problem MWDDS

Master thesis title: The problem of maximization lifetime of wireless sensor networks and methods for its solving

Abstract: One of the well researched concepts in graph theory are dominating sets. A dominating set represents subset of vertices such that every node of graph is either in that dominating set or has a neighbor in that dominating set. Dominating sets have important application when solving the problem of maximization lifetime of wireless sensor networks. Assigning node sensors to disjoint dominating sets may be the one way to increase network lifetime in wireless sensor networks. After assigning, dominating sets are then sequentially activated by a sleep–wake cycling mechanism in order to save energy of network. In this paper, the problem of maximization lifetime of wireless sensor networks *(MWDDS problem)* is considered. Two new modifications of greedy heuristic from the literature are proposed. For the purpose of evaluation of new modified greedy heuristics, extensive testing on set of 640 test instances is performed. In addition, preliminary experimental testing for determination of parameters for new heuristics is executed. Obtained experimental results show that our two new modified versions of greedy heuristic gave better results than existing methods from the literature regarding solution quality.

Keywords: lifetime maximization; wireless sensor networks; greedy algorithm; disjoint dominating sets; MWDDS problem

Sadržaj

Glava 1

Uvod

1.1 Osnovni pojmovi o grafovima

Neka je $G = (V, E)$ prost, neusmereni graf bez petlji. Skup V, pri čemu je $|V| = n$ je skup čvorova. $E \subseteq V \times V$ je skup ivica pri čemu je $|E| = m$. Za dva čvora $u, v \in V$ pri čemu je $u \neq v$ kaže se da su susedni ukoliko su povezani ivicom grafa - odnosno $(u, v) \in E$, inače su nesusedni. Za dati čvor $v \in V$ može se definisati skup $N(v) := \{u \in V | (v, u) \in E\}$ koji predstavlja skup suseda čvora v, odnosno otvorenu okolinu čvora v. Slično, sa $N[v] := N(v) \cup \{v\}$ može se definisati zatvorena okolina čvora v. Analogno, mogu se uvesti definicije otvorene i zatvorene okoline podskupa skupa čvorova. Ukoliko je $D \subseteq V$, otvorena okolina skupa D se definiše kao $N(D) := \cup_{v \in D} N(v)$. Slično, zatvorena okolina skupa D se definiše kao $N[D] := N(D) \cup D$. Iz definicija otvorene i zatvorene okoline skupa $D \subseteq V$, jasno je da važi $N(D) \subseteq N[D]$.

Definicija 1 (Dominantni skup)**.** *Podskup $D \subseteq V$ skupa čvorova prostog neusmerenog grafa bez petlji $G = (V, E)$ naziva se dominantni skup grafa G ako i samo ako svaki čvor $v \in V \setminus D$ ima barem jednog suseda iz skupa D.*

Može se primetiti sledeće: ukoliko je dat prost, neusmeren graf $G = (V, E)$, tada je skup V dominantni skup grafa G. Iz toga sledi da svaki prost, neusmeren graf G mora imati barem jedan dominantni podskup. Takođe, jasno je da je uslov dominantnosti skupa $D \subseteq V$ ekvivalentan sa $N[D] = V$.

Definicija 2 (Domatska particija)**.** *Skup $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$ podskupova $D_i \subseteq V$ naziva se domatska particija prostog neusmerenog grafa $G = (V, E)$, ukoliko je svaki*

podskup D_i $(i = 1, \ldots, k)$ dominantni skup od G i svi elementi skupa $\mathcal{D}$ su međusobno disjunktni, odnosno za sve $1 \leq i < j \leq k$ važi da je $D_i \cap D_j = \emptyset$.

Definicija 3 (Domatski broj)**.** *Domatski broj prostog, neusmerenog grafa $G = (V, E)$ je definisan kao $|\mathcal{D}^*|$ pri čemu je $|\mathcal{D}^*| := \max\{|\mathcal{D}| \,|\mathcal{D}$ je domatska particija od $G\}$.*

Naredna lema daje gornje ograničenje domatskog broja proizvoljnog grafa G. Lema je preuzeta iz rada [1].

Lema 1. *Neka je $G = (V, E)$ prost neusmeren graf. Domatski broj grafa G nije veći od $\delta + 1$, pri čemu je $\delta = min_{v \in V}|N(v)|$, odnosno najmanji stepen nekog čvora u grafu G.*

Dokaz. Neka za čvor $v \in V$ važi $v = \text{argmin}_{v \in V} |N(v)|$. Neka su $v_1, v_2, \ldots, v_\delta$ susedni čvorovi čvora v. Pretpostavimo suprotno: neka je $\mathcal{D} = \{D_1, D_2, \ldots, D_m\}$ domatska particija grafa G pri čemu je $m > \delta + 1$. Pošto su D_i, $i = 1, 2, ..., m$ disjunktni, tada svaki od čvorova $v, v_1, v_2, \ldots, v_\delta$ može pripadati najviše jednom dominantnom skupu D_i. Kako je još uz to i $m > \delta + 1$ - tada po Dirihleovom principu postoji $k \in \{1, ..., m\}$ tako da D_k ne sadrži nijedan od čvorova $v, v_1, v_2, \ldots, v_\delta$. Iz toga sledi da čvor v ne pripada D_k kao i da čvor v nema nijednog suseda koji se nalazi u skupu D_k iz čega sledi da D_k nije dominantan skup jer $v \notin N[D_k] = V$. Kontradikcija. □

1.2 Problemi dizajniranja bežičnih senzorskih mreža

Bežične senzorske mreže (engl. *Wireless sensor networks - WSN*) su u poslednjih 20 godina sve popularnije zbog njihove primene u brojnim oblastima poput medicine, građevine, monitoringa životne sredine, saobraćaja, poljoprivredi kao i mnogim drugim oblastima. Bežične senzorske mreže predstavljaju skup minijaturnih jeftinih senzorskih čvorova (senzora) koji mogu da se postave u gotovo svim uslovima i sredinama u kojima potpuno nezavisno, samostalno formiraju mrežnu strukturu u okviru koje su senzori sposobni da međusobno prikupljaju, obrađuju i razmenjuju podatke i da iste prosleđuju nadređenim računarima. Životni vek senzorskog čvora se najčešće izračunava kao količnik kapaciteta baterije i njegove prosečne potrošnje. Senzorski čvorovi su najčešće uređaji koji rade na baterije, odnosno njihov životni

vek je ograničen. Stoga, produžavanje životnog veka senzorskog čvora predstavlja ključni problem kod bežičnih senzorskih mreža.

Životni vek bežičnih senzorskih mreža predstavlja dužinu trajanja (vreme) u kojem mreža može nesmetano da funkcioniše i da izvršava operacije za koje je osposobljena. Stoga, životni vek WSN-a značajno zavisi od potrošnje energije senzorskih čvorova. Svaki senzor ima ograničeno vreme tokom kojeg može biti aktivan, dok mreža može nesmetano da funkcioniše u posmatranom periodu ukoliko je u svakom trenutku tog perioda cela mreža pokrivena aktivnim senzorima. Za WSN se kaže da je pokrivena aktivnim senzorima, ukoliko za svaki senzor te mreže važi jedno od sledeća dva uslova:

- senzor je aktivan (upaljen).
- senzor je neaktivan (ugašen) ali postoji aktivan senzor u mreži koji ima komunikaciju s tim senzorom.

Jedan od glavnih problema sa kojima se susrećemo kod WSN je maksimizacija dužine njenog životnog veka (engl. *Maximization the lifetime of WSN*).

Prema radu [2], strategije za uštedu potrošnje energije mogu biti klasifikovane na sledeći način:

- Sistem cikličnog paljenja i gašenja. (engl. *Sleep-wake cycling*);
- Podešavanje radijusa prenosa između bežičnih čvorova radi smanjenja potrošnje energije;
- Efikasno usmeravanje energije kao i prikupljanje podataka;
- Redukovanje količine prenosa podataka i izbegavanje redundantnih aktivnosti.

U ovom radu je korišćena prva strategija, odnosno mehanizam *Sleep-wake cycling*. Ova strategija rešava problem produžavanja životnog veka WSN-a tako što grupiše senzore u disjunktne podskupove i sukcesivno ih aktivira. U svakom trenutku, jedan od disjunktnih podskupova je aktivan. To znači da su svi senzori iz tog podskupa aktivni, dok su svi senzori koji ne pripadaju tom podskupu ugašeni.

Bežična senzorska mreža može biti predstavljena preko prostog neusmerenog grafa $G = (V, E)$, pri čemu svaki senzor predstavlja čvor grafa G, dok ivica povezuje dva čvora, ukoliko je uspostavljena komunikacija između dva senzora koji su predstavljeni sa ta dva čvora. Uslov pokrivenosti mreže nekim podskupom aktivnih

senzorskih čvorova je zagarantovan ukoliko je taj podskup čvorova dominantan skup grafa G.

Problem maksimizacije dužine životnog veka bežične senzorske mreže je izučavan u literaturi korišćenjem različitih pristupa. Najviše pristupa rešavanju ovog problema bilo je svođenjem na problem $k-$pokrivanja skupa ili problema domatske particije. Slijepčević i Potkonjak [3] su bili prvi koji su pristupili ovom problemu preko problema $k-$pokrivanja skupa i dokazali su da je problem NP težak koristeći polinomijalne vremenske transformacije problema od problema minimalnog pokrivanja skupa. Zbog činjenice da je problem NP težak, u literaturi su predloženi brojni aproksimativni algoritmi za rešavanje ovog problema, kao i algoritmi za neke od njegovih varijanti. Neki od primera su: pohlepne heuristike [3], memetički algoritmi [4, 5, 6], kuku pretraga (engl. *Cuckoo search*) [7] i genetski algoritmi [8].

Najčešći pristup rešavanja problema maksimizacije dužine životnog veka bežičnih senzorskih mreža je korišćenjem problema domatske particije grafa. Cilj ovog problema je da se pronađe particija skupa čvorova prostog neusmerenog grafa, pri čemu su elementi particije disjunktni dominantni skupovi, a broj dominantnih skupova maksimalan. U literaturi, problem domatske particije je poznat i pod nazivom Problem maksimizacije disjunktnih dominantnih skupova (engl. *Maximum disjoint dominant sets problem - MDDS*). Problem MDDS, koji pripada velikoj i značajnoj familiji problema dominantnih skupova [9, 10, 11], pripada klasi NP-teških problema, što je dokazano u [12]. Pre nekoliko godina, Pino se u radu [13] fokusirao na težinsku verziju problema MDDS, koju je nazvao: Problem maksimizacije težinskih disjunktnih dominantnih skupova (engl. *Maximum weighted disjoint dominant sets problem - MWDDS*).

U problemu MWDDS, svakom čvoru ulaznog grafa dodeljena je težina koja predstavlja dužinu njegovog životnog veka. Životni vek dominantnog skupa se definiše kao minimum svih životnih vekova od čvorova koji pripadaju tom skupu. Cilj problema MWDDS je pronaći domatsku particiju tako da je suma životnih vekova odgovarajućih dominantnih skupova maksimalna. Pino je u radu [13] predstavio tri algoritma lokalne pretrage za problem MWDDS. Svaki od ova tri algoritma kao ulaz ima početno rešenje koje je dobijeno modifikovanom pohlepnom heuristikom iz rada [14]. Nakon toga, algoritam iterativno zamenjuje čvorove iz različitih dominantnih skupova, u nadi da će time produžiti ukupan životni vek mreže. Više o problemima dizajniranja bežičnih senzorskih mreža je predstavljeno u radu [15].

Ostatak rada je organizovan na sledeći način. U glavi 2 dat je pregled i opis nekih od metaheurističkih metoda kao i neki načini hibridizacije metaheuristika sa drugim algoritmima. U glavi 3 su uvedene definicije i notacije kako bi mogao biti predstavljen problem MWDDS a zatim će biti prikazan i matematički model celobrojnog linearnog progamiranja za ovaj problem. U glavi 4, biće predstavljena pohlepna GH-MWDDS heuristika za rešavanje problema MWDDS predložena u radu [15] kao i dva parametarska modifikovana pohlepna algoritma nastala modifikacijom GH-MWDDS algoritma. U glavi 5, parametri ovih algoritama biće analizirani i biće izabrane najbolje vrednosti. Dalje u glavi, biće prikazani rezultati dobijeni novim predloženim heuristikama koji su analizirani i upoređeni sa rezultatima dobijenim u radu [15]. U glavi 6 je prezentovan zaključak rezultata dobijenih u ovom radu kao i dalji pravci u kojim bi dalja istraživanja iz ovog rada mogla biti nastavljena.

Glava 2

Metaheurističke metode

Metaheurističke metode (*metaheuristike*) predstavljaju klasu optimizacionih algoritama koje se koriste u cilju nalaženja aproksimativnih rešenja za probleme optimizacije. Metaheurističke metode za razliku od egzaktnih metoda, ne garantuju optimalnost rešenja problema koji se razmatra. Heurističke metode predstavljaju optimizacione algoritme koje se određuju u zavisnosti od problema koji se rešava. Za razliku od njih, metaheuristike su bazirane na opštem skupu pravila i strategija. Kao takve, imaju široku namenu i mogu se uz određeno prilagođavanje primeniti na širok spektar optimizacionih problema. Ne postoji univerzalna metoda kojom se može odrediti koja je najefikasnija metaheuristika za dati optimizacioni problem. Metaheuristike se najčešće koriste za rešavanje NP-teških problema, kada egzaktne metode usled vremenskih ili memorijskih ograničenja ne mogu da pronađu optimalno rešenje.

U dizajniranju metaheuristika, postoje dve glavne strategije koje bi trebalo da se primenjuju:

- Istraživanje prostora pretrage (diverzifikacija). Ovom strategijom se ispituje u kojim regionima prostora pretrage se nalaze najbolja rešenja.
- Intenzifikacija procesa pretrage. Ova strategija se koristi radi pronalaska što boljeg rešenja u okviru jednog regiona.

Ove dve strategije treba iskombinovati tako da se postigne balans između diverzifikacije i intenzifikacije procesa pretrage prostora rešenja za razmatrani problem, što nije jednostavan zadatak.

Postoji nekoliko načina na koji metaheuristički algoritmi mogu biti klasifikovani u zavisnosti od karakteristika, kao što su:

- **S-metaheuristike i P-metaheuristike.** Metaheurističke metode na bazi jednog rešenja (engl. *Single-solution based algorithms* - S-metaheuristics) su metode koje su zasnovane na iterativnom poboljšanju jednog rešenja. Metaheurističke metode na bazi populacije rešenja (engl. *Population based algorithms* - P-metaheuristics) su metode koje istovremeno rade na skupu rešenja koje se naziva populacija. Ove dve familije metaheurističkih metoda imaju komplementarne osobine. S-metaheuristike su više orijentisane ka intenzifikaciji procesa pretrage, dok su P-metaheuristike više orijentisane ka istraživanju prostora pretrage.

- **Determinističke i stohastičke metaheurističke metode.** Kod stohastičkih metaheurističkih metoda, proces izvršavanja algoritma je randomizovan. Stoga, pokretanje algoritma više puta može dati različite rezultate. Veliki broj metaheuristika su upravo stohastičke metode. S druge strane, proces determinističkih metaheuristika je unapred definisan i određen.

- **Metaheuristike inspirisane prirodnim pojavama ili procesima.** Postoji veliki broj metaheurističkih metoda čije su strategije inspirane pojavama ili procesima iz prirode. Neke od njih su: ponašanje populacije jedinki (ptice, pčele, mravi), proces prirodne evolucije, kretanje rojeva čestica, način funkcionisanja živih organizama ili nekih procesa u njima (nervni sistem - neuronske mreže (engl. *neural networks*), DNK računar (engl. *DNA computing*), veštački imuni sistemi (engl. *artificial immune sustems*), ćelijski automat (engl. *celullar automaton*) i mnogi drugi.)

U daljem nastavku ove sekcije su prikazane neke od najpopularnijih metaheuristika koje se koriste u rešavanju optimizacionih problema. Više o metaheuristikama se može pronaći u radovima [16] i [17].

2.1 Pohlepni algoritmi

Pohlepni algoritmi (engl. *greedy algorithms*) su algoritmi koji u svakoj iteraciji donose lokalno optimalne odluke u nadi da će takve odluke dovesti do globalno optimalnog rešenja. Pohlepni algoritmi su deterministički algoritmi koji se koriste za rešavanje kombinatornih optimizacionih problema.

Na početku svakog pohlepnog algoritma kreće se od praznog rešenja. U svakom koraku iteracije bira se jedan po jedan element sve dok se ne konstruiše kompletno

dopustivo rešenje. Svaki element koji se bira je lokalno optimalan u odnosu na zadatu funkciju evaluacije, što znači da se za njegov odabir ne uzimaju u obzir budući koraci u razmatranje kao ni celokupna slika posmatranog problema. S obzirom na to da se u svakoj iteraciji bira samo jedan element, veliki deo prostora dopustivih rešenja će ostati nepretražen. Pseudo kod pohlepnog algoritma prikazan je Algoritmom 1:

Algoritam 1 Pohlepni algoritam

Output: Rešenje optimizacionog problema dobijeno pohlepnim algoritmom.

1: $sol \leftarrow \emptyset$
2: **while** nije konstruisano kompletno rešenje s **do**
3: $\quad e_i \leftarrow LokalnaHeuristika(E \setminus \{e | e \in sol\})$
4: $\quad$ **if** $sol \cup e_i$ dopustivo rešenje **then**
5: $\quad\quad sol \leftarrow sol \cup e_i$
6: $\quad$ **end if**
7: **end while**
8: **return** sol

Na početku izvršavanja pohlepnog algoritma rešenje *sol* se inicijalizuje na prazan skup (korak 1). Zatim se u svakoj iteraciji dodaje jedan po jedan element u tekuće rešenje sve dok se ne konstruiše kompletno dopustivo rešenje (korak 2). U svakoj iteraciji se bira jedan element e_i koji je lokalno optimalan (korak 3). Ukoliko je $sol \cup e_i$ dopustivo rešenje problema, rešenje *sol* se ažurira dodavanjem elementa e_i (korak 5). Izlaz iz algoritma biće kompletno dopustivo rešenje *sol* (korak 8).

Može se primetiti da se u svakom koraku izvršavanja pohlepnih algoritama vrši strategija intenzifikacija procesa pretrage, dok diverzifikacija praktično i da ne postoji. Zbog toga, dobra ideja bi bila iskombinovati ideju pohlepnog algoritma sa metodom lokalne pretrage kako bi se dobio spoj diverzifikacije sa intenzifikacijom procesa pretrage. Jedan takav algoritam je GRASP metaheuristika o kojoj će biti više reči u narednoj podsekciji.

Neki od problema u čijem su rešavanju pohlepni algoritmi pokazali odlične performanse su: Hafmanovo kodiranje (engl. *Huffman encoding*), problem minimalnog razapinjućeg stabla (engl. *Minimal spanning tree problem*), problem trgovačkog putnika (engl. *Travelling salesman problem*), problem ranca (engl. *Knapsnack problem*) kao i mnogi drugi.

Iako pohlepni algoritmi često ne dolaze do optimalnih rešenja oni mogu proizvesti dobru aproksimaciju optimuma posmatranog problema. Prednost pohlepnih algoritama je u njihovoj efikasnosti i brzini izvršavanja. U mnogim situacijama, pohlepni algoritmi mogu dati rešenja visokog kvaliteta za relativno kratko vreme.

2.2 Ostale metaheuristike

Genetski algoritmi

Genetski algoritmi (engl. *Genetic algorithms - GA*) predstavljaju jedne od najpopularnijih evolutivnih algoritama koji pripadaju klasi P-metaheuristika. Osnovne postavke GA iznete su od strane Holanda 1975. godine u radu [18].

Genetski algoritam se zasniva na procesu prirodne evolucije jedne populacije jedinki pod dejstvom genetskih operatora. U fazi inicijalizacije GA se generiše populacija rešenja, pri čemu svako rešenje iz populacije predstavlja jednu jedinku koja je data svojim genetskim kodom. Kodiranje je najčešće binarno, ali može biti i celobrojno, realno ili kodirano nekim drugim nizom simbola. Inicijalno generisanje populacije se vrši nasumično, kako bi u populaciji jedinki postojao što raznovrsniji genetski kod. Svaka jedinka se evaluira funkcijom prilagođenosti, kojom se ocenjuje njen kvalitet. Prilagođenost je često u direktnoj vezi sa funkcijom cilja, ali može zavisiti i od drugih faktora npr. jedinstvenost genetskog materijala. U svakoj generaciji, izvršava se niz operacija nad populacijom: *selekcija*, *ukrštanje* i *mutacija*, respektivno.

Kriterijum zaustavljanja ovog algoritma može biti: broj dostignutih generacija, dostignut broj uzastopnih generacija bez poboljšanja rešenja, prekoračenje vremenskog limita, itd. Kao izlaz algoritma biće dato najbolje dobijeno rešenje.

Optimizacija mravljim kolonijama

Optimizacija mravljim kolonijama (engl. *Ant colony optimization - ACO*) je P-metaheuristika inspirisana kretanjem i ponašanjem mrava u potrazi za hranom. ACO metaheuristika je prvi put predložena u doktorskoj disertaciji M. Doriga [19].

Većina mrava nema izraženo čula vida. Mravi prilikom kretanja ispuštaju supstancu koja se zove feromon. Preostali mravi mogu da osete trag od feromona i da pomoću toga prate kretanja prethodnih mrava.

ACO algoritam počinje inicijalizacijom kolonije mrava, gde svaki mrav predstavlja jedno dopustivo rešenje. Takođe, u početnoj fazi algoritma potrebno je odrediti matricu feromona $L = [\lambda_{ij}]$ čiji su elementi unapred date vrednosti dobijene za svaku komponentu rešenja. Algoritam se sastoji iz dva osnovna koraka – konstrukcija rešenja i ažuriranje tragova feromona. U prvom koraku svake iteracije, svaki mrav iz populacije može se posmatrati kao pohlepno stohastička procedura koja konstruiše

kompletno rešenje (faza konstrukcije). Svaka sledeća komponenta parcijalnog rešenja bira se sa određenom verovatnoćom koja zavisi od trenutnih vrednosti u matrici feromona L. Na kraju svake iteracije vrši se ažuriranje matrice feromona u dve faze u zavisnosti od kvaliteta dobijenog parcijalnog rešenja i stope evaporacije. Prva faza je faza evaporacije koja ima cilj da eliminiše putanje koje se više ne koriste. Druga faza je faza pojačavanja koja ima cilj da pojača trag na putanjama koje su vodile do kvalitetnih rešenja.

Kriterijum zaistavljanja ACO algoritma može biti: dostignut određeni broj iteracija, dostignuto vreme izvršavanja, dostignut određeni kvalitet rešenja itd. Najbolje dobijeno rešenje ovom procedurom biće izlaz iz ACO algoritma.

Jedan od glavnih izazova ovog problema jeste dobar izbor parametara - veličina populacije mrava, stopa evoporacije feromonskih tragova kao i parametar intenzifikacije rešenja.

Optimizacija rojem čestica

Optimizacija rojem čestica (engl. *Particle Swarm Optimization - PSO*) je P-metaheuristika čija je definicija inspirisana ponašanjem inteligencije rojeva. Prvi put je predstavljena od strane Kenedija i Eberharta u radu [20].

U inicijalnoj fazi PSO algoritma generiše se n dopustivih rešenja. Svako rešenje svojim položajem definiše jednu česticu, dok populacija dopustivih rešenja predstavlja roj čestica. Svakoj od n čestica se zadaje početni položaj kao i vektor brzine koji predstavlja pravac kretanja čestice. Čestice će se kretati u svakoj iteraciji kroz prostor pretrage u cilju pronalaska najboljeg rešenja. Na početku, i-toj čestici iz roja se dodeljuje proizvoljan vektor položaja x_i^0 i proizvoljan vektor brzine v_i^0, pri čemu je $i = 1, 2, ..., n$. U svakoj iteraciji j se pamti najbolji položaj svake čestice i i označava se sa p_i^j. Takođe, pamti se i najbolja pozicija celog roja u iteraciji j u oznaci p_g^j. U narednoj iteraciji, čestice će promeniti svoj položaj i brzinu koristeći svoje prethodno iskustvo ali i u odnosu na iskustvo preostalih čestica u roju. Ažuriranje vektora brzine čestice roja i u iteraciji j se vrši po sledećoj formuli:

$$v_i^j = wv_i^{j-1} + c_1 r_1 (p_i^j - x_i^j) + c_2 r_2 (p_g^j - x_g^j).$$

Sa w se obeležava parametar inercije koji kontroliše uticaj prethodne brzine čestice na novu brzinu. Za velike vrednosti parametra w, čestica će imati veću brzinu i kretaće se do udaljenijih delova prostora pretrage, odnosno uticaće na pojačavanje diverzifikacije. Za male vrednosti parametra w, postiže se efekat sličan lokalnoj

pretrazi, odnosno intenzifikacija prostora pretrage. Parametar c_1 kontroliše uticaj iskustva čestice (faktor kognitivnog učenja), dok se parametrom c_2 kontroliše uticaj okolnih čestica (faktor socijalnog učenja). Parametri r_1 i r_2 se biraju nasumično iz intervala $(0, 1)$. Pozicija čestice i u iteraciji j se ažurira po sledećoj formuli:

$$x_i^j = x_i^{j-1} + v_i^{j-1}.$$

Ova procedura se izvršava sve do ispunjavanja nekog unapred definisanog kriterijuma zaustavljanja.

Jedna od glavnih prednosti PSO metaheuristike je jednostavnost implentacije kao i vremenska efikasnost, što je čini primenljivom metodom za široku klasu optimizacionih problema. Jedna od mana ove metode je veliki broj parametara, stoga, dobar odabir parametara može biti vremenski zahtevan.

Simulirano kaljenje

Simulirano kaljenje (engl. *Simulated Annealing, SA*) je S-metaheuristika koja koristi principe lokalnog pretraživanja i predstavlja kombinaciju stohastičkog i determinističkog pristupa. Prvi put je predložena od strane Kirkpatrika u radu [21]. Inspiracija za naziv ove metaheuristike dolazi iz procesa kaljenja čelika. Čelik se najpre zagreva do visoke temperature a zatim se sporo hladi, pri čemu se ovim procesom dostiže jaka i čvrsta struktura kristalne rešetke metala.

Ovaj algoritam kao ulazni podatak ima neko dopustivo rešenje x^*, kao i definisanu vrednost $T > 0$ koja predstavlja temperaturu. Početno rešenje x^* može biti slučajno generisano ili generisano nekom heuristikom. U svakom koraku iteracije, na slučajan način se pronalazi naredno rešenje $x^{'}$ iz okoline tekućeg rešenja x^*. Ukoliko je $x^{'}$ bolje od x^* (u smislu vrednosti funkcije cilja), $x^{'}$ postaje tekuće rešenje. Ukoliko je $x^{'}$ lošije, biće prihvaćeno kao tekuće rešenje ali sa određenom verovatnoćom koja zavisi od parametra T. U svakoj iteraciji vrednost T se smanjuje po nekom unapred definisanom pravilu - što je u analogiji sa smanjivanjem temperature u procesu kaljenja čelika. Posle određenog broja iteracija, vrednost T bi trebalo da postane bliska nuli. Da ne bi pri kraju izvršavanja algoritma bila prihvaćena loša rešenja, dobra ideja bi bila da se verovatnoća biranja lošijeg rešenja smanjuje kroz iteracije. Na primer, ukoliko se radi o problemu minimizacije, jedan od načina na koji to može da se uradi jeste da se verovatnoća prihvatanja lošijeg rešenja definiše sa:

$$p(T) = e^{-\delta/T},\ \delta = f(x^{'}) - f(x^*).$$

Za velike vrednosti promenljive T, verovatnoća $p(T)$ biće bliska jedinici, dok će za $T \to 0$ važiti $p(T) \to 0$.

Tabu pretraga

Tabu pretraga (engl. *Tabu search - TS*) je S-metaheuristika koja je prvi put predstavljena od strane Glovera u radu [22]. Ova metaheuristička metoda se bazira na lokalnoj pretrazi, a glavna ideja ove metode je uvođenje „tabu liste", koja predstavlja kratkoročnu memoriju u kojoj se nalazi skup „zabranjenih" rešenja/poteza. Tabu lista pamti rešenja/poteze koja su dovodila do zaglavljivanja u lokalnim optimumima ili rešenja koja su već posećena tokom procesa pretrage (posmatra se ceo proces pretrage ili samo izvestan broj prethodnih iteracija). Rešenja koja se nalaze u tabu listi ne mogu biti iskorišćena u narednim koracima iteracije.

U fazi inicijalizacije TS se zadaje neko dopustivo rešenje x^*, koje može biti nasumično generisano ili dobijeno kao rezultat neke heuristike. U unapred definisanoj okolini tekućeg najboljeg rešenja x^*, izvršava se lokalna pretraga i pronalazi se rešenje $x^{'}$. Ukoliko je $x^{'}$ bolje od x^*, tekuće rešenje postaje $x^{'}$. Takođe, bolje rešenje od x^* i $x^{'}$ se ubacuje u tabu listu, kako bi bilo isključeno iz pretrage u narednih nekoliko iteracija. Algoritam se zaustavlja nakon ispunjavanja nekog unapred definisanog kriterijuma zaustavljanja a za konačno rešenje biće proglašeno najbolje trenutno rešenje. Kriterijum zaustavljanja za TS algoritam može biti: dostignut maksimalan broj iteracija, dostignut broj iteracija bez popravke rešenja itd.

Pohlepna stohastičko-adaptivna procedura pretrage

Pohlepna stohastičko-adaptivna procedura pretrage (engl. *Greedy randomized adaptive searching procedure - GRASP*) je metaheuristička metoda sa višestartnim pristupom zasnovana na lokalnom pretraživanju. Ova metoda se najčešće koristi za rešavanje kombinatornih optimizacionih problema. Svaka iteracija GRASP algoritma sastoji se iz dve faze: pohlepno stohastičko-adaptivne faze konstrukcije rešenja (engl. *Greedy randomized solution construction - GRC*) i faze lokalne pretrage (engl. *Local search - LS*) koja služi za poboljšavanje rešenja. U svakoj iteraciji u fazi konstrukcije se kreira jedno dopustivo rešenje. Pohlepni algoritam mora imati stohastičku komponentu u cilju konstrukcije različitih rešenja u svakoj iteraciji. Rešenje konstrisano u GRC fazi je ulaz za LS fazu u kojoj pokušavamo da dobijemo poboljšanja. U svakoj iteraciji pamte se dostignuti lokalni optimumi, odnosno reše-

nje dobijeno kao izlaz iz LS faze, a najbolje pronađeno rešenje tokom svih GRASP iteracija predstavlja krajnji rezultat GRASP metode.

Jedna od prednosti korišćenja GRASP metaheuristike jeste mogućnost da proizvede visoko kvalitetna rešenja za relativno kratko vreme. Takođe, implementacija ove metaheuristike je često veoma jednostavna. Zbog toga, ova metoda je veoma popularan izbor u rešavanju velikog broja optimizacionih problema.

Metoda promenljivih okolina

Metoda promenljivih okolina (engl. *Variable Neighborhood Search - VNS*) je S-metaheuristika koja je prvi put predložena od strane Mladenovića i Hansena u radu [23]. Osnovna ideja ove metode zasniva se na sistematičnoj promeni unapred definisanih okolina radi dobijanja boljeg rešenja. Tri glavne činjenice na kojima se bazira ova metoda su:

1. Lokalni optimum u odnosu na jednu okolinu ne mora biti nužno lokalni optimum u odnosu na drugu okolinu.

2. Globalni optimum je lokalni optimum u odnosu na sve okoline.

3. Za većinu problema u praksi, lokalni optimumi u odnosu na različite okoline su relativno blizu jedan drugog.

VNS predstavlja jedan od najpopularnijih algoritama za rešavanje problema kombinatorne optimizacije. Neke od glavnih karakteristika su: jednostavnost, jasan sled izvršavanja algoritama kao i vremenska efikasnost. Zbog uopštene definicije ove metaheuristike, u literaturi je predložen veliki broj različitih varijanti ove metode.

Neke od varijanti VNS koje se sreću u literaturi su: **Osnovna metoda promenljivih okolina** (engl. *Basic Variable Neighborhood Search - BVNS*)**Metoda promenljivog spusta** (engl. *Variable Neighborhood Descent - VND*), **Redukovana metoda promenljivih okolina** (engl. *Reduced Variable Neighborhood Search - RVNS*), **Uopštena metoda promenljivih okolina** (engl. *General Variable Neighborhood Search - GVNS*), **Iskošena metoda promenljivih okolina** (engl. *Skewed Variable Neighborhood Search - SVNS*), **Metoda promenljivih okolina sa dekompozicijom** (engl. *Variable Neighborhood Decomposition Search - VNDS*), **Paralelna metoda promenljivih okolina** (engl. *Parallel Variable Neighborhood Search - PVNS*). Više o VNS metodi i njenim varijantama može se pronaći u radu [24].

2.3 Hibridizacija metaheuristike sa drugim (meta)heurističkim ili egzaktnim metodama

Hibridni algoritam predstavlja kombinaciju dva ili više algoritama ili tehnika koje se međusobno kombinuju za rešavanje nekog problema. Iz razloga što koriste različite pristupe, hibridni algoritmi često daju bolje rezultate nego pojedinačne komponente zasebno. Glavni izazov prilikom konstrukcije hibridnog algoritma jeste selekcija pojedinačnih algoritama, kao i način njihovog kombinovanja, koji zavisi od konkretnog problema. Kombinovanje komponenata iz metaheuristike i nekog drugog algoritma naziva se *hibridizacija metaheuristike*. Više o hibridizaciji metaheuristika sa drugim algoritmima se može pronaći u radovima [25] i [26].

Hibridizacija metaheuristike sa (meta)heuristikom.

Ukoliko želimo da poboljšamo performanse dobijene metaheuristike, prva ideja bi mogla biti kombinovanje metaheuristike sa nekom drugom metaheuristikom ili heuristikom. Postoji nekoliko različitih pristupa ovom problemu:

Prvi pristup hibridizacije može biti zamena određenih delova algoritama sa drugom (meta)heuristikom. Drugi pristup kombinovanja bi mogao da bude sledeći: rešenja dobijena metaheuristikom postaju inicijalna rešenja za drugi algoritam, ili obrnuto. Ovaj pristup često ne daje značajna poboljšanja metaheuristike. Treći pristup je da se kroz korake izvršavanja algoritma koriste naizmenično strategije iz obe komponente. U literaturi, najbolje performanse su pokazale hibridizacije P-metaheuristika i S-metaheuristika. Razlog tome su njihova komplementarna svojstva: dok su S-metaheuristike više orijentisane ka lokalnoj pretrazi i intenzifikaciji rešenja, P-metaheuristike su više orijentisane ka istraživanju prostora pretrage.

Hibridizacija metaheuristike sa metodom programiranja sa ograničenjima.

Programiranje sa ograničenjima (engl. *Constraint programming - CP*) predstavlja paradigmu za rešavanje kombinatornih optimizacionih problema. Problem programiranja sa ograničenjima se modeluje sa promenljivama, njihovim domenima i skupom ograničenja koji moraju da budu zadovoljeni. Postoji nekoliko načina na koje se može izvršiti hibridizacija metaheuristike sa CP, od kojih :

1. Metaheuristike se izvršavaju pre CP, pri čemu dobijena rešenja postaju inicijalna rešenja za drugi algoritam, i obrnuto.

2. Metoda lokalne pretrage kod metaheuristika koristi CP kako bi efikasno pretražila okolinu rešenja. Ovaj pristup je vrlo efikasan u slučaju kada prostor pretrage okoline sadrži veliki broj dopustivih rešenja.

3. Metaheuristike koje konstruišu svoja rešenja korak po korak. Benefit korišćenja CP je da se u svakom koraku smanji prostor pretrage parcijalnog rešenja. Jedan od primera metaheuristike koja se pokazala efikasna u ovakvom tipu hibridizacije je optimizacija mravljim kolonijama.

Hibridizacija metaheuristike sa dinamičkim programiranjem.

Dinamičko programiranje je metod u kom se rešavanje nekog problema svodi na rešavanje podproblema. Rešenja svakog podproblema se memorišu, tako da nije neophodno ponovno izračunavanje istih problema. U samoj metodi, mora biti određeno na koji način rešenje problema može biti konstruisano od rešenja njegovih podproblema.

Jedan od primera na koji se metaheuristika može efikasno ukombinovati sa dinamičkim programiranjem je iterativna dina-pretraga (engl. *iterated dynasearch*). Prilikom izvršavanja metaheuristike, primećeno je da se u proseku dobijaju bolje performanse ukoliko se u koraku lokalne pretrage poveća okolina u kojoj se vrši pretraga. U nekim slučajevima, okolina može biti velika i samim tim je vremenski zahtevno vršenje lokalne pretrage u toj okolini. U tom slučaju, metaheuristika može biti ubrzana tehnikama dinamičkog programiranja. Naime, vreme pretraživanja okoline u procesu lokalne pretrage može biti eksponencijalno, dok se tehnikama dinamičkog programiranja brzina lokalne pretrage može redukovati na polinomijalno vreme izvršavanja.

Hibridizacija metaheuristike sa tehnikom pretrage stabla.

Kod nekih algoritama, prostor pretrage formira strukturu stabla pretrage. Svaka putanja od korena stabla do lista, predstavlja jedno dopustivo rešenje optimizacionog problema. U svakom koraku, rešenje se konstruiše deo po deo (engl. *solution construction step*), pri čemu će metaheuristika navoditi proces pretrage. Ovakav vid hibridizacije je efikasan za metaheuristike u kojima se, počevši od praznog skupa,

parcijalno rešenje gradi deo po deo sve dok se ne dobije kompletno dopustivo rešenje. Neki od primera ovakvih metaheuristika su optimizacija mravljim kolonijama i GRASP.

Glava 3

Problem maksimizacije težinskih disjunktnih dominantnih skupova - problem MWDDS

Neka je zadat prost neusmeren graf $G = (V, E)$. Cilj problema maksimizacije disjunktnih dominantnih skupova (MDDS) je odrediti domatsku particiju grafa G za koju je broj dominantnih skupova maksimalan.

Pino je u radu [13], kao uopštenje MDDS problema, posmatrao težinsku varijantu problema MDDS i nazvao ga problemom maksimizacije težinskih disjunktnih dominantnih skupova (engl. *Maximum weighted disjoint dominant sets problem - MWDDS*). Dakle, problem MWDDS predstavlja uopštenje problema MDDS.

3.1 Problem MWDDS

U ovoj podsekciji biće opisan problem MWDDS predstavljen u radu [13]. Neka je zadat prost neusmeren graf $G = (V, E)$ i funkcija lifetime. Funkcija lifetime : $V \to \mathbb{R}^+$ predstavlja težinsku funkciju koja svakom čvoru u grafu $v \in V$ pridružuje pozitivan realan broj lifetime$(v) > 0$. Svaka domatska particija $\mathcal{D}$ od G predstavlja dopustivo rešenje ovog problema. Neka je $\mathcal{D} = \{D_1, ..., D_{|\mathcal{D}|}\}$ jedno dopustivo rešenje problema. Kvalitet iliti težina dominantnog podskupa D_i se definiše kao $\min\{\text{lifetime}(v) | v \in D_i\}$. Drugim rečima, kvalitet dominantnog podskupa D_i predstavlja vrednost najmanjeg životnog veka svih čvorova koji pripadaju D_i. Funkcija cilja f, svakoj domatskoj particiji dodeljuje pozitivan realan broj i definiše se kao $f(\mathcal{D}) := \sum_{i=1}^{|\mathcal{D}|} \min\{\text{lifetime}(v) | v \in D_i\}$. Cilj problema MWDDS je pronaći domatsku

particiju $\mathcal{D}^*$ za koju je vrednost funkcije $f(\mathcal{D}^*)$ maksimalna. Matematička formulacija problema MWDDS je sledeća:

$$\max f(\mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} \min\{\text{lifetime}(v) \mid v \in D_i\} \tag{3.1}$$

$$D_i \subseteq V \qquad i = 1, \ldots, |\mathcal{D}| \tag{3.2}$$

$$N[D_i] = V \qquad i = 1, \ldots, |\mathcal{D}| \tag{3.3}$$

$$D_i \cap D_j = \emptyset \qquad 1 \leq i < j \leq |\mathcal{D}| \tag{3.4}$$

Cilj problema MWDDS je maksimizacija funkcije cilja (3.1). Funkcija cilja $f(\mathcal{D})$ je definisana kao suma koja vrši sabiranje prolazeći kroz sve dominantne skupove u rešenju, pri čemu svaki sabirak sume predstavlja vrednost životnog veka od čvora sa najmanjom vrednošću životnog veka od svih čvorova koji se nalaze u tom dominantnom skupu. Uslov (3.2) obezbeđuje da su svi D_i podskupovi skupa čvorova V. Jednačina (3.3) obezbeđuje da su svi D_i dominantni skupovi, dok jednačina (3.4) osigurava disjunktnost tih skupova.

Lema 2 (Lema o ograničenju funkcije cilja MWDDS problema)**.** *Za svaku domatsku particiju $\mathcal{D}$ MWDDS problema važi: $f(\mathcal{D}) \leq \min_{v \in V} \sum_{v' \in N[v]} \text{lifetime}(v')$.*

Dokaz. Neka je $\mathcal{D} = \{D_1, ..., D_{|\mathcal{D}|}\}$ proizvoljna domatska particija grafa $G = (V, E)$ i neka je $v \in V$ proizvoljan čvor grafa G. Pošto su svi $D_i \in \mathcal{D}$ dominantni skupovi, tada važi $v \in N[D_i]$, odnosno $N[v] \cap D_i \neq \emptyset$. Iz toga se zaključuje da za svako $D_i \in \mathcal{D}$ postoji $v_i' \in N[v]$ tako da važi $v' \in D_i$. $f(\mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} \min\{\text{lifetime}(v) \mid v \in D_i\} \leq \sum_{i=1}^{|\mathcal{D}|} \text{lifetime}(v_i')$. S obzirom na disjunktnost navedenih dominantnih skupova D_i, može se tvrditi da su svi v_i' međusobno različiti čvorovi. Iz toga sledi: $\sum_{i=1}^{|\mathcal{D}|} \text{lifetime}(v_i') \leq \sum_{v' \in N[v]} \text{lifetime}(v')$, odnosno $f(\mathcal{D}) \leq \sum_{v' \in N[v]} \text{lifetime}(v')$. S obzirom na to da navedeno tvrđenje važi za proizvoljan čvor u grafu G, iz toga sledi da važi $f(\mathcal{D}) \leq \min_{v \in V} \sum_{v' \in N[v]} \text{lifetime}(v')$. □

3.2 Matematički model celobrojnog linearnog programiranja problema MWDDS

Matematički model celobrojnog linearnog programiranja problema MWDDS predložen u radu [15], koristi sledeće parametre i skupove promenljivih:

- x_{ij} - binarna promenljiva koja odgovara svakom paru (v_i, D_j), pri čemu su: $v_i (i = 1, \ldots, n)$ čvorovi a $D_j (j = 1, \ldots, \delta(G) + 1)$ disjunktni dominantni skupovi ili prazni skupovi, pri čemu je $\delta(G) := \min\{deg(v) \mid v \in V\}$. Prema lemi 1, maksimalan broj disjunktnih dominantnih skupova je $\delta(G)+1$. Promenljiva x_{ij} ima vrednost 1 ukoliko $v_i \in D_j$, inače ima vrednost 0.

- y_j - binarna promenljiva $(j = 1, \ldots, \delta(G) + 1)$ koja ima vrednost 1 ukoliko je D_j dominantan skup. Ukoliko je D_j prazan, tada y_j ima vrednost 0.

- M - parametar koji se definiše kao $M := \max\{\text{lifetime}(v) \mid v \in V\}$ i predstavlja maksimalni životni vek među svim čvorovima u grafu G.

- z_j - realna promenljiva $z_j \in [0, M]\, (M \in \mathbb{R}^+)$ čija vrednost predstavlja težinu j-tog dominantnog skupa.

Iz definicije parametra M, može se primetiti da važi $M \geq z_j$, za svako $(j = 1, \ldots, \delta(G) + 1)$.

Koristeći gornju notaciju i uvedene skupove promenljivih, matematički model celobrojnog linearnog programiranja problema MWDDS preuzet iz rada [15] glasi:

$$\max \sum_{j=1}^{\delta(G)+1} z_j \tag{3.1}$$

$$\sum_{j=1}^{\delta(G)+1} x_{ij} \leq 1 \qquad i = 1, \ldots, n \tag{3.2}$$

$$\sum_{v_k \in N[v_j]} x_{kj} \geq y_j \qquad j = 1, \ldots, \delta(G) + 1 \tag{3.3}$$

$$y_j \geq x_{ij} \qquad i = 1, \ldots, n \qquad j = 1, \ldots, \delta(G) + 1 \tag{3.4}$$

$$x_{ij} \cdot \text{lifetime}(v_i) + (1 - x_{ij}) \cdot M \geq z_j \qquad i = 1, \ldots, n \qquad j = 1, \ldots, \delta(G) + 1 \tag{3.5}$$

$$y_j \cdot M \geq z_j \qquad j = 1, \ldots, \delta(G) + 1 \tag{3.6}$$

$$y_j \geq y_{j+1} \qquad j = 1, \ldots, \delta \tag{3.7}$$

$$z_j \geq z_{j+1} \qquad j = 1, \ldots, \delta \tag{3.8}$$

- Cilj problema MWDDS je maksimizovati funkciju cilja koja je definisana kao suma težina svakog dominantnog skupa (3.1).

- Uslov (3.2) obezbeđuje da svaki čvor u grafu pripada najviše jednom dominantnom skupu. Ovaj uslov zapravo obezbeđuje disjunktnost skupova D_j.

- Uslov (3.3) obezbeđuje da svi neprazni skupovi D_j ispunjavaju uslov dominantnosti za graf G. Kako je desna strana ove nejednakosti binarna promenljiva y_j, ovaj uslov neće biti ispunjen jedino u slučaju da je leva strana nejednakosti jednaka 0, a desna strana jednaka 1. U tom slučaju čvor v_k ne pripada D_j, kao ni svi susedi čvora v_k. Iz toga bi sledilo da je $v_k \notin N[D_j]$, što se kosi sa pretpostavkom o dominantnosti skupa D_j.

- Uslov (3.4) obezbeđuje da proizvoljan čvor ne može pripadati praznom skupu. Pošto su obe strane nejednakosti binarne promenljive, ovaj uslov nije ispunjen samo u slučaju $y_j = 0$ i $x_{ij} = 1$. To bi značilo da čvor i pripada praznom skupu, što nije moguće.

- Uslov (3.5) obezbeđuje sledeće: ukoliko je $x_{ij} = 1$, tada čvor i pripada dominantnom skupu D_j a uslov (3.5) se svodi na lifetime$(v_i) \geq z_j$, što je ispunjeno jer važi $z_j = \min_{v \in D_j}$ lifetime(v).
  Ukoliko je $x_{ij} = 0$, tada se uslov svodi na $M \geq z_j$, što je uvek ispunjeno po definiciji parametra M.

- Uslov (3.6) obezbeđuje da za sve prazne skupove D_j važi $z_j = 0$.

- Uslovi (3.7) i (3.8) su neophodni za dobijanje jedinstvenog rešenja. Promenljive z_j, uz ova 2 uslova, biće sortirane opadajuće po težini dominantnih skupova. Prazni skupovi imaće težinu 0 i nalaziće se na kraju sortiranog niza.

3.3 Ilustrativni primer

Slika 3.1 ilustruje problem MWDDS na primeru grafa sa 9 čvorova i 10 ivica.

Na slici 3.1a, razmatrana instanca je grafički prikazana. Čvorovi grafa su numerisani brojevima od 1 do 9. U tabeli 3.1 prikazan je životni vek za svaki čvor grafa. Jednostavnosti radi, životni vek svakog čvora i iznosi $0.1i$.

Slika 3.1b predstavlja jedno dopustivo rešenje $\mathcal{D} := \{D_1 = \{1, 2, 7, 9\}, D_2 = \{4, 5, 6, 8\}\}$ pri čemu se rešenje $\mathcal{D}$ sastoji iz dva dominantna skupa $D_1 = \{1, 2, 7, 9\}$ i $D_2 = \{4, 5, 6, 8\}$. Životni vek čvorova 1, 2, 7 i 9 je 0.1, 0.2, 0.7 i 0.9, respektivno. Stoga, životni vek skupa D_1 je $\min\{0.1, 0.2, 0.7, 0.9\} = 0.1$. Slično, životni vek čvorova 4, 5, 6 i 8 je 0.4, 0.5, 0.6 i 0.8, respektivno. Stoga, životni vek skupa D_2 je

$\min\{0.4, 0.5, 0.6, 0.8\} = 0.4$. Kao posledica toga, vrednost funkcije cilja $f(\mathcal{D})$ biće suma brojeva 0.1 i 0.4 što iznosi 0.5.

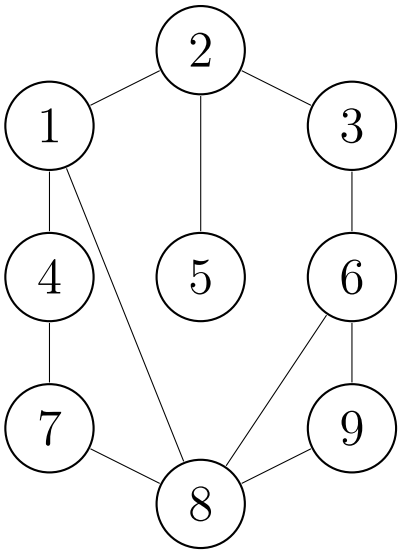
Slika 3.1c predstavlja optimalno rešenje za ovu instancu. U optimalnom rešenju $\mathcal{D}^* := \{D_1 = \{2, 4, 9\}, D_2 = \{5, 6, 7, 8\}\}$, životni vek skupova D_1 i D_2 iznosi 0.2 i 0.5, respektivno. Stoga, vrednost funkcije cilja za ovo rešenje je 0.7. S obzirom na to da graf sadrži čvor stepena 1, po lemi 1, svako dopustivo rešenje problema MWDDS za ovu instancu sadrži najviše dva disjunktna dominantna skupa.

Može se primetiti da za ovako definisan graf $G = (V, E)$ i težinsku funkciju lifetime važi da je $\min_{v \in V} \sum_{v' \in N[v]} \text{lifetime}(v') = 0.7$, pri čemu se minimum dostiže za čvor 5 jer važi $\sum_{v' \in N[5]} \text{lifetime}(v') = \sum_{v' \in \{2,5\}} \text{lifetime}(v') = 0.2{+}0.5 = 0.7$. Prema lemi 2, funkcija cilja za svako dopustivo rešenje ovog problema je manja ili jednaka od 0.7. S obzirom na to da rešenje sa slike 3.1c ima funkciju cilja jednaku 0.7, može se zaključiti da to predstavlja jedno optimalno rešenje problema.

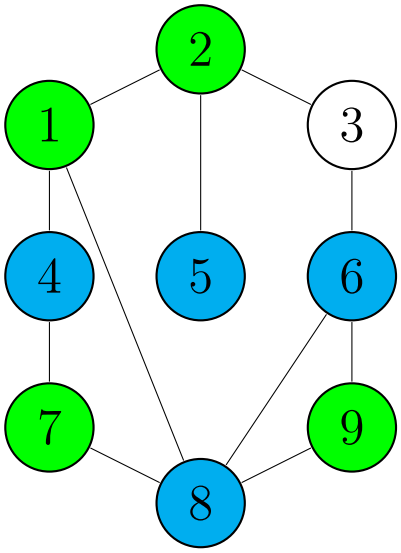
Ukoliko bi u rešenje sa slike 3.1c u skup D_1 bio dodatno ubačen čvor 3, dobijeno rešenje bi bilo $\mathcal{D} := \{D_1 = \{2, 3, 4, 9\}, D_2 = \{5, 6, 7, 8\}\}$. Za ovo rešenje takođe važi da je $f(\mathcal{D}) = 0.7$, što znači da MWDDS problem ne mora da ima jedinstveno optimalno rešenje.

Tabela 3.1: Težine odgovarajućih čvorova instance

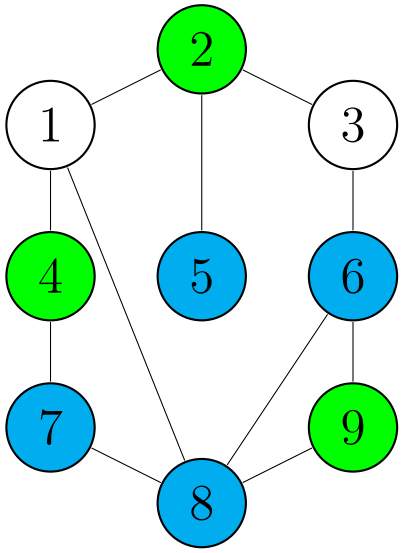
Čvor	1	2	3	4	5	6	7	8	9
Težina čvora	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9



(a) Primer instance problema. Na slici je prikazan graf sa 9 čvorova i 10 ivica.



(b) Jedno dopustivo rešenje problema $\mathcal{D} := \{D_1 = \{1, 2, 7, 9\}, D_2 = \{4, 5, 6, 8\}\}$, pri čemu je $f(\mathcal{D}) = 0.5$.



(c) Optimalno rešenje problema $\mathcal{D}^* := \{D_1 = \{2, 4, 9\}, D_2 = \{5, 6, 7, 8\}\}$, pri čemu je $f(\mathcal{D}^*) = 0.7$.

Slika 3.1: Ilustrativni primer problema MWDDS

Glava 4

Pohlepne heuristike za rešavanje problema MWDDS

4.1 GH-MWDDS

U ovoj sekciji je opisana pohlepna heuristika za rešavanje problema MWDDS, u oznaci GH-MWDDS, koja je prvi put predstavljena u radu [15]. Pseudokod pohlepne metaheuristike GH-MWDDS dat je Algoritmom 2, koji je preuzet iz rada [15]. Kao ulaz, zadat je prost, težinski, neusmeren graf $G = (V, E)$, kao i funkcija lifetime : $V \to \mathbb{R}^+$. Izlaz algoritma je rešenje $\mathcal{D}$ koje predstavlja skup disjunktnih dominantnih skupova. U inicijalnoj fazi algoritma, skup D je postavljen na prazan skup (*korak* 1). Dalje, skup V_{rem} koji je na početku inicijalizovan na V, predstavlja skup preostalih čvorova odnosno čvorova koji još nisu iskorišćeni za formiranje dominantnih skupova (*korak* 3). V_{done} predstavlja skup svih čvorova koji su već iskorišćeni za kreiranje dominantnih skupova. Brojač k označava indeks dominantnog skupa koji bi trebalo da bude kreiran u toj iteraciji i na početku je postavljen na vrednost 0 (*korak* 2).

U svakoj iteraciji $k \geq 1$ algoritam generiše jedan dominantni skup D_k koristeći čvorove koji se trenutno nalaze u V_{rem} (*korak* 5). Konstrukcija dominantnog skupa D_k u k-toj iteraciji se ostvaruje na sledeći način. Na početku, D_k se inicijalizuje na prazan skup. U svakom trenutku, svaki čvor ulaznog grafa G pripada tačno jednom od sledećih skupova:

- Crni čvorovi: čvorovi koji trenutno pripadaju D_k.
- Sivi čvorovi: čvorovi u $\mathcal{G}(D_k) := N(D_k) \setminus D_k$, odnosno, čvorovi koji nisu u D_k ali za koje postoji crni čvor koji im je sused.

- Beli čvorovi: čvorovi u $\mathcal{W}(D_k) := V \setminus N[D_k]$, koji nisu ni crni ni sivi.

Lema 3. *Za ovako definisane skupove $D_k, \mathcal{G}(D_k)$ i $\mathcal{W}(D_k)$ važe sledeća tvrđenja:*

1. *$D_k, \mathcal{G}(D_k)$ i $\mathcal{W}(D_k)$ su međusobno disjunktni skupovi.*
2. *$D_k \cup \mathcal{G}(D_k) \cup \mathcal{W}(D_k) = V$.*

Dokaz. Dokaz leme 3 sledi iz činjenice da je presek svaka dva od sledeća tri skupa: $D_k, \mathcal{G}(D_k)$ i $\mathcal{W}(D_k)$ - prazan skup. Takođe, u dokazu će biti iskorišćen identitet $N(D_k) \subseteq N[D_k]$.

1. a) $D_k \cap \mathcal{G}(D_k) = D_k \cap (N(D_k) \setminus D_k) = D_k \cap N(D_k) \cap D_k^c = N(D_k) \cap \emptyset = \emptyset$.
   b) $\mathcal{G}(D_k) \cap \mathcal{W}(D_k) = (N(D_k) \setminus D_k) \cap (V \setminus N[D_k]) = N(D_k) \cap D_k^c \cap V \cap N[D_k]^c \subseteq D_k^c \cap N(D_k) \cap N(D_k)^c = D_k^c \cap \emptyset = \emptyset \implies \mathcal{G}(D_k) \cap \mathcal{W}(D_k) = \emptyset$.
   c) $\mathcal{W}(D_k) \cap D_k = V \cap N[D_k]^c \cap D_k \subseteq D_k^c \cap D_k = \emptyset \implies \mathcal{W}(D_k) \cap D_k = \emptyset$.
2. $D_k \cup \mathcal{G}(D_k) \cup \mathcal{W}(D_k) = D_k \cup (N(D_k) \setminus D_k) \cup (V \setminus N[D_k]) = N[D_k] \cup (V \setminus N[D_k]) = V$.

□

Iz leme 3 može se zaključiti sledeće: u svakom koraku konstruisanja skupa D_k važi da je svaki čvor obojen u jednu od tri boje: belu, sivu ili crnu. Dalje, uočava se da je D_k iterativno generisan tako što se, polazeći od praznog skupa, u svakom koraku konstrukcije dodaje tačno jedan čvor iz skupa $\{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}$. To jest, u svakom koraku se u skup D_k dodaje po jedan čvor koji je bele ili sive boje i nije prethodno korišćen za konstrukciju dominantnih skupova u nekoj od prethodnih iteracija (*korak* 12). Svi čvorovi iz skupa $\{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}$ se evaluiraju pohlepnom funkcijom $score()$, koja je definisana na sledeći način:

$$score(v) := \text{lifetime}(v) \cdot white\_degree(v) \qquad \forall v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem} \quad (4.1)$$

$$white\_degree(v) := |N[v] \cap \mathcal{W}(D_k)| \quad (4.2)$$

Drugim rečima, pohlepna funkcija *score* čvora v se dobija množenjem životnog veka čvora sa brojem belih čvorova iz zatvorene okoline čvora v. U svakoj iteraciji biće izabran čvor v^* kao onaj za koji važi $v^* = \text{argmax}\{score(v) \mid v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}\}$ (*korak* 12). Operator argmax vraća argument za koji

zadata funkcija dostiže maksimum na zadatom skupu. Odnosno, čvor sa najvećom vrednošću funkcije *score*() biće sledeći iskorišćen u konstrukciji dominantnog skupa. Ukoliko u nekom trenutku konstrukcije više čvorova imaju najveću vrednost funkcije *score*(), tada će među čvorovima koji imaju najveću vrednost funkcije *score*() biti izabran čvor sa najmanjim indeksom.

Algoritam se zaustavlja kada se od neiskorišćenih čvorova (V_{rem}) ne može formirati više nijedan dominantni skup. Uslov zaustavljanja ovog algoritma može da se predstavi kao jedan od sledeća dva ekvivalentna uslova:

1. Ukoliko za barem jedan čvor $v \in V$ važi sledeće: čvor v kao i svi susedi čvora v se nalaze u V_{done}. Preciznije, ukoliko postoji bar jedan $v \in V$ tako da je $N[v] \subseteq V_{done}$ *(korak 5)*.

2. V_{rem} nije dominantan.

Ukoliko važi uslov 1, tada postoji čvor $v \in V$ tako da $N[v] \subseteq V_{done}$. Pretpostavimo suprotno, da je moguće formirati još jedan dominantni skup D_k. Zbog toga što važi $N[v] \subseteq V_{done}$, tada za svako $u \in N[v]$ važi da $u \notin D_k$, stoga $v \notin N[D_k] = V$, što je kontradikcija. Ukoliko važi uslov 2, tada svaki podskup $D_k \subseteq V_{rem}$ nije dominantni skup jer V_{rem} nije dominantan skup.

Algoritam 2 GH-MWDDS

Input: Prost težinski neusmeren graf G.
Output: Familija disjunktnih dominantnih skupova $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$.

1: $\mathcal{D} \leftarrow \emptyset$
2: $k \leftarrow 0$
3: $V_{rem} \leftarrow V$
4: $V_{done} \leftarrow \emptyset$
5: **while** ne postoji $v \in V$ tako da je $N[v]$ podskup od V_{done} **do**
6: $k \leftarrow k + 1$
7: **for** $v \in V$ **do**
8: $color(v) \leftarrow$ WHITE
9: **end for**
10: $D_k \leftarrow \emptyset$
11: **while** D_k nije dominantan skup od G **do**
12: $v^* \leftarrow \text{argmax}\{score(v) \mid v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}\}$
13: $D_k \leftarrow D_k \cup \{v^*\}$
14: $V_{rem} \leftarrow V_{rem} \setminus \{v^*\}$
15: **for** $u \in N(v^*)$ **do**
16: **if** $color(u) =$ WHITE **then**
17: $color(v) \leftarrow$ GRAY
18: **end if**
19: **end for**
20: $color(v^*) \leftarrow$ BLACK
21: **end while**
22: #Reduce(D_k, V_{rem}, V_{done}) ▷ Naredba uklanjanja redundantnih čvorova
23: $\mathcal{D} \leftarrow \mathcal{D} \cup \{D_k\}$
24: $V_{done} \leftarrow V_{done} \cup D_k$
25: **end while**
26: **return** $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$

Jedan od načina za dalje poboljšanje rešenja jeste uklanjanje redundantnih čvorova pre dodavanja D_k u rešenje $\mathcal{D}$. Formalno, čvor $v \in D_k$ je redundantan ukoliko je svaki čvor iz njegove zatvorene okoline sused sa barem dva čvora iz D_k. Odnosno, $N[v] \subset \cup_{u \in D_k \setminus \{v\}} N[u]$. Uključivanjem *koraka 22* iz Algoritma 2, dobija se algoritam GH-MWDDS+ koji iterativno uklanja redundantne čvorove iz rešenja. U tu svrhu, nakon što se konstruiše D_k, određuje se funkcija $F_{D_k} : V \to \mathbb{N}$ definisana kao: $F_{D_k}(v) = |N[v] \cap D_k|$, za svako $v \in V$. Drugim rečima, funkcijom F_{D_k} svakom čvoru $v \in V$ se dodeljuje broj čvorova iz D_k koji pripadaju zatvorenoj okolini čvora v. Čvor $v \in D_k$ je redundantan čvor dominantnog skupa D_k ukoliko važi:

$$(\forall v' \in N[v]) \quad F_{D_k}(v') \geq 2. \tag{4.3}$$

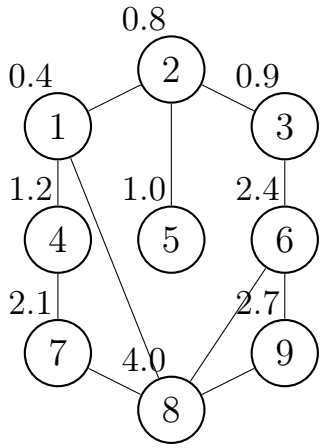
Ukoliko za čvor $v \in D_k$ važi uslov (4.3), tada je čvor v redundantan i on biva izbačen iz rešenja. Drugim rečima, za takav čvor v važiće da se svi čvorovi iz njegove zatvorene okoline nalaze u zatvorenoj okolini barem dva čvora iz D_k. Stoga, izbacivanjem čvora $v \in D_k$ neće se narušiti dominantnost skupa D_k. U procesu izbacivanja, u *naredbi* 22, skupovi D_k i V_{rem} se ažuriraju na sledeći način: $D_k \leftarrow D_k \setminus \{v\}$, $V_{rem} \leftarrow V_{rem} \cup \{v\}$. Proces se ponavlja dokle god postoji $v \in D_k$ za koji važi uslov (4.3).

Neki od benefita izbacivanja redundantnih čvorova su sledeći:

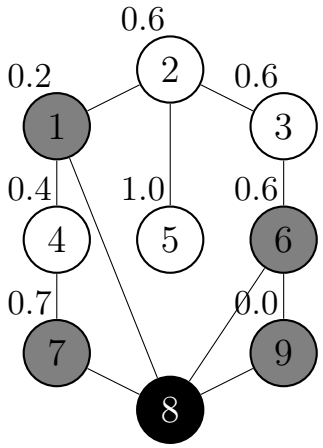
- Izbačeni čvorovi se u kasnijim iteracijama mogu iskoristiti za konstruisanje dominantnih skupova.
- Izbačeni čvor $v \in D_k$ može biti onaj za koji važi $v = \text{argmin}\{\text{lifetime}(v^{'}) \mid v^{'} \in D_k\}$. U tom slučaju, životni vek dominantnog skupa D_k se povećava, što utiče na povećanje funkcije cilja.

Ilustrativni primer pohlepnog algoritma GH-MWDDS

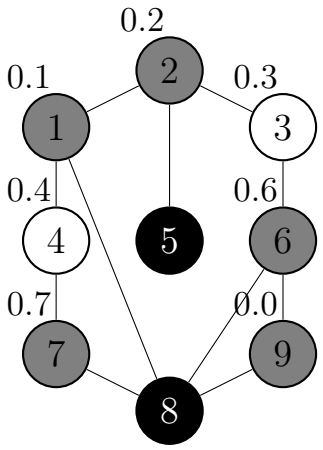
U ovoj podsekciji biće predstavljen ilustrativni primer pohlepnog GH-MWDDS algoritma. Za ulazni graf G biće korišćen graf iz sekcije 3.3, koji je prikazan na slici 4.1a. Ukoliko se čvor v nalazi u skupu $\{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}$, za njega će biti izračunata vrednost $score(v)$ iz izraza 4.1 i 4.2. Na slikama 4.1 i 4.2, vrednost $score(v)$ je prikazana pored odgovarajućeg čvora v. Na slici 4.1 prikazani su koraci konstruisanja prvog dominantnog skupa D_1 dok su na slici 4.2 prikazani koraci konstruisanja drugog dominantnog skupa D_2. Nakon konstruisanja skupa D_2 (slika 4.2d) može se primetiti da skup $V_{rem} = \{1, 3\}$ nije dominantan čime se prekida izvršavanje algoritma GH-MWDDS.



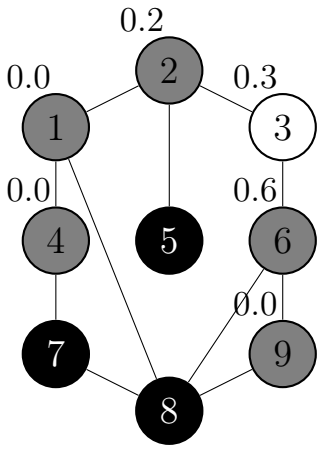
(a) $V_{rem} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Izabran je čvor 8.



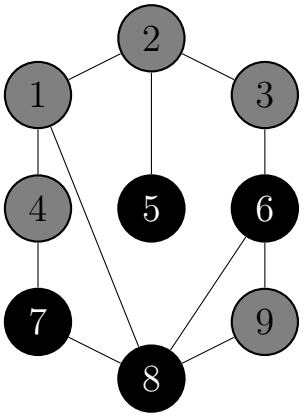
(b) $V_{rem} = \{1, 2, 3, 4, 5, 6, 7, 9\}$. Izabran je čvor 5.



(c) $V_{rem} = \{1, 2, 3, 4, 6, 7, 9\}$. Izabran je čvor 7.

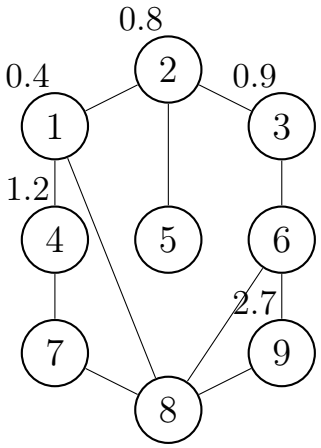


(d) $V_{rem} = \{1, 2, 3, 4, 6, 9\}$. Izabran je čvor 6.

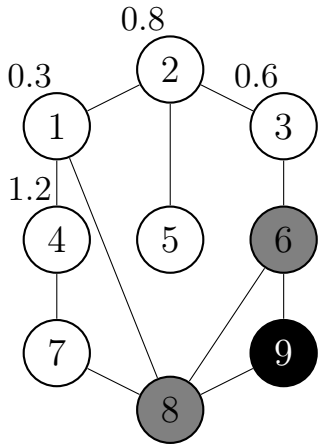


(e) $V_{rem} = \{1, 2, 3, 4, 9\}$. Konstruisan je skup $D_1 = \{5, 6, 7, 8\}$.

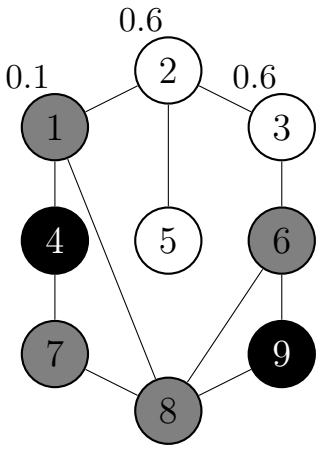
Slika 4.1: Koraci GH-MWDDS algoritma za konstruisanje dominantnog skupa $D_1 = \{5, 6, 7, 8\}$



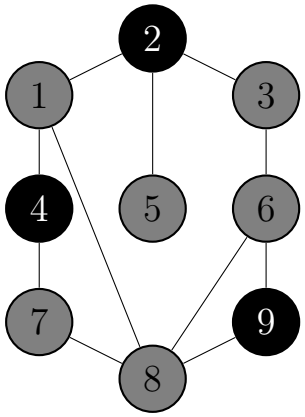
(a) $V_{rem} = \{1, 2, 3, 4, 9\}$. Izabran je čvor 9.



(b) $V_{rem} = \{1, 2, 3, 4\}$. Izabran je čvor 4.



(c) $V_{rem} = \{1, 2, 3\}$. Izabran je čvor 2.



(d) $V_{rem} = \{1, 3\}$. Konstruisan je dominantni skup $D_2 = \{2, 4, 9\}$.

Slika 4.2: Koraci GH-MWDDS algoritma za konstruisanje dominantnog skupa $D_2 = \{2, 4, 9\}$.

4.2 Mod-GH-MWDDS

Na osnovu Algoritma 2 koji prikazuje strukturu GH-MWDDS, može se primetiti da se u svakoj iteraciji čvorovi evaluiraju pohlepnom funkcijom $score()$. Nakon evaluacije, bira se čvor v^* kao čvor koji ima najveću vrednost $score(v^*)$. Različit izbor pohlepne funkcije dovodi do različitih rešenja. U nastavku podsekcije biće predstavljen algoritam Mod-GH-MWDDS koji ima drugačije izabranu pohlepnu funkciju $score(v)$ u odnosu na GH-MWDDS algoritam, predložen u radu [15].

Podsetimo se da je u GH-MWDDS algoritmu, pohlepna funkcija *score*() definisana izrazom 4.1. Cilj ovakvog načina uvođenja pohlepne funkcije *score*() u radu [15] jeste da izabran čvor v ima podjednako velike vrednosti lifetime(v) i $white\_degree(v)$. U ovom master radu je predložena modifikacija algoritma GH-MWDDS koja polazi od dve jednostavnije definicije pohlepne funkcije $score(v)$.

Prva definicija je data sa:

$$score(v) := \text{lifetime}(v) \qquad \forall v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem} \tag{4.4}$$

Definicija 4.4 dovodi do toga da će uvek biti izabran čvor sa najvećim mogućim životnim vekom. Ovo dalje vodi ka tome da se veliki broj čvorova iskoristi za pravljenje jednog dominantnog skupa, jer se nigde ne uzima u obzir pokrivenost grafa tim čvorovima. Životni vek takvih dominantnih skupova biće veliki, ali biće formiran samo mali broj dominantnih skupova.

Druga definicija pohlepne funkcije $score(v)$ data je sa:

$$score(v) := white\_degree(v) \qquad \forall v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem} \tag{4.5}$$

Korišćenjem definicije (4.5), uvek će biti izabran čvor v sa najvećom mogućom vrednošću $white\_degree(v)$. Ovakav izbor pohlepne funkcije dovodi do toga da se veoma brzo konstruišu dominantni skupovi jer se uvek biraju čvorovi koji su susedi sa najvećim brojem preostalih čvorova, pa se uslov dominantnosti skupa brzo postiže. Takođe, u ovom slučaju, biće konstruisan veliki broj dominantnih skupova. Glavna mana ovog načina odabira pohlepne funkcije *score*() je ta što se u njenoj definiciji nigde ne uzima u obzir životni vek čvorova. Dominantni skupovi konstruisani na ovaj način imaju malu vrednost životnog veka, stoga, ovakav način konstruisanja dominantnih skupova nije pogodan.

Iz izložene diskusije, može se zaključiti da pohlepna funkcija koja je iskorišćena za GH-MWDDS predstavlja balans između dve strategije:

- Brzo konstruisanje dominantnih skupova, pri čemu je veličina tih skupova najmanja moguća.
- Konstrukcija dominantnih skupova koristeći čvorove velikog životnog veka.

U cilju otklanjanja navedenih nedostataka funkcije *score*() iz rada [15], u ovom master radu je predložena nova definicija funkcije *score*() koja je implementirana u modifikovani pohlepni algoritam pod nazivom Mod-GH-MWDDS. Implementacija

Mod-GH-MWDDS algoritma je slična kao GH-MWDDS. Ključni elementi u kojima se Mod-GH-MWDDS razlikuje od GH-MWDDS su:

1. Ulaznim podacima algoritma, dodaju se tri nova parametara $p \in [0,1]$, $\alpha \in \mathbb{R}^+$ i $rep \in \mathbb{N}$.

2. Pohlepna funkcija $score(v)$ je definisana na sledeći način:

$$score(v) := \text{lifetime}(v)^{\alpha} \cdot white\_degree(v) \qquad \forall v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem} \tag{4.6}$$

   Korišćenjem nove funkcije $score()$ obezbeđuje se bolji balans između funkcija $white\_degree(v)$ i lifetime(v). Smanjivanjem parametra α, povećava se vrednost lifetime$^{\alpha}(v)$, jer je lifetime$(v) \in (0,1)$, za svako $v \in V$. Stoga, za male vrednosti α, pohlepna funkcija $score()$ iz definicije (4.6) daje slična rešenja kao pohlepna funkcija $score()$ iz definicije (4.4). Slično, povećavanjem parametra α smanjuje se vrednost lifetime$^{\alpha}(v)$. Stoga, za velike vrednosti α, pohlepna funkcija $score()$ iz definicije (4.6) daje slična rešenja kao pohlepna funkcija $score()$ iz definicije (4.5).

3. U svakoj iteraciji se pronalaze v^* i v^{**} kao čvorovi za koje funkcija $score(v)$ dostiže maksimum na datim skupovima:

$$v^* = \text{argmax}\{score(v) \mid v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}\}.$$
$$v^{**} = \text{argmax}\{score(v) \mid v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem} \setminus \{v^*\}\}.$$

   Sa verovatnoćom p se bira čvor v^*, dok se sa verovatnoćom $1-p$ bira čvor v^{**}. Ukoliko je skup $\{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}$ jednočlan, tada se bira čvor v^* (jer je u tom slučaju on jedini izbor). Ukoliko u nekom trenutku konstrukcije više čvorova imaju najveću vrednost funkcije $score()$, tada će među čvorovima koji imaju najveću vrednost funkcije $score()$ biti izabran čvor sa najmanjim indeksom.

4. Zbog elementa 3, algoritam nije deterministički. Algoritam se pokreće rep puta i među tim rešenjima se bira najbolje.

Pseudo-kod pohlepne metode Mod-GH-MWDDS prikazan je Algoritmom 3:

Algoritam 3 Mod-GH-MWDDS

Input: Prost težinski neusmeren graf G, p, α, rep
Output: Familija disjunktnih dominantnih skupova $\mathcal{D} = \{D_1, D_2, \ldots, D_k\}$.

1: $\mathcal{D} \leftarrow \emptyset$
2: $\mathcal{D}' \leftarrow \emptyset$
3: $k \leftarrow 0$
4: $V_{rem} \leftarrow V$
5: $V_{done} \leftarrow \emptyset$
6: **for** $i \in \{1, \ldots, rep\}$ **do**
7: **while** ne postoji $v \in V$ tako da je $N[v]$ podskup od V_{done} **do**
8: $k \leftarrow k + 1$
9: **for** $v \in V$ **do**
10: $color(v) \leftarrow$ WHITE
11: **end for**
12: $D_k \leftarrow \emptyset$
13: **while** D_k nije dominantan skup od G **do**
14: $v^* \leftarrow \text{argmax}\{score(v) \mid v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}\}$
15: $r \leftarrow$ slučajan realan broj iz intervala $(0, 1)$
16: **if** $r > p$ i $\{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem}$ nije jednočlan. **then**
17: $v^* \leftarrow \text{argmax}\{score(v) \mid v \in \{\mathcal{G}(D_k) \cup \mathcal{W}(D_k)\} \cap V_{rem} \setminus \{v^*\}\}$
18: **end if**
19: $D_k \leftarrow D_k \cup \{v^*\}$
20: $V_{rem} \leftarrow V_{rem} \setminus \{v^*\}$
21: **for** $u \in N(v^*)$ **do**
22: **if** $color(u) =$ WHITE **then**
23: $color(v) \leftarrow$ GRAY
24: **end if**
25: **end for**
26: $color(v^*) \leftarrow$ BLACK
27: **end while**
28: #Reduce(D_k, V_{rem}, V_{done}) ▷ Naredba uklanjanja redundantnih čvorova
29: $\mathcal{D} \leftarrow \mathcal{D} \cup \{D_k\}$
30: $V_{done} \leftarrow V_{done} \cup D_k$
31: **end while**
32: $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{\mathcal{D}\}$
33: **end for**
34: **return** $\text{argmax}\{f(\mathcal{D}) \mid \mathcal{D} \in \mathcal{D}'\}$

Prva *while* petlja (*korak 7*) služi za konstruisanje dominantnih skupova. U svakom prolazu kroz ovu petlju, biće konstruisan tačno jedan dominantni skup. Petlja prestaje sa izvršavanjem ukoliko se od preostalih čvorova ne može formirati domi-

nantni skup. Druga *while* petlja (*korak 13*) služi za konstruisanje jednog dominantnog skupa. U svakoj iteraciji ove *while* petlje, tačno jedan čvor se dodaje u skup. Petlja prestaje sa izvršavanjem u trenutku kada skup u koji smo dodavali čvorove postane dominantan. Jedna od ključnih razlika algoritma Mod-GH-MWDDS u odnosu na algoritam GH-MWDDS jeste u *koracima* 16 i 17. Dok u GH-MWDDS algoritmu, u svakoj iteraciji se dodaje čvor sa najvećom vrednošću funkcije $score()$, u Mod-GH-MWDDS algoritmu sa verovatnoćom p biramo čvor sa najvećom vrednošću funkcije $score()$, dok sa verovatnoćom $1-p$ biramo čvor sa drugom najvećom vrednošću funkcije $score()$. Na ovaj način biće konstruisano rep rešenja (*korak 6*) među kojima se na kraju izvršavanja algoritma bira rešenje sa najvećom vrednošću funkcije cilja (*korak 34*).

Uključivanjem *koraka 28* algoritma 3, dobija se algoritam Mod-GH-MWDDS+ koji iterativno uklanja redundantne čvorove iz rešenja. U toku izvršavanja algoritma, nakon što se konstruiše dominantni skup D_k, vrši se provera da li u tom dominantnom skupu postoje redundantni čvorovi. Važno je napomenuti redosled pretrage redundantih čvorova. U ovom master radu, čvorovi su numerisani brojevima 1,2,...,$|V|$ i tim redom se izvršavalo ispitivanje redundantnosti svakog od čvorova. Drugačiji redosled izbacivanja čvorova iz dominantnog skupa mogao bi da dovede do različitih rešenja.

Osvrnimo se i na nedostatke korišćenja problema MWDDS za problem produžavanja životnog veka bežičnih senzorskih mreža. Naime, problem MWDDS zahteva disjunktnost dominantnih skupova (uslov (3.4)). S tim u vezi, proizvoljan senzorski čvor može pripadati najviše jednom dominantnom skupu D_k, odnosno može biti korišćen samo u jednom ciklusu paljenja i gašenja. Drugim rečima, jednom ugašen aktivni senzor više nikad ne postaje aktivan i samim tim ne može više nikad da učestvuje u radu WSN. Nakon jednog ciklusa, u jednom dominantnom skupu D_k, čvor sa najmanjom dužinom životnog veka biva ugašen usled potrošnje njegove baterije (jer je vreme trajanja WSN koristeći aktivne senzore iz D_k jednako $\min_{v \in D_k} \text{lifetime}(v)$). Međutim, preostalim senzorskim čvorovima iz skupa D_k životni vek biva umanjen za životni vek senzorskog čvora sa najmanjim životnim vekom. Pošto će preostali čvorovi iz D_k imati nenula životni vek (osim u slučaju kada svi čvorovi iz D_k imaju isti životni vek), oni mogu biti iskorišćeni i u narednim ciklusima rada WSN.

Formulacija problema MWDDS može biti modifikovana tako da se uslov disjunktnosti dominantnih skupova zameni uslovom da čvor može pripadati u više dominantnih skupova dokle god ima nenula životni vek. Nakon konstruisanja do-

minantnog skupa D_k, životni vek čvorova $v \in D_k$ biće ažuriran kao: lifetime$(v) \leftarrow$ lifetime$(v) - \min\{\text{lifetime}(v) | v \in D_k\}$. Svako rešenje problema MWDDS bi ujedno bilo i rešenje novog problema, odnosno, novi problem bi bio proširenje problema MWDDS.

Pohlepna heuristika Mod-GH-MWDDS može biti primenjena uz neznatne modifikacije i na novi problem. Naime, nakon kreiranja dominantnog skupa D_k, životni vek svih čvorova u D_k bio bi umanjen za vrednost $\min\{\text{lifetime}(v) | v \in D_k\}$. Tokom konstruisanja dominantnog skupa D_k iz skupa V_{rem} se izbacuje jedan po jedan element koji se koristi u konstrukciji D_k. Nakon konstruisanja dominantnog skupa D_k, u skup V_{rem} bi bili vraćeni svi čvorovi iz D_k koji imaju životni vek veći od 0.

Glava 5

Eksperimentalni rezultati

Pohlepni algoritmi iz rada [15], GH-MWDDS i GH-MWDDS+ su implementirani u ANSI C++ koristeći GCC 7.4.0 za kompajliranje koda. Performanse ovih algoritama su upoređene sa tri algoritma lokalne pretrage koji su predloženi u radu [13]: Algoritam lokalne pretrage (LS), Algoritam fiksirane dubine (FD) i Algoritam promenljive dubine (VD). U cilju fer upoređivanja pohlepnih algoritama iz rada [15] sa algoritmima LS, FD i VD iz rada [13], autori rada [15] su iskoristili kod algoritama lokalne pretrage iz rada [13] i na svojoj mašini pokrenuli algoritme. Algoritmi LS, FD, VD [13] kao i GH-MWDDS, GH-MWDDS+ [15] su pokretani na mašini sa procesorom 64-bit 2.5-GHz Intel Core^{TM} i5-7200U i 8GB RAM memorije.

Za ispitivanje performansi modela celobrojnog linearnog programiranja u radu [15], korišćen je egzaktni rešavač IBM ILOG CPLEX Academic Edition 20.1 u radu, čije je vreme izvršavanja ograničeno na 2 sata. CPLEX rešavač je pokretan na grupi mašina sa Intel®Xeon® Silver 4210 CPUs sa 10 logičkih jezgara, 2.2GHz i 92GB RAM memorije.

Heuristike Mod-GH-MWDDS i Mod-GH-MWDDS+ predložene u ovom master radu su implementirane u programskom jeziku C++ i okruženju Visual Studio 2020. Rezultati testiranja algoritama Mod-GH-MWDDS i Mod-GH-MWDDS+ prikazani u tabelama su dobijeni na mašini sa procesorom AMD Ryzen 7 2700X Eight-Core Processor i 32 GB RAM memorije.

5.1 Instance

U cilju evaluacije predloženih heuristika korišćeno je ukupno 640 instanci. Instance koje su korišćene u ovom radu su identične instancama generisanim u radu

[15]. Svaka instanca je generisana pomoću dva glavna parametra: broj čvorova mreže n i prosečan stepen čvora mreže d, pri čemu je $d = 2m/n$ gde je m broj ivica mreže. Za broj čvorova mreže n korišćene su vrednosti 50, 100, 150, 200 i 250. Za svako n, posmatrano je 5 do 8 različitih prosečnih stepena čvora mreže d, tako da su pokriveni slučajevi od retkih do gustih mreža. Dodatno, svakom čvoru mreže dodeljen je slučajan realan broj iz intervala (0,1) kao njegova težina (lifetime). Dalje, za svaku kombinaciju (n, d) generisano je po 20 različitih mreža na slučajan način što rezultira ukupnom broju od 640 instanci za eksperimentalnu evaluaciju i poređenja.

5.2 Analiza i izbor parametara za Mod-GH-MWDDS

Algoritam Mod-GH-MWDDS kao ulazne parametre, osim grafa G, ima još tri parametra: p, α i rep. Za parametar rep je izabrana vrednost 10. Cilj je odrediti kombinaciju (p, α) koja vodi najboljim performansama, odnosno daje najbolja rešenja za ovaj problem. Za parametar p, izabrane vrednosti su 0.4, 0.5, 0.6, 0.8 i 1.0, dok su za parametar α izabrane vrednosti 1.0, 1.5, 2.0, 2.5 i 3.0. Za svaku kombinaciju (n, d) postoji 20 instanci dok su za parametarsku analizu na slučajan način odabrane po tri instance, što rezultira ukupnom broju od 96 instanci za parametarsku analizu. Tih 96 instanci su najpre podeljene u 5 grupa u zavisnosti od n $(n = 50, 100, 150, 200, 250)$, a zatim je za svaku grupu izračunata prosečna vrednost funkcije cilja. Algoritam Mod-GH-MWDDS je ovom precedurom pokrenut za svaku kombinaciju (p, α).

U tabelama 5.1, 5.2, 5.3, 5.4 i 5.5 su prikazani rezultati parametarskog testiranja redom za $n = 50$, 100, 150, 200, 250. Prosečna vrednost funkcije cilja je podebljana u tabeli ukoliko ima najveću vrednost za posmatrane parametre. Nakon dobijenih rezultata za $n = 50, 100, 150$, parametri sa vrednošću $p = 0.4, 0.5$ i $\alpha = 2.5, 3.0$ su izbačeni iz testiranja u nastavku jer su davali najmanju prosečnu vrednost funkcije cilja.

Tabela 5.1: Rezultati parametarske analize za $n = 50$

$p\backslash\alpha$	**1.0**	1.5	2.0	2.5	3.0
0.4	4.4516	4.4561	4.3836	4.3959	4.3967
0.5	4.5003	4.4675	4.4102	4.4645	4.3435
0.6	4.5909	4.5443	4.5071	4.4271	4.3799
0.8	**4.6509**	4.6335	4.6021	4.5542	4.4913
1.0	4.4396	4.4698	4.4486	4.3391	4.3663

Tabela 5.2: Rezultati parametarske analize za $n = 100$

$p\backslash\alpha$	1.0	**1.5**	2.0	2.5	3.0
0.4	6.5899	6.7090	6.6597	6.6411	6.5133
0.5	6.7021	6.7411	6.7278	6.6557	6.6144
0.6	6.7924	6.7918	6.7522	6.6497	6.6289
0.8	6.9036	**6.9296**	6.9139	6.8499	6.7894
1.0	6.8131	6.8159	6.7023	6.7334	6.5057

Tabela 5.3: Rezultati parametarske analize za $n = 150$

$p\backslash\alpha$	**1.0**	1.5	2.0	2.5	3.0
0.4	9.1723	9.1711	9.1299	9.0963	9.0152
0.5	9.2410	9.3056	9.2333	9.1647	9.1373
0.6	9.3564	9.3861	9.3541	9.3236	9.1898
0.8	**9.5538**	9.5238	9.5260	9.4385	9.3529
1.0	9.4264	9.5122	9.4769	9.4734	9.2989

Tabela 5.4: Rezultati parametarske analize za $n = 200$

$p\backslash\alpha$	**1.0**	1.5	2.0
0.6	10.3107	10.4593	10.4112
0.8	**10.5923**	10.5784	10.5837
1.0	10.4997	10.5179	10.5202

Tabela 5.5: Rezultati parametarske analize za $n = 250$

$p\backslash\alpha$	1.0	**1.5**	2.0
0.6	12.9127	13.0380	13.0405
0.8	13.1417	**13.2179**	13.2169
1.0	13.0138	13.1434	13.1530

Iz dobijenih rezultata, može se primetiti da su dve kombinacije parametara $(p, \alpha) = (0.8, 1.0)$ i $(p, \alpha) = (0.8, 1.5)$ dale najbolje rezultate. Za krajnji odabir najbolje kombinacije, posmatraće se ove dve parametarske kombinacije na svih 32 parova (n, d) koji su korišćeni za testiranje. Za 21 uređeni par (n, d), kombinacija $(p, \alpha) = (0.8, 1.5)$ vodi ka najvećoj vrednosti prosečne funkcije cilja. Za 11 uređenih parova (n, d), kombinacija $(p, \alpha) = (0.8, 1.0)$ se pokazala boljom, dok su za jedan par dale jednake rezultate.

Na osnovu rezultata prikazanih u tabelama 5.1, 5.2, 5.3, 5.4, 5.5, kao najbolja parametarska kombinacija je odabrana $(p, \alpha) = (0.8, 1.5)$ i ona je korišćena u daljoj eksperimentalnoj analazi za algoritam Mod-GH-MWDDS. Ista parametarska kombinacija je korišćena pri testiranju algoritma za Mod-GH-MWDDS+.

5.3 Analiza rezultata

U tabeli 5.6 su prikazani rezultati dobijeni egzaktnim CPLEX rešavačem i još četiri algoritma za rešavanje problema MWDDS na istim instancama korišćenim u ovom master radu. Algoritmi LS, FD i VD su predloženi u radu [13], dok su CPLEX i GH-MWDDS predloženi u radu [15]. U prvoj koloni tabele 5.6 je zadat broj čvorova n, dok je u drugoj koloni zadat prosečan stepen čvora grafa d. Primetimo da se sa povećanjem parametra d dobijaju gušće mreže. Kako je za svaku kombinaciju (n, d) generisano 20 instanci, sve vrednosti po kolonama tabele 5.6 predstavljaju aritmetičku sredinu vrednosti dobijenih za 20 instanci za svaku kolonu. Kolone označene sa CPLEX sadrže rezultate dobijene CPLEX rešavačem, pri čemu prva kolona od njih predstavlja prosečnu funkciju cilja dok su u drugoj zabeležene vrednosti prosečnog procentualnog odstupanja od najboljeg rešenja dobijenim CPLEX rešavačem. Rezultati LS, FD i VD algoritama iz rada [15] dati su u narednih 6 kolona, za svaki algoritam po dve kolone. Za svaki od njih je prikazano prosečno rešenje (f) kao i prosečno vreme izvršavanja $(t(s))$ izraženo u sekundama. Naredna kolona $(|\mathcal{D}|)$

predstavlja prosečnu kardinalnost rešenja, odnosno broj dominantnih skupova koji pripadaju domatskoj particiji. Prema radu [15], sva tri algoritma pokreću se sa istim inicijalnim rešenjem. Takođe, jedine operacije koje se vrše nad tim inicijalnim rešenjem je zamena čvorova iz različitih dominantnih skupova. Na taj način, broj dominantnih skupova ostaje nepromenjen, stoga ($|\mathcal{D}|$) kolona se odnosi na sva tri algoritma. U poslednjoj koloni prikazani su rešenje i vreme izvršavanja GH-MWDDS algoritma.

U tabeli 5.7 prikazani su rezultati 4 algoritma, GH-MWDDS, GH-MWDDS+ [15] kao i Mod-GH-MWDDS i Mod-GH-MWDDS+. Struktura tabele 5.7 slična je kao u tabeli 5.6. Za svaki od ova 4 algoritma prikazano je prosečno rešenje (f) kao i prosečna veličina rešenja ($|\mathcal{D}|$). Za algoritme GH-MWDDS i GH-MWDDS+ prikazano je prosečno vreme izvršavanja $t(s)$. S obzirom na to da algoritmi Mod-GH-MWDDS i Mod-GH-MWDDS+ kao izlaz imaju 10 rešenja - za njih je prikazano prosečno vreme do dobijanja najboljeg rešenja po instanci.

Iz dobijenih rezultata može se zaključiti sledeće:

- Matematički model celobrojnog linearnog programiranja koji je rešavan egzaktnim rešavačem CPLEX se pokazao korisnim samo za instance malih dimenzija. CPLEX je dao najbolja rešenja za sve instance za koje je $n = 50$. U radu [15] je pokazano da je za instance pri čemu je $n = 50$ i $d = 15$, egzaktni rešavač dao optimalna rešenja za 8 od 20 instanci.

- Već za instance pri kojima je $n = 100$, CPLEX rešavač se pokazao neefikasnim. Za većinu instanci, počevši od $(n, d) = (150, 60)$, CPLEX ne uspeva da pronađe nijedno rešenje osim trivijalnog rešenja koje ne sadrži nijedan dominantni skup. Sa druge strane, pohlepne heuristike su dale znatno bolja rešenja u odnosu na kvalitet i u odnosu na vreme izvršavanja.

- Iz navedenih rezultata u tabeli 5.6, očigledno je da tri algoritma lokalne pretrage (LS, FD, VD) predložene od strane Pina u radu [13] daju loše rezultate, što je direktna posledica neodgovarajućeg načina primene ovih algoritama na razmatrani problem. Naime, svaki od ova tri algoritma, polazi od početnog inicijalnog rešenja. U pokušaju da se poveća funkcija cilja, čvorovi iz jednog dominantnog skupa prelaze u drugi dominantni skup i obrnuto. Primenom tih algoritama, broj dominantnih skupova u rešenju je nepromenjen. Skup rešenja koji se može dobiti ovim algoritmima je prilično mali, stoga ne uspevaju da pronađu dovoljno dobra rešenja.

- Kao što je i očekivano, veliki broj rešenja sadrži redundantne čvorove. Algoritam GH-MWDDS+ uklanja redundantne čvorove iz rešenja dobijenih GH-MWDDS algoritmom. Slično, algoritam Mod-GH-MWDDS+ uklanja redundantne čvorove iz rešenja dobijenih Mod-GH-MWDDS algoritmom. Iz rezultata prikazanih u tabeli 5.7, može se primetiti da poboljšanja ovih algoritama daju uvek bolja rešenja dok je vreme neophodno da se izvrši uklanjanje redundantnih čvorova zanemarljivo veće.

- Na osnovu rezultata iz tabela 5.6 i 5.7, može se primetiti da je algoritam Mod-GH-MWDDS+ dao bolja rešenja za svaku grupu instanci od svih algoritama do sada predloženih u literaturi. Samo je za grupu instanci za koje je $n = 50$, CPLEX rešavač pokazao bolje performanse.

- Poredeći vremena izvršavanja algoritama Mod-GH-MWDDS i Mod-GH-MWDDS+ sa odgovarajućim vremenima algoritmima GH-MWDDS i GH-MWDDS+, može se primetiti da je vreme izvršavanja algoritama predstavljenih u radu [15] nekoliko stotina puta veće od vremena izvršavanja algoritama predstavljenih u ovom master radu (videti tabelu 5.7). Razlog njihove vremenske neefikasnosti u odnosu na pohlepne algoritme iz rada [15] je njihova neoptimalna implementacija. Naime, pohlepni algoritmi u ovom master radu su implementirani sa vremenskom složenošću $O(n^2m)$, pri čemu je n broj čvorova a m broj ivica grafa. Nasuprot tome, u radu [15] pokazano da algoritam GH-MWDDS ima vremensku složenost $O(nm)$. Ključni koraci u kojima se algoritam Mod-GH-MWDDS razlikuje od algoritma GH-MWDDS, su korak 6 i koraci 15,16 i 17 iz algoritma 3. Koraci 15,16 i 17 ne povećavaju vremensku složenost algoritma, dok se u koraku 6 se nalazi *for* petlja koja se izvršava 10 puta. Stoga, ukoliko bi se algoritam Mod-GH-MWDDS implementirao po uzoru na algoritam GH-MWDDS iz rada [15], vreme izvršavanja Mod-GH-MWDDS algoritma bi bilo u proseku oko 5 puta veće u odnosu na GH-MWDDS algoritam.

Tabela 5.6: Rezultati različitih algoritama za rešavanje problema MWDDS

n	d	CPLEX		LS		FD		VD		$\|\mathcal{D}\|$	GH-MWDDS		
		f	σ(%)	f	t(s)	f	t(s)	f	t(s)		f	t(s)	$\|\mathcal{D}\|$
50	15	**2.779**	32.839	0.395	0.003	0.477	0.019	0.555	1.984	2.55	2.155	0.000	5.75
	20	**3.922**	121.088	0.649	0.004	0.825	0.055	1.014	7.651	3.55	3.353	0.001	8.05
	25	**5.098**	158.038	1.011	0.006	1.245	0.055	1.490	4.885	4.05	4.595	0.001	10.05
	30	**6.432**	200.567	1.664	0.007	2.460	0.124	2.780	13.117	6.30	5.926	0.001	12.90
	35	**7.929**	197.434	2.200	0.012	2.914	0.204	3.166	25.048	6.85	7.376	0.000	15.30
100	20	2.467	405.669	0.283	0.023	0.333	0.188	0.423	30.458	2.45	**2.784**	0.001	7.00
	30	3.202	>1000.0	0.431	0.017	0.535	0.165	0.576	31.796	2.95	**4.501**	0.003	10.90
	40	3.338	>1000.0	0.859	0.021	1.259	0.394	1.341	183.921	4.75	**6.679**	0.001	14.80
	50	3.857	>1000.0	1.361	0.038	2.018	0.796	2.407	225.704	6.00	**8.500**	0.001	18.65
	60	5.804	823.022	2.019	0.027	2.884	0.830	3.226	362.775	7.15	**11.131**	0.001	23.40
150	30	0.055	>1000.0	0.287	0.053	0.326	0.316	0.326	106.602	2.85	**4.098**	0.002	10.65
	40	0.028	>1000.0	0.557	0.050	0.670	0.595	0.745	156.994	3.80	**5.816**	0.003	14.05
	50	0.011	>1000.0	0.615	0.077	0.927	1.139	1.009	182.630	3.95	**7.479**	0.003	17.50
	60	0	>1000.0	0.848	0.046	1.506	2.576	1.521	646.268	4.90	**9.281**	0.002	21.00
	70	0	>1000.0	1.373	0.066	2.293	4.219	2.464	1028.650	6.65	**11.388**	0.004	24.45
	80	0	>1000.0	1.689	0.065	2.729	2.599	3.036	549.900	7.20	**13.508**	0.005	28.90
	90	0	>1000.0	2.209	0.055	3.817	3.793	4.193	906.030	9.05	**15.485**	0.006	32.95
200	40	0	>1000.0	0.256	0.049	0.317	0.648	0.370	81.750	2.65	**5.424**	0.004	13.65
	50	0	>1000.0	0.374	0.099	0.475	1.252	0.483	186.900	3.45	**6.784**	0.005	16.60
	60	0	>1000.0	0.678	0.112	0.697	2.083	0.917	313.850	3.75	**8.656**	0.005	20.45
	70	0	>1000.0	1.004	0.170	1.652	6.463	1.680	3112.950	5.95	**10.351**	0.006	23.65
	80	0	>1000.0	0.840	0.076	1.624	5.874	1.795	1629.400	5.35	**12.447**	0.008	27.15
	90	0	>1000.0	1.153	0.109	1.924	5.260	2.045	1364.400	5.65	**13.978**	0.012	30.55
	100	0	>1000.0	1.503	0.091	2.836	9.396	3.098	3024.830	7.65	**15.868**	0.009	34.30
250	50	0	>1000.0	0.266	0.136	0.314	0.867	0.329	557.676	2.95	**6.681**	0.016	16.85
	60	0	>1000.0	0.526	0.221	0.892	7.638	0.945	1400.788	4.55	**8.244**	0.012	19.70
	70	0	>1000.0	0.689	0.264	1.050	6.906	1.326	2380.266	5.10	**9.636**	0.013	22.95
	80	0	>1000.0	0.652	0.241	1.163	9.801	1.445	647.763	5.40	**11.520**	0.013	26.15
	90	0	>1000.0	0.841	0.324	1.448	18.394	1.591	1242.663	5.80	**12.938**	0.013	29.55
	100	0	>1000.0	1.117	0.228	1.981	20.789	2.443	2210.880	6.45	**14.733**	0.016	32.70
	120	0	>1000.0	1.259	0.146	2.577	10.442	2.781	2249.350	7.20	**18.348**	0.016	39.70
	140	0	>1000.0	2.135	0.121	4.275	16.885	4.713	6624.596	10.50	**22.478**	0.020	47.25

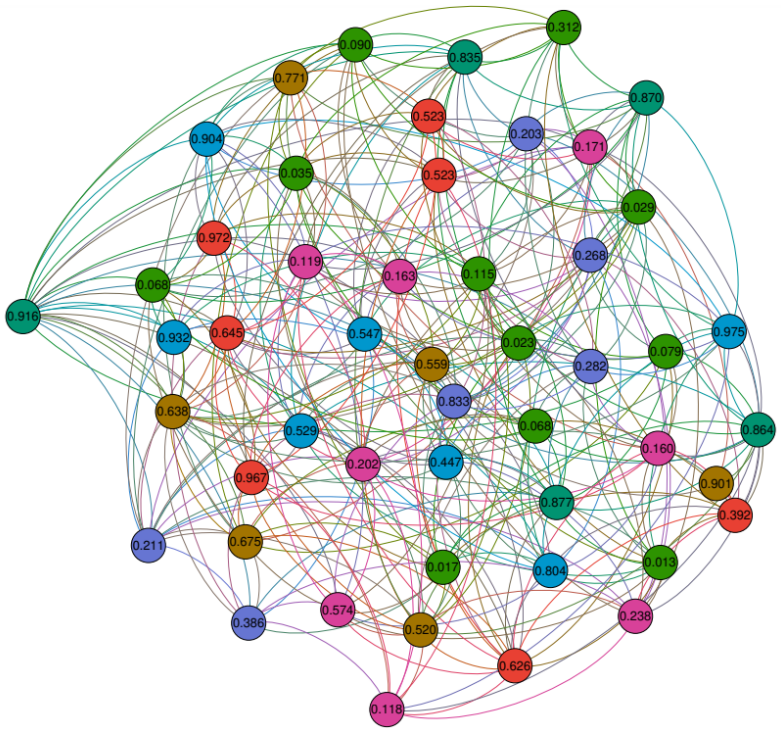
Tabela 5.7: Rezultati pohlepnih heuristika za rešavanje problema MWDDS

n	d	GH-MWDDS			GH-MWDDS+			Mod-GH-MWDDS			Mod-GH-MWDDS+		
		f	t(s)	$\|\mathcal{D}\|$	f	t(s)	$\|\mathcal{D}\|$	f	t(s)	$\|\mathcal{D}\|$	f	t(s)	$\|\mathcal{D}\|$
50	15	2.155	0.000	5.75	2.191	0.002	6.15	2.316	1.117	5.85	**2.403**	1.113	6.25
	20	3.353	0.001	8.05	3.450	0.000	8.35	3.528	1.975	8.00	**3.633**	1.194	8.45
	25	4.595	0.001	10.05	4.672	0.000	10.55	4.744	1.857	10.30	**4.899**	1.190	10.60
	30	5.926	0.001	12.90	6.042	0.000	13.35	5.998	1.589	12.65	**6.185**	1.604	13.30
	35	7.376	0.000	15.30	7.546	0.001	15.90	7.572	2.026	15.45	**7.756**	2.048	16.20
100	20	2.784	0.001	7.00	2.842	0.001	7.40	2.960	8.853	6.95	**3.080**	6.563	7.70
	30	4.501	0.003	10.90	4.580	0.001	11.15	4.720	12.764	10.85	**4.854**	7.348	11.30
	40	6.679	0.002	14.80	6.794	0.002	15.20	6.838	20.161	14.50	**6.976**	8.094	15.00
	50	8.500	0.001	18.65	8.525	0.002	19.10	8.613	11.381	18.10	**8.820**	8.889	18.85
	60	11.131	0.001	23.40	11.174	0.002	23.60	11.118	11.422	22.80	**11.297**	11.927	23.50
150	30	4.098	0.002	10.65	4.141	0.002	10.90	4.353	24.349	10.20	**4.504**	21.513	10.80
	40	5.816	0.003	14.05	5.872	0.002	14.35	6.082	25.982	13.55	**6.199**	20.384	14.10
	50	7.479	0.003	17.50	7.570	0.002	17.85	7.722	27.140	17.00	**7.880**	30.686	17.55
	60	9.281	0.002	21.00	9.371	0.005	21.45	9.481	31.467	20.35	**9.660**	34.907	21.15
	70	11.388	0.004	24.45	11.446	0.005	25.05	11.472	35.160	24.20	**11.694**	48.062	24.80
	80	13.508	0.005	28.90	13.611	0.003	29.15	13.557	44.430	28.20	**13.737**	45.861	28.70
	90	15.485	0.006	32.95	15.589	0.005	33.55	15.560	41.221	32.50	**15.772**	41.910	33.20
200	40	5.424	0.004	13.65	5.486	0.005	13.95	5.722	111.353	13.30	**5.850**	58.961	13.65
	50	6.784	0.005	16.60	6.848	0.006	17.05	7.074	42.926	16.15	**7.214**	54.680	16.80
	60	8.656	0.005	20.35	8.710	0.008	20.55	8.917	63.254	19.60	**9.056**	54.952	20.05
	70	10.351	0.006	23.65	10.395	0.008	24.10	10.606	68.510	23.05	**10.750**	95.818	23.55
	80	12.447	0.008	27.15	12.529	0.003	27.40	12.635	82.136	26.45	**12.826**	99.479	27.05
	90	13.978	0.012	30.55	14.046	0.009	31.00	14.054	68.947	29.80	**14.202**	76.240	30.45
	100	15.868	0.009	34.30	15.993	0.009	34.65	15.966	79.996	33.65	**16.186**	66.540	34.20
250	50	6.681	0.016	16.85	6.783	0.008	16.95	7.028	80.206	16.10	**7.189**	117.362	16.70
	60	8.244	0.012	19.70	8.285	0.010	19.95	8.557	103.203	19.05	**8.706**	83.523	19.60
	70	9.636	0.013	22.95	9.699	0.013	23.15	9.984	161.423	22.05	**10.120**	136.049	22.70
	80	11.520	0.013	26.15	11.571	0.013	26.50	11.797	185.269	25.40	**11.892**	154.082	26.05
	90	12.938	0.013	29.55	12.978	0.016	29.90	13.142	101.829	28.45	**13.226**	143.137	29.15
	100	14.733	0.016	32.70	14.800	0.016	33.15	14.975	140.526	31.95	**15.154**	161.956	32.70
	120	18.348	0.016	39.70	18.418	0.018	40.25	18.434	174.783	38.80	**18.593**	156.871	39.35
	140	22.478	0.020	47.25	22.514	0.020	47.70	22.639	154.486	46.20	**22.840**	168.705	46.95

U tabeli 5.8 prikazana su procentualna odstupanja (σ) prosečnih vrednosti funkcija cilja algoritama GH-MWDDS, GH-MWDDS+ i Mod-GH-MWDDS u odnosu na algoritam Mod-GH-MWDDS+. Na osnovu rezultata prikazanih u tabeli 5.8 može se zaključiti da algoritam Mod-GH-MWDDS+ ima bolje performanse u poređenju sa pohlepnim heuristikama predloženim u radu [15]. Povećanjem parametra d se smanjuje procentualno odstupanje rešenja dobijenim Mod-GH-MWDDS+ algoritmom u odnosu na pohlepne heuristike GH-MWDDS, GH-MWDDS+ i Mod-GH-MWDDS.

Tabela 5.8: Procentualna odstupanja vrednosti funkcije cilja dobijena algoritmima GH-MWDDS, GH-MWDDS+ i Mod-GH-MWDDS od vrednosti funkcije cilja dobijene Mod-GH-MWDDS+ algoritmom

n	**d**	**GH-MWDDS**	**GH-MWDDS+**	**Mod-GH-MWDDS**
		σ(%)	σ(%)	σ(%)
50	15	10.32	8.82	3.62
	20	7.70	5.03	2.89
	25	6.20	4.63	3.16
	30	4.18	2.31	3.02
	35	4.89	2.70	2.37
100	20	9.61	7.72	3.89
	30	7.27	5.64	2.76
	40	4.25	2.60	1.97
	50	3.62	3.34	2.34
	60	1.46	1.08	1.58
150	30	9.01	8.05	3.35
	40	6.17	5.27	1.88
	50	4.86	3.93	2.00
	60	3.92	2.99	1.85
	70	2.61	2.12	1.89
	80	1.66	0.91	1.31
	90	1.81	1.16	1.34
200	40	7.28	6.22	2.18
	50	5.96	5.07	1.94
	60	4.41	3.82	1.53
	70	3.71	3.30	1.33
	80	2.95	2.31	1.48
	90	1.57	1.09	1.04
	100	1.96	1.19	1.35
250	50	7.06	5.64	2.23
	60	5.30	4.83	1.71
	70	4.78	4.16	1.34
	80	3.12	2.69	0.79
	90	2.17	1.87	0.63
	100	2.77	2.33	1.18
	120	1.31	0.94	0.85
	140	1.58	1.42	0.88
Prosek		**4.54**	**3.63**	**1.85**



Slika 5.1: Rešenje jedne instance pri čemu je $(n, d) = (50, 15)$ za koju je CPLEX dobio optimalno rešenje.

Na slici 5.1 je prikazano optimalno rešenje jedne instance za koju je $(n, d) = (50, 15)$. Rešenje predstavlja domatsku particiju grafa sa 6 dominantnih skupova čiji su elementi obojeni jednom od 6 boja. Čvorovi zelene boje ne pripadaju nijednom od dominantnih skupova. Rešenje je dobijeno egzaktnim CPLEX rešavačem. Slika je preuzeta iz rada [15].

Glava 6

Zaključak

U ovom radu posmatran je problem maksimizacije životnog veka bežičnih senzorskih mreža koristeći problem maksimuma težinskih disjunktnih dominantnih skupova. Da bi se ispitale performanse predloženih algoritama iz ovog master rada, upotrebljeno je 640 instanci različitih dimenzija (uključujući i retke i guste grafove). Iz dobijenih rezultata, zaključuje se da su navedene modifikovane pohlepne heuristike dale bolje rezultate od pohlepnih heuristika predloženih u radu [15] na svakoj grupi instanci. Takođe, primećeno je da je CPLEX rešavač koristan isključivo za instance najmanjih dimenzija, dok za instance velikih dimenzija ne daje čak ni dopustivo rešenje.

Može se primetiti da je ovaj pristup rešavanja primenljiv i za nepovezane grafove. Takođe, ovaj pristup može da se primeni i na usmerene grafove. Jedini aspekt u kome bi se razlikovao takav pristup sa pristupom korišćenim u ovom radu jeste definicija dominantnog skupa (pogledati [27]), kao i načina u kojim se kreiraju dominantni skupovi u svakoj iteraciji pohlepne heuristike.

Dalja istraživanja i unapređenje rada mogu se realizovati u više pravaca:

- Algoritmi LS, FD i VD predloženi od Pina u radu [13] mogu se iskoristiti u hibridizaciji sa algoritmom Mod-GH-MWDDS+. Rešenja dobijena ovom pohlepnom heuristikom mogu se posmatrati kao inicijalna rešenja za ova tri algoritma. Zatim, primenom nekog od ova tri algoritma mogu se čvorovi iz jednog dominantnog skupa prebaciti u drugi i obrnuto, u cilju povećanja ukupne funkcije cilja.

- U cilju poboljšanja funkcije cilja Mod-GH-MWDDS+ algoritama, može se osmisliti drugačiji redosled izbacivanja redundantnih čvorova.

- Na kraju izvršavanja pohlepne heuristike Mod-GH-MWDDS+, generisano je 10 rešenja pri čemu je među njima izabrano rešenje sa najvećom funkcija cilja. Na ovaj način (ili neki drugi), može biti generisana populacija različitih rešenja. Na dobijenoj populaciji, može se ispitati koja rešenja imaju u sebi kvalitetne dominantne skupove koji bi bili iskorišćeni u finalnom rešenju.

Bibliografija

[1] Hedetniemi S.T. Cockayne E. J. Towards a theory of domination in graphs. *Networks*, 7:247–261, 1977.

[2] Cardei M.; Thai M.T.; Li Y.; Wu W. Energy-efficient target coverage in wireless sensor networks. In *Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 486–502, Miami, FL, USA, 13–17 March 2005.

[3] Slijepcevic S.; Potkonjak M. Power efficient organization of wireless sensor networks. In *Proceedings of the ICC 2001—IEEE International Conference on Communications*, volume 2, pages 472–476, Helsinki, Finland, 11–14 June 2001.

[4] Wang H.; Li Y.; Chang T.; Chang S. An effective scheduling algorithm for coverage control in underwater acoustic sensor network. *Sensors*, 18:2512, 2018.

[5] Liao C.C.; Ting C.K. An effective scheduling algorithm for coverage control in underwater acoustic sensor network. *Cybern*, 48:2245–2258, 2017.

[6] Chen Z.; Li S.; Yue W. Memetic algorithm-based multi-objective coverage optimization for wireless sensor networks. *Sensors*, 14:20500–20518, 2014.

[7] Balaji S.; Anitha M.; Rekha D.; Arivudainambi D. Energy efficient target coverage for a wireless sensor network. *Measurement*, 165:108167, 2020.

[8] D'Ambrosio C.; Iossa A.; Laureana F.; Palmieri F. A genetic approach for the maximum network lifetime problem with additional operating time slot constraints. *Soft Comput*, pages 1–7, 2020.

[9] Li J.; Potru R.; Shahrokhi F. A Performance Study of Some Approximation Algorithms for Computing a Small Dominating Set in a Graph. *Algorithms*, 13:339, 2020.

[10] Li R.; Hu S.; Liu H.; Li R.; Ouyang D.; Yin M. Multi-Start Local Search Algorithm for the Minimum Connected Dominating Set Problems. *Mathematics*, 7:1173, 2019.

[11] Bouamama S.; Blum C. An Improved Greedy Heuristic for the Minimum Positive Influence Dominating Set Problem in Social Networks. *Algorithms*, 14:79, 2021.

[12] Garey M.; Johnson D. *Computers and Intractability: A Guide to the Theory of NP Completeness*. W. H. Freeman, New York, NY, USA, 1979.

[13] Pino T.; Choudhury S.; Al-Turjman F. Dominating set algorithms for wireless sensor networks survivability. *IEEE Access*, 6:17527–17532, 2018.

[14] Islam K.; Akl S.G.; Meijer H. Maximizing the lifetime of wireless sensor networks through domatic partition. In *Proceedings of the 2009 IEEE 34th Conference on Local Computer Networks*, pages 436–442, Zurich, Switzerland, 20–23 October 2009.

[15] Balbal S.; Bouamama S.; Blum C. A Greedy Heuristic for Maximizing the Lifetime of Wireless Sensor Networks Based on Disjoint Weighted Dominating Sets. *Algorithms*, 14:486–502, 2021. on-line at: `https://doi.org/10.3390/a14060170`.

[16] Kochenberger G.A. Glover F.W. *Handbook of Metaheuristics*. International Series in Operations Research and Management Science, Kluwer Academic Publishers, 2003.

[17] E.G. Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[18] J.H. Holland. *Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

[19] M. Dorigo. *Optimization, learning and natural algorithms.* PhD thesis, Politecnico di Milano, 1992.

[20] J. Kennedy; R. Eberhard. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, page 1942–1948, Piscataway, NJ, 1995.

[21] S. Kirkpatrick; C. D. Gelatt Jr; M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[22] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

[23] N. Mladenovic; P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[24] Hansen P.; Mladenovic N.; Todosijevic R.; Hanafi S.;. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454, 2017.

[25] Blum C.; Puchinger J.; Raidl G.R.; Roli A. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing Journal*, 11(6):4135–4151, 2011.

[26] Blum C.; Puchinger J.; Raidl G.R.; Roli A. A brief survey on hybrid metaheuristics. In *4th International Conference on Bioinspired Optimization Methods and their Applications*, pages 3–16, Ljubljana, 2010.

[27] Pang C.; Zhang R.; Zhang Q.; Wang J. Dominating sets in directed graphs. *Inf. Sci.*, 180:3647–3652, 2010.