Univerzitet u Beogradu

Matematički fakultet

Master rad

Rešavanje problema optimalnog planiranja bežičnih meš mreža primenom metaheurističkih metoda

Student:
Lazar Mrkela

Mentor:
dr Zorica Stanimirović

Beograd, septembar 2016.

Mentor: **prof. dr Zorica Stanimirović**
Matematički fakultet, Univerzitet u Beogradu

Članovi komisije: **prof. dr Miodrag Živković**
Matematički fakultet, Univerzitet u Beogradu

prof. dr Miroslav Marić
Matematički fakultet, Univerzitet u Beogradu

Datum odbrane: _____

Rešavanje problema optimalnog planiranja bežičnih meš mreža primenom metaheurističkih metoda

Sažetak - Bežične meš mreže (engl. *wireless mesh network*) predstavljaju novu tehnologiju bežičnih mreža. Jedna od osnovnih karakteristika ovakvih mreža jeste niska cena instalacije i održavanja. Ovaj rad se bavi problemom optimalnog planiranjem bežičnih meš mreža u cilju minimizacije troškova instalacije mreže. Problem je rešavan pomoću dve metaheurističke metode, genetskim algoritmom i metodom promenljivih okolina. Pored toga, urađena je i hibridizacija ove dve metode. Implementirane metode su testirane na generisanim instancama, a rezultati su upoređeni sa rezultatima egzaktnog rešavača.

Ključne reči: Bežične meš mreže, optimizacija, genetski algoritam, metoda promenljivih okolina.

Metaheuristic Methods for Optimal Wireless Mesh Network Planning

Abstract - Wireless mesh networks represent a new technology for wireless networks. One of the basic characteristics of these networks is low installation and maintenance cost. This work focuses on the problem of optimal planning of wireless mesh networks with the goal of minimizing installation costs. The problem is solved with two metaheuristics, genetic algorithm and variable neighborhood search. Besides, hybridization of these two methods is presented. Implemented methods are tested on the generated instances and the results are compared with the ones obtained by exact solver.

Key words: Wireless mesh networks, optimization, genetic algorithm, variable neighborhood search.

Sadržaj

1 Uvod

Bežične mreže (engl. *wireless networks*) su već duže vreme sastavni deo svakodnevnog života. Veliki broj korisnika mobilnih telefona, široka primena radio tehnologija kao što su *WiFi* ili *Bluetooth*, govore u prilog toj činjenici. Bežični pristup Internetu je sve prisutniji, nalazi se i u parkovima, gradskom prevozu, obrazovnim ustanovama, tržnim centrima, aerodromima, itd. Želja za ostvarivanjem mrežne povezanosti bilo gde, bilo kada, jednostavno i jeftino, dovodi do stalne potrebe za razvijanjem novih tehnologija bežičnih mreža. Jedna od takvih tehnologija su i bežične meš mreže (engl. *wireless mesh networks*).

Princip funkcionisanja bežičnih meš mreža je sličan principu na kojem funkcioniše Internet. Paketi podataka se prenose sa jednog uređaja na drugi sve dok ne dođu do određene destinacije. Zato, uređaji ovakve mreže moraju da imaju sposobnost rutiranja (engl. *routing*). Dodatno, neki uređaji mogu imati sposobnost povezivanja sa drugim tipovima mreža, što omogućava njihovu integraciju.

Osnovna prednost bežičnih meš mreža u odnosu na ostale vrste mreža je mogućnost jednostavnijeg instaliranja i održavanja, što povlači niže troškove. Pored toga, uređaji ove mreže sami uspostavljaju i održavaju međusobnu povezanost, što im daje druge dve bitne karakteristike: robustnost i pouzdanost. Takođe, mreže se lako mogu proširivati inkrementalnim dodavanjem novih uređaja, što dovodi do visoke fleksibilnosti. Ovakve osobine bežičnim meš mrežama daju mogućnost različitih primena, kao što su umrežavanje u okviru preduzeća ili lokalne zajednice, sistema za nadzor, u slučaju vanrednih stanja ili na mestu nesreće, u ruralnim područjima i slično [1].

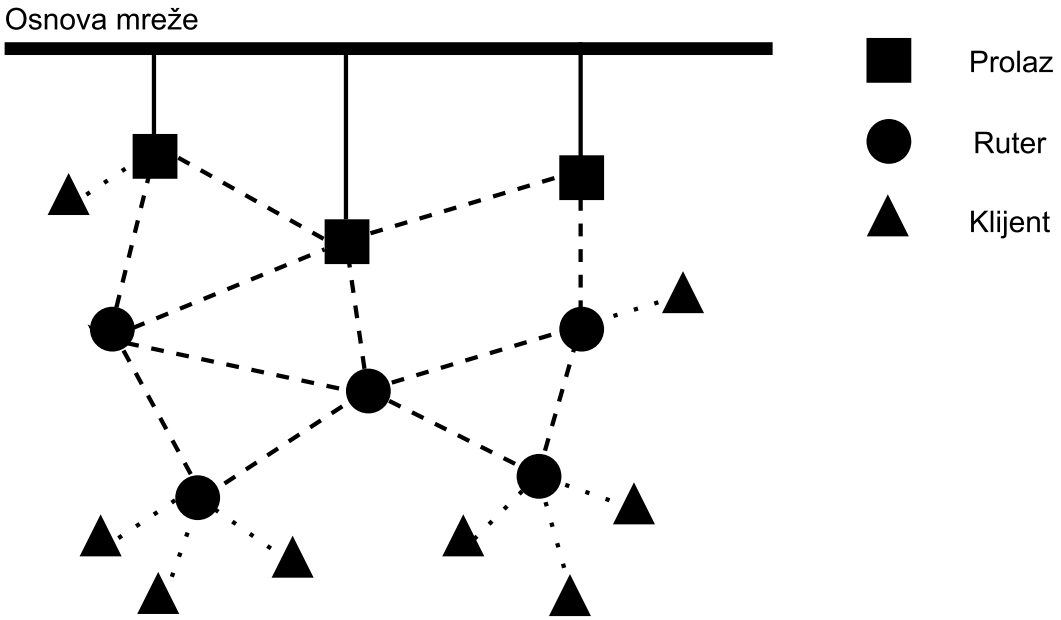
Razvijanje ove nove tehnologije zahteva istraživanja u različitim pravcima, kao što su razvoj i poboljšanje mrežnih protokola, algoritama rutiranja i poboljšanje bezbednosti. Pored toga, uočeno je da je jedan od glavnih uzroka loših performansi ovakvih mreža loše planiranje. Dalje, pažljivo planiranje može dovesti i do značajnih ušteda materijalnih sredstava potrebnih za instalaciju [7].

Kako bi se proces planiranja vršio automatski, problem se definiše matematički kao problem kombinatorne optimizacije. U zavisnosti od potreba korisnika i konkretne primene mreže, problemi se mogu definisati na različite načine. Međutim, takvi problemi planiranja su obično NP-teški, pa je za mreže većih dimenzija proces optimalnog planiranja previše spor. Jedan od načina rešavanja ovog problema jeste primena metaheurističkih metoda, koje u relativno kratkom vremenu izvršavanja mogu dati rešenja viskog kvaliteta. Neretko se rešenja dobijena metaheurističkim metodma poklapaju sa poznatim optimalnim rešenjima, ali je njihov nedostatak u tome što se optimalnost ne može dokazati. U ovom radu problem se rešava metodom promenljivih okolina i genetskim algoritmom.

U drugom poglavlju ovog rada je dat opis problema optimalnog planiranja bežičnih meš mreža, njegova matematička formulacija, kao i pregled postojećih metoda iz literature. U trećem poglavlju su date osnovne postavke metode promenljivih okolina. Četvrto poglavlje sadrži opis predložene implementacije metode promenljivih okolina za rešavanje razmatranog problema. U petom poglavlju je prikazan osnovni koncept genetskog algoritma, njegova struktura i operatori. Opis konkretne implementacije genetskog algoritma za rešavanje razmatranog problema dat je u šestom poglavlju. U sedmom poglavlju je predložena hibridizacija metode promenljivih okolina i genetskog algoritma za reševanja razmatranog problema. U osmom poglavlju su dati eksperimentalni rezultati imeplementiranih metoda i izvršeni su analiza i poređenje dobijenih rezultata. Zaključak i mogući pravci daljeg istraživanja su izneti u devetom poglavlju.

2 Problem optimalnog planiranja bežičnih meš mreža

Bežična meš mreža (BMM) se izgrađuje od tri vrste uređaja: ruteri, prolazi (engl. *gateway*) i klijenti. Pored toga što klijentima pružaju pristup mreži, ruteri i prolazi se mogu međusobno povezivati. Za razliku od rutera, prolazi imaju imaju sposobnost žičanog povezivanja. Oni imaju ulogu mrežnih prolaza ka žičanoj osnovi mreže (engl. *wired backbone*). Zbog navedenih dodatnih sposobnosti, prolazi imaju veću cenu instalacije. Prolazi se još nazivaju i pristupnim tačkama (engl. *access point*). Na slici 1 dat je šematski prikaz ovakve mreže.



Slika 1: *Šematski prikaz bežične meš mreže*

Neka su date određene lokacije na kojima se mogu instalirati ruteri i prolazi, kao i lokacije klijenata. Problem planiranja BMM koji je razmatran u ovom radu se sastoji u izboru nekog podskupa datih lokacija i izboru vrste uređaja koji će biti instaliran na određenoj lokaciji, tako da se minimizuju ukupni instalacioni troškovi. Dodatno, u obzir se moraju uzeti zahtevi za pokrivenošću svih klijenata i nesmetano rutiranje saobraćaja kroz mrežu.

2.1 Matematička formulacija problema

Matematička formulacija problema omogućava rešavanje pomoću egzaktnih rešavača. U ovom radu se koristi formulacija problema planiranja BMM koja je predložena u radu [3]. Prvo se definišu oznake koje se koriste pri formulaciji. Ulazni parametri problema su definisani u tabeli 1.

Tabela 1: *Ulazni parametri problema.*

S	skup potencijalnih lokacija za instalaciju uređaja
I	skup lokacija klijenata
S_i	podskup lokacija koje mogu da pokriju klijenta i, a koje su smeštene u niz sortiran u nerastućem poretku po jačini signala
L_i	broj elemenata skupa S_i
c_j	cena instalacije rutera na lokaciji j
p_j	dodatna cena instalacije prolaza na lokaciji j
d_i	saobraćaj generisan od strane klijenta i
u_{jl}	kapacitet veze između lokacija j i l
u_{jB}	kapacitet veze između lokacije j i osnove mreže B
v_j	pristupni kapacitet lokacije j
a_{ij}	uređaj na lokaciji j pokriva klijenta i
b_{jl}	lokacije j i l mogu biti povezane

Binarne promenljive koje se koriste pri formulaciji definisane su sa (1)-(4). Realna promenljiva $f_{jl} \geq 0$ predstavlja količinu protoka saobraćaja na vezi između lokacija j i l. Realna promenljiva $f_{jB} \geq 0$ predstavlja količinu protoka saobraćaja između lokacije j i osnove.

$$x_{ij} = \begin{cases} 1, & \text{ako je klijent } i \text{ pridružen lokaciji } j \\ 0, & \text{inače} \end{cases} \tag{1}$$

$$z_j = \begin{cases} 1, & \text{ako je uređaj instaliran na lokaciji } j \\ 0, & \text{inače} \end{cases} \tag{2}$$

$$w_{jB} = \begin{cases} 1, & \text{ako je prolaz instaliran na lokaciji } j \\ 0, & \text{inače} \end{cases} \tag{3}$$

$$y_{jl} = \begin{cases} 1, & \text{ako je uspostavljena veza između lokacije } j \text{ i } l \\ 0, & \text{inače} \end{cases} \tag{4}$$

Planiranje BMM se formuliše kao problem linearnog programiranja:

$$\min \quad \sum_{j \in S} (c_j z_j + p_j w_{jB}) \tag{5}$$

$$\text{s.t.} \quad \sum_{j \in S} x_{ij} = 1 \quad \forall i \in I \tag{6}$$

$$x_{ij} \leq z_j a_{ij} \quad \forall i \in I \ \forall j \in S \tag{7}$$

$$\sum_{i \in I} d_i x_{ij} + \sum_{l \in S} (f_{lj} - f_{jl}) - f_{jB} = 0 \ \forall j \in S \tag{8}$$

$$f_{lj} + f_{jl} \leq u_{jl} y_{jl} \quad \forall j, l \in S \tag{9}$$

$$\sum_{i \in I} d_i x_{ij} \leq v_j \quad \forall j \in S \tag{10}$$

$$f_{jB} \leq u_{jB} w_{jB} \quad \forall j \in S \tag{11}$$

$$y_{jl} \leq z_j, \ \ y_{jl} \leq z_l \quad \forall j, l \in S \tag{12}$$

$$y_{jl} \leq b_{jl} \quad \forall j, l \in S \tag{13}$$

$$z_{j_\ell^{(i)}} + \sum_{h=\ell+1}^{L_i} x_{ij_h^{(i)}} \leq 1 \ \forall \ell = 1 ... L_i - 1 \ \forall i \in I \tag{14}$$

$$x_{ij}, z_j, y_{jl}, w_{jB} \in \{0, 1\} \quad \forall i \in I \ \forall j, l \in S \tag{15}$$

Cilj problema planiranja BMM je minimizovati funkciju cilja (engl. *objective function*) koja predstavlja ukupnu cenu instalacije mreže (5). Ukupna cena je zbir troškova sa svih lokacija. Trošak na jednoj lokaciji predstavlja cenu instaliranja rutera na toj lokaciji i dodatnu cenu za instaliranje prolaza, ako je on instaliran na toj lokaciji. Prethodno definisana ograničenja imaju sledeća značenja:

(6) Garantuje se pokrivenost svih klijenata, pri čemu je svaki klijent dodeljen tačno jednoj lokaciji.

(7) Klijent može biti pridružen nekoj lokaciji samo ako je na toj lokaciji instaliran neki uređaj i ako ta lokacija pokriva tog klijenta.

(8) Garantuje balans saobraćaja kroz neku lokaciju. Prva suma predstavlja ukupan sabraćaj svih klijenata pridruženih posmatranoj lokaciji, druga predstavlja razliku saobraćaja od susednih lokacija ka posmatranoj i od posmatrane lokacije ka susednim, na kraju se oduzima saobraćaj od posmatrane lokacije ka osnovi. Ukupna vrednost mora biti jednaka nuli za svaku lokaciju.

(9) Ukupan saobraćaj između dve lokacije ne sme premašiti kapacitet veze između te dve lokacije.

(10) Ukupan saobraćaj između klijenata, pridruženih nekoj lokaciji i te lokacije ne sme biti veći od pristupnog kapaciteta te lokacije.

(11) Saobraćaj od neke lokacije ka osnovi postoji samo ako je na toj lokaciji instaliran prolaz i u tom slučaju količina saobraćaja ne sme biti veća od kapaciteta veze između te lokacije i osnove.

(12) Veza između dve lokacije može biti uspostavljena samo ako na obe lokacije postoji instaliran uređaj.

(13) Veza između lokacije i i lokacije l može biti uspostavljena samo ako to dozvoljava parametar b_{jl}.

(14) Klijent mora biti pridružen najboljoj lokaciji u odnosu na jačinu signala. Za svakog klijenta postoji uređen skup lokacija koje ga pokrivaju. Ako na nekoj lokaciji iz tog skupa postoji instaliran uređaj, onda klijent ne može biti pridružen drugoj lokaciji koja se u sortiranom nizu lokacija za datog klijenta nalazi nakon posmatrane lokacije.

(15) Promenljive odlučivanja su binarne prirode.

2.2 Postojeći načini rešavanja

Problem planiranja BMM je privukao pažnju mnogih istraživača iz oblasti optimizacije. U većini radova koji se odnose na planiranje BMM, istovremeno se razmatra više funkcija cilja, gde se uređaji mreže postavljaju s ciljem povećanja povezanosti mreže, tj. njene džinovske komponente (engl. *giant component*) i pokrivenosti klijenata. Uređaji se postavljaju u ćelije date oblasti podeljene pravougaonom mrežom. Pomenuti problem je rešavan tabu pretragom (engl. *tabu search*) [47], simuliranim kaljenjem (engl. *simulated annealing*) [40] i genetskim algoritmima (engl. *genetic algorithm*) [46].

Autori rada [36] koriste ista ograničenja za ostvarivanje zahtevanog protoka kao u modelu (5) - (15), ali u cilju smanjivanja broja prolaza. Problem razmatran u [36] je rešavan simuliranim kaljenjem i pretraživanjem usponom (engl. *hill climbing*).

Više modela BMM se razmatra u radu [8] sa ciljem istovremene optimizacije instalacionih troškova i karakteristika mreže koje se odnose na protok podataka. Karakteristike mreže koje se odnose na protok podataka mogu biti optimizovane ujednačavanjem opterećenja (engl. *load balancing*) na vezama mreže, smanjivanjem interferencije (engl. *interference*) u okviru mreže, povećanjem ukupnog protoka kroz mrežu, tj. iskorišćenosti mreže (engl. *link utilization*). Problemi razmatrani u [8] su rešavani hibridom metode roja čestica (engl. *particle swarm optimization*) i genetskog algoritma.

Detaljniji pregled postojećih načina rešavanja problema planiranja BMM dat je u radu [7]. Koliko je autoru poznato, ne postoji rad koji se bavio rešavanjem problema predstavljenog modelom (5) - (15) metaheurističkim metodama. Autori rada [3] problem su rešavali heuristikom zasnovanom na linearnoj relaksaciji, koju su konstruisali za prošireni model. Model uzima u obzir uticaj interferencije na pristupni kapacitet lokacija, kao i na veze mreže. Pored toga, u radu [3] je predstavljen i model mreže sa više radio interfejsa i radio kanala, koji je takođe rešavan heurističkim pristupom.

2.2.1 Problem pokrivanja skupa

Prilikom rešavanja problema predstavljenog modelom (5) - (15) javlja se problem pokrivanja skupa (engl. *set covering problem*), koji predstavlja jedan od fundamentalnih problema kombinatorne optimizacije [12]. Poznato je da je problem pokrivanja skupa NP-težak u jakom smislu [20].

Izučavanjem ovog problema može se doći do ideja koje će pomoći pri konstrukciji algoritama predstavljenih u ovom radu.

Problem pokrivanja skupa se obično zapisuje u matričnoj formi, traži se pokrivanje redova $m \times n$ matrice $A = (a_{ij})$ sa minimalnim brojem kolona, gde su elementi matrice nule i jedinice. Kolona j pokriva red i ako je $a_{ij} = 1$. Težinska varijanta problema (engl. *weighted set covering problem*) dodeljuje svakoj koloni cenu c_j i traži da ukupna cena kolona uključenih u rešenje bude minimalna. Binarna promenljiva x_j ima vrednost jedan ako je kolona j uključena u rešenje, inače nula. Opisani problem se može definisati kao problem celobrojnog linearnog programiranja na sledeći način [5]:

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_i \geq 1 \quad \forall i \;\; 1, ..., m \\ & x_j \in \{0, 1\} \quad \forall j \;\; 1, ..., n \end{aligned}$$

Lako se uočava sličnost sa podproblemom pokrivanja klijenata koji se javlja u planiranju BMM. Potencijalne lokacije za uređaje odgovaraju kolonama matrice A, dok klijenti odgovaraju redovima. Instaliranje uređaja odgovara uključivanju odgovarajuće kolone u rešenje. Naravno, problem pokrivanja klijenata je složeniji i uključuje ograničene kapacitete lokacija i prioritete. Kada su svi klijenti pokriveni bar jednim uređajem, potrebno je svakog klijenta dodeliti prioritetnom uređaju, pri tome može doći probijanja kapaciteta tog uređaja, pa je potrebno instalirati još drugih uređaja. S obzirom da razmatrani problem sadrži problem pokrivanja skupa, onda je on NP-težak [3].

Problem pokrivanja skupa je rešavan metaheurističkim metodama, kao što su simulirano kaljenje [11], genetski algoritmi [6], tabu pretraga [13], optimizacija mravljim kolonijama [39], optimizacija kolonijom pčela [16], kao i jednostavnim heurističkim pristupom [32].

3 Metoda promenljivih okolina

Osnovna ideja metode promenljivih okolina jeste sistematična promena okolina prilikom lokalne pretrage (engl. *local search*). Metodu su predložili Mladenović i Hansen u radu [33]. Lokalna pretraga polazi od početnog rešenja, u njegovoj okolini nalazi bolje rešenje i zatim postupak nastavlja od tog rešenja, sve dok uspeva da pronađe bolje rešenje, tj. dok ne dođe do lokalnog optimuma. Metoda promenljivih okolina zasnovana je na tri činjenice [24]:

1. Lokalni optimum u odnosu na jednu okolinu ne mora da predstavlja lokalni optimum u odnosu na drugu,
2. Globalni optimum je lokalni optimum u odnosu na sve moguće okoline,
3. Za mnoge probleme lokalni optimumi u odnosu na različite okoline su relativno blizu.

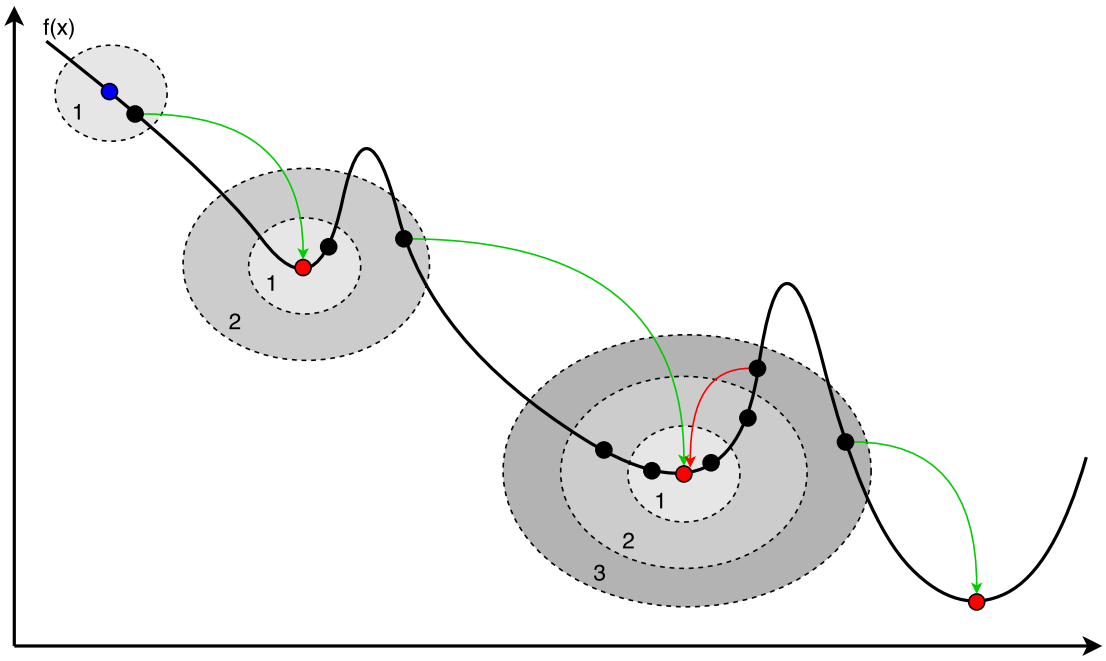
Empirijski utvrđena, poslednja činjenica ukazuje na to da lokalni optimum često daje neke informacije o globalnom. Zato se okoline lokalnog optimuma istražuju u potrazi za boljim rešenjem u nadi da će pretraga dovesti do globalnog optimuma. Izbor konkretnih okolina zavisi od problema koji se rešava.

Metoda promenljivih okolina se koristi za rešavanje brojnih problema kombinatorne i globalne optimizacije, a pregled primena može se naći u [23] i [25]. U literaturi se mogu naći brojni primeri uspešne primene metode promenljivih okolina za rešavanje diskretnih lokacijskih problema, videti [22], [29], [34], [35], [18], itd.

3.1 Osnovna varijanta metode promenljivih okolina

Osnovna varijanta metode promenljivih okolina (engl. *basic variable neighborhood search*) pored lokalnog pretraživanja, uključuje i stohastičku komponentu koja se naziva razmrdavanje (engl. *shaking*). Cilj ove komponente je da spreči da se pretraga zaglavi u nekom lokalnom optimumu. Osnovni koraci ove varijante su:

- *Inicijalizacija.* Izbor skupa okolina $N_k, k = 1, ..., k_{max}$, koji se koristi u fazi razmrdavanja. Konstruisanje početnog rešenja x. Izbor kriterijuma zaustavljanja.
- Ponavljanje narednih koraka sve dok se ne ispuni kriterijum zaustavljanja:
  1. Postaviti $k \leftarrow 1$;
  2. Ponavljati naredne korake sve dok je $k \leq k_{max}$:
     (a) *Razmrdavanje.* Generisanje slučajnog rešenja x' iz okoline $N_k(x)$;
     (b) *Lokalna pretraga.* Primeni neku metodu lokalne pretrage sa početnim rešenjem x', rezultat pretrage označi sa x'';
     (c) *Prihvatanje rešenja.* Ako je tako dobijeno rešenje x'' bolje od x, postavi $x \leftarrow x''$ i $k \leftarrow 1$; inače, postavi $k \leftarrow k + 1$;



Slika 2: *Ilustracija rada osnovne metode promenljivih okolina na primeru funkcije jedne promenljive.*

Način realizacije osnovne metode je ilustrovan na slici 2. Početno rešenje je označeno plavom tačkom, a lokalni optimumi crvenim tačkama, pri čemu je krajnji desni lokalni optimum ujedno i globalni. Crnim tačkama označena su slučajno generisana rešenja u koraku razmrdavanja. Zelenim strelicama je označeno da lokalna pretraga od datog rešenja vodi do novog trenutno najboljeg rešenja. Posebno je crvenom strelicom označeno da jedno od rešenja iz treće okoline dovodi do vraćanja na staro rešenje, zato je potrebna još jedna iteracija algoritma kako bi se došlo do globalnog optimuma.

Razmrdavanje se zasniva na trećoj od činjenica navedenih na početku ovog poglavlja. Često se u blizini jednog lokalnog optimuma nalazi još lokalnih optimuma, od kojih neki može biti globalni. Slučajnim izborom rešenja iz neke okoline lokalnog optimuma i primenom lokalne pretrage na to rešenje može se doći do nekog od tih, potencijalno boljih optimuma. Obično se razmrdavanje vrši u rastućem redosledu u odnosu na veličinu okolina. Tako se više istražuju rešenja blizu trenutnog optimuma, ali ukoliko to nije dovoljno, prelazi se na udaljenija [26].

3.2 Metoda promenljivog spusta

Deterministička promena okolina u okviru lokalne pretrage daje metodu promenljivog spusta (engl. *variable neighborhood descent*). Ova varijanta metode promenljivh okolina ne sadrži stohastičku komponentu, odnosno fazu razmrdavanja. Metoda promenljivog spusta je zasnovana na prvoj od činjenica navedenih na početku ovog poglavlja. Ideja metode jeste da kada pretraga ne može da nađe bolje rešenje od trenutnog u nekoj okolini, pokuša sa pretragom u sledećoj okolini, a da se pri pronalasku boljeg rešenja vrati na prvu okolinu. Od redosleda okolina zavisi koliko će često koja okolina biti istraživana. Osnovni koraci metode su [23]:

- *Inicijalizacija.* Izbor skupa okolina $N_k, k = 1, ..., k_{max}$. Konstruisanje početnog rešenja x.

- Postaviti $k \leftarrow 1$;
- Ponavljati naredne korake sve dok je $k \leq k_{max}$:
  1. *Istraživanje okoline.* Nađi najboljeg suseda x' rešenja x u okolini $N_k(x)$;
  2. *Prihvatanje rešenja.* Ako je tako dobijeno rešenje x' bolje od x, postavi $x \leftarrow x'$ i $k \leftarrow 1$; inače, postavi $k \leftarrow k + 1$;

3.3 Redukovana metoda promenljivih okolina

Izbacivanjem koraka lokalne pretrage iz osnovne varijante dobija se redukovana metoda promenljivih okolina (engl. *reduced variable neighborhood search*). Ovakva metoda je pogodna za rešavanje instanci velikih dimenzija za relativno kratko vreme, jer je izbačena lokalna pretraga vremenski najzahtevnija. Osnovni koraci metode su:

- *Inicijalizacija.* Izbor skupa okolina $N_k, k = 1, ..., k_{max}$, koji se koristi u fazi razmrdavanja. Konstruisanje početnog rešenja x. Izbor kriterijuma zaustavljanja.
- Ponavljanje narednih koraka sve dok se ne ispuni kriterijum zaustavljanja:
  1. Postaviti $k \leftarrow 1$;
  2. Ponavljati naredne korake sve dok je $k \leq k_{max}$:
     (a) *Razmrdavanje.* Generisanje slučajnog rešenja x' iz okoline $N_k(x)$;
     (b) *Prihvatanje rešenja.* Ako je tako dobijeno rešenje x' bolje od x, postavi $x \leftarrow x'$ i $k \leftarrow 1$; inače, postavi $k \leftarrow k + 1$;

3.4 Metoda promenljivih okolina sa dekompozicijom

Ako se u svakom koraku osnovne metode umesto lokalne pretrage u kompletnom prostoru pretrage rešava potproblem u nekom potprostoru prostora pretrage, dobija se metoda promenljivih okolina sa dekompozicijom (engl. *variable neighboorhood decomposition search*). Ideja je da se u koraku razmrdavanja na slučajan način generiše potproblem, a da se zatim na njega primeni lokalna pretraga. Osnovni koraci metode su:

- *Inicijalizacija.* Izbor skupa okolina $N_k, k = 1, ..., k_{max}$, koji se koristi u fazi razmrdavanja. Konstruisanje početnog rešenja x. Izbor kriterijuma zaustavljanja.
- Ponavljanje narednih koraka sve dok se ne ispuni kriterijum zaustavljanja:
  1. Postaviti $k \leftarrow 1$;
  2. Ponavljati naredne korake sve dok je $k \leq k_{max}$:
     (a) *Razmrdavanje.* Generisanje slučajnog rešenja x' iz okoline $N_k(x)$; Neka je y skup elemenata rešenja koji se pojavljuju u x', a ne pojavljuju se u x, odnosno postavi $y \leftarrow x' \setminus x$;

(b) *Lokalna pretraga.* Primeni neku metodu lokalne pretrage u prostoru od y, rezultat pretrage označi sa y'; Sa x'' označi odgovarajuće rešenje u celom prostoru pretrage, odnosno postavi $x'' \leftarrow (x' \setminus y) \cup y'$;

(c) *Prihvatanje rešenja.* Ako je tako dobijeno rešenje x'' bolje od x, postavi $x \leftarrow x''$ i $k \leftarrow 1$; inače, postavi $k \leftarrow k + 1$;

3.5 Iskošena metoda promenljivih okolina

Kako bi se istražili delovi prostora pretrage koji su daleko od trenutno najboljeg rešenja, dozvoljava se prihvatanje i nekih lošijih rešenja. Da li će to lošije rešenje biti prihvaćeno zavisi od toga koliko je lošije i koliko je je udaljeno od trenutno najboljeg rešenja. Tako dobijena varijanta metode naziva se iskošena (adaptivna) metoda promenljivih okolina (engl. *skewed variable neighboorhood search*). Osnovni koraci metode su:

- *Inicijalizacija.* Izbor skupa okolina $N_k, k = 1, ..., k_{max}$, koji se koristi u fazi razmrdavanja. Definisanje rastojanja r između bilo koja dva rešenja. Konstruisanje početnog rešenja x i izračunavanje vrednosti njegove funkcije cilja $f(x)$. Izbor kriterijuma zaustavljanja i parametra α. Postavi $x_{opt} \leftarrow x$ i $f_{opt} \leftarrow f(x)$;

- Ponavljanje narednih koraka sve dok se ne ispuni kriterijum zaustavljanja:

  1. Postaviti $k \leftarrow 1$;
  2. Ponavljati naredne korake sve dok je $k \leq k_{max}$:
     (a) *Razmrdavanje.* Generisanje slučajnog rešenja x' iz okoline $N_k(x)$;
     (b) *Lokalna pretraga.* Primeni neku metodu lokalne pretrage sa početnim rešenjem x', rezultat pretrage označi sa x'';
     (c) *Ažuriranje optimuma.* Ako je $f(x'') < f_{opt}$, postavi $f_{opt} \leftarrow f(x'')$ i $x_{opt} \leftarrow x''$;
     (d) *Prihvatanje rešenja.* Ako je $f(x'') - \alpha r(x, x'') < f(x)$, postavi $x \leftarrow x''$ i $k \leftarrow 1$; inače, postavi $k \leftarrow k + 1$;

3.6 Uopštena metoda promenljivih okolina

Kada se u osnovnoj varijanti kao lokalna pretraga koristi metoda promenljivog spusta, dobija se uopštena metoda promenljivih okolina (engl. *general variable neighborhood search*). Metoda promenljivog spusta se izvršava kao nezavisna procedura, tj. ima poseban skup okolina, a kao ulaz dobija početno rešenje. Osnovni koraci metode su:

- *Inicijalizacija.* Izbor skupa okolina $N_k,\ k = 1, ..., k_{max}$, koji se koristi u fazi razmrdavanja. Izbor skupa okolina $N_l,\ l = 1, ..., l_{max}$, koje se koriste u okviru metode promenljivog spusta. Konstruisanje početnog rešenja x. Izbor kriterijuma zaustavljanja.

- Ponavljanje narednih koraka sve dok se ne ispuni kriterijum zaustavljanja:

  1. Postaviti $k \leftarrow 1$;
  2. Ponavljati naredne korake sve dok je $k \leq k_{max}$:

(a) *Razmrdavanje.* Generisanje slučajnog rešenja x' iz okoline $N_k(x)$;

(b) *Metoda promenljivog spusta.* Postavi $l \leftarrow 1$; Ponavljati naredne korake sve dok je $l \leq l_{max}$:

   i. *Istraživanje okoline.* Nađi najboljeg suseda x'' rešenja x' u okolini $N_l(x')$;

   ii. *Prihvatanje rešenja.* Ako je tako dobijeno rešenje x'' bolje od x', postavi $x' \leftarrow x''$ i $l \leftarrow 1$; inače, postavi $l \leftarrow l + 1$;

(c) *Prihvatanje rešenja.* Ako je tako dobijeno rešenje x' bolje od x, postavi $x \leftarrow x'$ i $k \leftarrow 1$; inače, postavi $k \leftarrow k + 1$;

Izmene rešenja kojima se dobijaju susedi iz okoline rešenja zavise od konkretnog problema. Na primer, ako je rešenje predstavljeno binarnim nizom susedna rešenja jedne okoline se mogu dobiti invertovanjem jednog bita. Druga okolina može biti definisana invertovanjem dva bita, itd. Početno rešenje se obično konstruiše nekom pohlepnom heuristikom koja takođe zavisi od konkretnog problema. Prilikom istraživanja okoline, pronalazak najboljeg suseda (engl. *best improvement*) može biti vremenski zahtevan, pa se umesto najboljeg može tražiti prvi bolji sused (engl. *first improvement*).

4 Predložena metoda promenljivih okolina za optimalno planiranje BMM

U ovom poglavlju opisane su karakteristike predloženog VNS algoritma za rešavanje problema optimalnog planiranja BMM. Osnovna struktura predloženog VNS algoritma je data kao algoritam 1.

Početno rešenje se dobija postavljanjem rutera na sve lokacije. Prilikom istraživanja okolina, često proveravanje ostvarenog protoka kroz mrežu je vremenski zahtevno, pogotovo ako je mreža gusta. Kako bi algoritam bio efikasniji moguće je rešavanje podeliti u dve faze, VNS1 i VNS2. Obe faze algoritma su implementirane kao uopštene metode promenljivih okolina. Rešenje dobijeno u prvoj fazi se dodatno transformiše pohlepnim algoritmom, a zatim se prosleđuje drugoj fazi kao početno rešenje. Pohlepni algoritam dodaje prolaze i rutere sve dok rešenje ne postane dopustivo u odnosu na ograničenje protoka. U svakom koraku se prvo dodaje uređaj čiji je odnos cene i količine novog protoka minimalan.

Algoritam 1: *Osnovna struktura predloženog VNS algoritma*

```
1: x ← postavi rutere na sve pozicije;
2: VNS1(x);
3: while x nije dopustivo u odnosu na ograničenje protoka do
4:    dodaj uređaj sa minimalnim odnosom cene i količine novog protoka
5: VNS2(x);
6: return x;
```

Neke karakteristike su zajedničke za obe faze. Kriterijum zaustavljanja je određen kombinacijom maksimalnog broja iteracija I_{max} i maksimalnog broja uzastopnih iteracija bez poboljšanja rešenja S_{max}. U okviru lokalne pretrage, istraživanje okoline se vrši metodom prvog poboljšanja i vrši se slučajnim redosledom [28]. Rešenje dobijeno lokalnom pretragom se uvek prihvata ako je bolje od trenutno najboljeg, a sa verovatnoćom p se prihvata ako ima istu vrednost funkcije cilja, a različitu reprezentaciju. Broj takvih uzastopnih prihvatanja je ograničen na c_{max} [17]. U narednim odeljcima biće detaljno objašnjeni elementi i način realizacije koraka predložene VNS metode.

4.1 Reprezentacija rešenja

Parcijalno rešenje r se predstavlja nizom N brojeva iz skupa $\{0,1,2\}$. Pozicije u nizu odgovaraju rednim brojevima lokacija, a N je ukupan broj lokacija. Broj 1 na poziciji i označava da je instaliran ruter na i-toj lokaciji, 2 da je instaliran prolaz, a 0 da nema instaliranih uređaja na toj lokaciji. Na ovaj način, iz reprezentacije rešenja se direktno mogu dobiti vrednosti promenljivih z i w.

Kako bi se omogućila efikasna provera korektnosti rešenja i efikasno generisanje susednih rešenja, ostali podaci instance se čuvaju u odgovarajućim strukturama podataka. Povezanost lokacija se predstavlja kao graf koji je implementiran preko liste povezanosti. Svakoj lokaciji odgovara čvor grafa koji sadrži listu susednih lokacija sa kapacitetima veza.

Efikasna provera pokrivenosti klijenata i pridruživanje klijenata lokacijama omogućeni su sa dve strukture podataka. Prva predstavlja niz listi, gde svaki element niza odgovara jednoj

lokaciji, a lista sadrži klijente koje ta lokacija pokriva. Dodatno, čuva se i trenutno slobodan kapacitet te lokacije. Druga struktura predstavlja niz, gde svaki element odgovara jednom klijentu i sadrži uređenu listu lokacija koje pokrivaju tog klijenta i poziciju lokacije kojoj je klijent pridružen.

4.2 Računanje funkcije cilja

Na osnovu prethodno opisane reprezentacije rešenja, funkcija cilja se izračunava korišćenjem naredne jednačine:

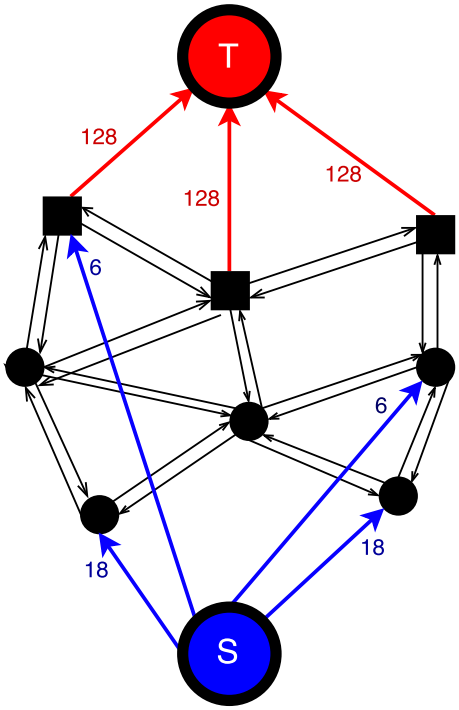
$$f(r) = \sum_{i=0}^{N-1} cost(i), \quad cost(i) = \begin{cases} c_i, & r[i] = 1 \\ c_i + p_i, & r[i] = 2 \\ 0, & r[i] = 0 \end{cases}$$

Pored vrednost funkcije cilja, potrebno je proveriti ograničenja problema i rekonstruisati kompletno rešenje. Prvo se vrši pridruživanje klijenata lokacijama, tj. određivanje vrednosti promenljivih x_{ij}. Na osnovu ograničenja (6)-(7), (10) i (14), klijent se pridružuje prvoj lokaciji sa instaliranim uređajem iz uređenog skupa lokacija koje ga pokrivaju, ako takva lokacija nije već popunila svoj pristupni kapacitet. Pri tome je moguće da ne postoji odgovarajuća lokacija kojoj dati klijent može da se pridruži - u tom slučaju se rešenje označava nedopustivim.

Kada je obezbeđena pokrivenost svih klijenata, prelazi se na ostvarivanje zahtevanog protoka kroz mrežu, tj. na određivanje vrednosti promenljivih y_{jl}, f_{jl}, f_{jB}. Na osnovu ograničenja (8)-(9) i (11)-(13), problem može da se svede na određivanje maksimalnog protoka kroz transportnu mrežu (engl. *maximum flow problem*). Mreža se modeluje usmerenim grafom, pri čemu čvorovi grafa odgovaraju lokacijama. Saobraćaj između dve lokacije može da se odvija u oba smera uz ograničenje kapaciteta veze. Stoga, veza između dve lokacije predstavlja se sa dve suprotno usmerene grane između dva odgovarajuća čvora, gde obe grane imaju dati kapacitet [41]. Za svaku lokaciju, na osnovu pridruženih klijenata, znamo ukupnu zahtevanu količinu protoka te lokacije. Takve lokacije označavamo kao izvore (engl. *source*), dok lokacije na kojima su instalirani prolazi označavamo ponorima (engl. *sink*) i kod njih posmatramo kapacitet veze ka osnovi mreže.

Potrebno je utvrditi da li je moguće ostvariti zahtevani protok od klijenata do osnove mreže. Ovakav problem se naziva problemom cirkulacije sa zahtevima (engl. *circulation with demands*) i može se svesti na problem određivanja maksimalnog protoka [31]. Kako se algoritmi za problem maksimalnog protoka mogu primeniti na mreže sa jednim izvorom i jednim ponorom, nepohodno je dodatno transformisati mrežu. Uvodi se jedan novi izvor umesto svih ostalih, kao i jedan novi ponor. Od svake lokacije se ka novom izvoru dodaje grana sa kapacitetom koji je jednak zahtevanom protoku te lokacije. Slično, od svakog prolaza dodaje se grana ka ponoru sa kapacitetom jednakim kapacitetu veze prolaza ka osnovi mreže. Ako je maksimalni protok kroz takvu mrežu jednak ukupnom zahtevanom protoku, rešenje je dopustivo.

Opisani postupak se može ilustrovati primerom na slici 3. Sa S je označen novi izvor, a sa T novi ponor. Svaki klijent zahteva 6 jedinica protoka, a svaki prolaz omogućava 128 jedinica protoka ka osnovi mreže. Kapacitet grana između lokacija nije naveden radi preglednosti, može se uzeti da je 56 jedinica. Da bi rešenje kojem odgovara mreža na slici bilo dopustivo potrebno je da maksimalni protok bude jednak 48.



Slika 3: *Bežična meš mreža sa slike 1 transformisana u transportnu mrežu.*

Pri izračunavanju maksimalnog protoka određuju se i vrednosti protoka na pojedinačnim vezama. Maksimalni protok se može odrediti u polinomijalnom vremenu nekim od poznatih algoritama, kao što je Ford-Fulkersonov algoritam [14]. Treba napomenuti da takvi algoritmi obično rade sa celobrojnim protokom (što odgovara razmatranom problemu jer se radi o binarnim podacima), dok je u matematičkoj formulaciji problema dozvoljeno da protok uzima realne vrednosti (tako je omogućena efikasnija primena egzaktnih rešavača). Cilj određivanja protoka u ovom matematičkom modelu je utvrđivanje dopustivosti rešenja, dok se za rutiranje saobraćaja kroz mrežu nakon instalacije koriste druge tehnike.

4.3 Definicija okolina

Važan korak za konstrukciju VNS algoritma za rešavanje razmatranog problema je definisanje adekvatnih struktura okolina. Jedna grupa okolina koje se koriste u okviru VNS metode se odnosi na operacije sa ruterima, a druga grupa okolina na operacije sa prolazima. Pored toga, okoline se mogu podeliti prema tome da li se koriste u fazi razmrdavanja ili fazi lokalne pretrage. Za fazu razmrdavanja se koriste sledeće okoline:

- *Ukloni ruter* (N_1): Susedno rešenje se dobija tako što se ukloni slučajno izabrani ruter. Ako se na taj način dobija skup nepokrivenih klijenata, slučajnim redosledom se dodaju ruteri na lokacije koje mogu da pokriju te klijente, sve dok se ne pokriju svi klijenti [32]. Zatim, ako postoji neostvaren protok, na susedne lokacije se dodaju ruteri sve dok se zahtevani protok ne ostvari. Lokacije su susedne ukoliko postoji veza između njih, što je definisano ulaznim podacima za konkretnu instancu problema koji se rešava.

- *Premesti prolaz* (N_2): Susedno rešenje se dobija tako što se slučajno izabrani prolaz premesti na slučajno izabranu lokaciju.

- *Ukloni\premesti* (N_3): Ova okolina predstavlja kombinaciju prethodne dve. Susedno rešenje se dobija tako što se prvo primeni postupak opisan za okolinu N_1, a zatim se na dobijeno rešenje primeni postupak za okolinu N_2.

Okoline tipa N_1, N_2, N_3 veličine k se dobijaju ponavljanjem odgovarajućih, gore opisanih postupaka k puta, gde je $k = 1, 2, ..., k_{max}$.

U okviru lokalne pretrage koriste se sledeće okoline:

- *Ukloni nepotrebni ruter* (N_4): Ukloni se ruter koji nije potreban.

- *Premesti ruter* (N_5): Ukloni se ruter sa jedne lokacije i premesti na novu lokaciju gde je cena instalacije rutera niža. Kada se ukloni ruter sa originalne pozicije dobija se skup nepokrivenih klijenata. Za novu lokaciju rutera razmatraju se samo lokacije koje pokrivaju bar jednog klijenta tog skupa. Ukoliko je takav skup prazan, razmatraju se samo susedne lokacije.

- *Postavi 1 ukloni 2* (N_6): Postavlja se novi ruter na neku lokaciju, zatim se uklanjaju dva postojeća rutera. Razmatraju se samo lokacije koje pokrivaju bar jednog klijenta iz skupa klijenata koje pokriva novi ruter ili susedne lokacije ako je takav skup prazan.

- *Ukloni 1 postavi 2* (N_7): Ukloni se jedan ruter, a zatim se postave dva čiji je zbir cena instalacije manji od prvog. Kada se ukloni ruter sa originalne pozicije dobija se skup nepokrivenih klijenata. Za lokacije dva nova rutera razmatraju se samo lokacije koje pokrivaju bar jednog klijenta tog skupa. Ukoliko je takav skup prazan, razmatraju se samo susedne lokacije.

- *Ukloni nepotrebni prolaz* (N_8): Ukloni se prolaz koji nije potreban.

- *Premesti prolaz* (N_9): Postojeći prolaz na nekoj lokaciji se premešta na susednu lokaciju, ako se pri tome cena rešenja smanjuje. Moguće je da se prolaz premesti na lokaciju na kojoj već postoji ruter ili na praznu lokaciju. Pri tome se može ukloniti i ruter sa originalne pozicije.

Sve prethodno definisane okoline sadrže samo dopustiva rešenja, pri generisanju susednog rešenja proverava se pokrivenost klijenata i ostvareni protok, ako rešenje nije dopustivo traži se novo.

4.4 Faza VNS1

U prvoj fazi, označenoj sa VNS1, se određuje optimalno postavljanje rutera uz ograničenje pokrivenosti klijenata. Pri generisanju susednih rešenja iz definisanih okolina ograničenje vezano za protok se zanemaruje. Metoda promenljivih okolina za razmrdavanje koristi samo okolinu N_1 veličine od 1 do k_{max}, a u okviru lokalne pretrage se koriste okoline N_4, N_5, N_6 i N_7. Pseudokod prve faze prikazan je algoritmom 2.

Algoritam 2: *Struktura faze VNS1*

```
1: I_count ← 0; S_count ← 0; c_count ← 0;
2: while I_count ≤ I_max and S_count ≤ S_max do
3:   k ← 1;
4:   improvement ← false;
5:   while k ≤ k_max do
6:     x' ← rešenje generisano iz okoline N_1^k(x);
7:     l ← 4;
8:     while l ≤ 7 do
9:       x'' ← prvo rešenje iz okoline N_l(x') bolje od x';
10:      if x'' postoji
11:        x' ← x''; l ← 4;
12:      else
13:        l ← l + 1;
14:    r ← slučajan broj iz intervala [0,1];
15:    if f(x') < f(x)
16:      x ← x'; k ← 1; c_count ← 0; improvement ← true;
17:    else if f(x') = f(x) and x' ≠ x and c_count ≤ c_max and r ≤ p
18:      x ← x'; k ← 1; c_count ← c_count + 1;
19:    else
20:      k ← k + 1;
21:  I_count ← I_count + 1;
22:  if improvement
23:    S_count ← 0;
24:  else
25:    S_count ← S_count + 1;
```

4.5 Faza VNS2

U okviru druge faze, označene sa VNS2, u obzir se uzima i ograničenje pokrivenosti klijenata i ograničenje zahtevanog protoka. U okviru metode promenljivog spusta, okoline se istražuju sledećim redosledom: $N_4, N_8, N_5, N_9, N_6, N_7$. Razmrdavanje se vrši sledećim redosledom: N_1, N_2, N_3. Pri tome su okoline N_1 i N_2 veličine 1, a N_3 je veličine od 1 do $k_{max} - 2$. Pseudokod druge faze dat je algoritmom 3.

Algoritam 3: *Struktura faze VNS2*

```
1: I_count ← 0 ; S_count ← 0 ; c_count ← 0;
2: redosled_okolina ← {4,8,5,9,6,7};
3: while I_count ≤ I_max and S_count ≤ S_max do
4:    k ← 1;
5:    improvement ←false;
6:    while k ≤ k_max do
7:       if k ≤ 3
8:          x' ← rešenje generisano iz okoline N_k^1(x);
9:       else
10:         x' ← rešenje generisano iz okoline N_3^{k-2}(x);
11:      l ← 1;
12:      while l ≤ 6 do
13:         x'' ← prvo rešenje iz okoline N_{redosled_okolina[l]}(x') bolje od x';
14:         if x'' postoji
15:            x' ← x''; l ← 1;
16:         else
17:            l ← l + 1;
18:      r ← slučajan broj iz intervala [0,1];
19:      if f(x') < f(x)
20:         x ← x'; k ← 1; c_count ← 0; improvement ←true;
21:      else if f(x') = f(x) and x' ≠ x and c_count ≤ c_max and r ≤ p
22:         x ← x'; k ← 1; c_count ← c_count + 1;
23:      else
24:         k ← k + 1;
25:   I_count ← I_count + 1;
26:   if improvement
27:     S_count ← 0;
28:   else
29:     S_count ← S_count + 1;
```

5 Genetski algoritam

Osnovna ideja ove metaheuristike jeste simuliranje procesa prirodne evolucije populacije jedinki. Tokom vremena, populacija evoluira i prilagođava se okruženju. Bolje prilagođene jedinke imaju veću šansu da prežive i učestvuju u razmnožavanju, a time i da prenesu svoj genetski materijal u narednu generaciju. Tako slabije jedinke i njihov genetski materijal postepeno nestaju iz populacije [27, 4].

U literaturi nalazimo brojne primene genetskog algoritma za rešavanje NP-teških optimizacionih problema [21, 37], posebno raznih diskretnih lokacijskih problema, videti [43], [2], [45], [44], [15], itd.

5.1 Osnovna struktura algoritma

Genetski algoritam polazi od inicijalne populacije jedinki, gde svaka jedinka odgovara jednom rešenju u prostoru pretrage. Zatim iterativno primenjuje genetske operatore na jedinke populacije. Prilagođenost (engl. *fitness function*) jedinke se računa na osnovu funkcije cilja. Jedinke izabrane na osnovu prilagođenosti se ukrštaju (engl. *crossover*) i stvaraju nove jedinke, koje imaju karakteristike oba roditelje. Očekuje se da se ukrštanjem dve kvalitetne jedinke nekad može dobiti još bolja jedinka. Dodatno, vrši se mutacija nekih gena jedinki, kako bi se očuvala raznovrsnost genetskog materijala. Tako dobijene jedinke zamenjuju celu prethodnu populaciju ili njen manje prilagođeni deo [6]. Osnovni koraci su:

- Generisanje inicijalne populacije
- Ponavljati naredne korake dok se ne ispuni kriterijum zaustavljanja
    1. Izračunati prilagođenost svake jedinke
    2. Selekcija
    3. Ukrštanje
    4. Mutacija

Kriterijum zaustavljanja može biti dostignut maksimalan broj generacija, maksimalan broj generacija bez poboljšanja najbolje jedinke, isteklo unapred zadato maksimalno vreme izvršavanja algoritma, pronađeno optimalno rešenje i slično. Složeniji kriterijum se može dobiti praćenjem raznovrsnosti genetskog materijala i zaustavljanjem kad raznovrsnost populacije opadne ispod unapred zadate vrednosti. Na primer, algoritam se može zaustaviti ako na svakoj poziciji u genetskom kodu imamo neku vrednost koja je ista za više od 90% jedinki populacije [38].

5.2 Inicijalna populacija

U većini slučajeva, inicijalna populacija se generiše na slučajan način. Ovaj postupak je obično jednostavan i dovodi do raznovrsnosti genetskog materijala, ali se ne može garantovati dobar kvalitet ovako dobijenih polaznih rešenja. Bolji kvalitet jedinki u početnoj populaciji se može postići upotrebom neke heuristike prilikom generisanja početne populacije ili nekog njenog dela. Pri tome se mora voditi računa o vremenskoj zahtevnosti te heuristike i posledicama po raznovrsnost genetskog materijala. Raznovrsnost genetskog materijala incijalne populacije je važna za sprečavanje preuranjene konvergencije.

5.3 Funkcija prilagođenosti

Prilagođenost jedinki se računa na osnovu funkcije cilja. Najčešće se za prilagođenost koristi direktno funkcija cilja, ali pod uslovom da njeno računanje nije vremenski zahtevno. U zavisnosti od konkretnog problema funkcija cilja se transformiše na različite načine, na primer skaliranjem. Funkcija prilagođenosti ne mora da bude ni neprekidna ni glatka.

Kodiranje rešenja može biti takvo da za neke genetske kodove ne postoji odgovarajuće dopustivo rešenje problema, jedinke sa takvim genetskim kodom se označavaju kao nekorektne. Nekorektne jedinke mogu da se uklone iz populacije, da se poprave tako da postanu korektne ili da im se smanji prilagođenost uvođenjem kaznene funkcije. Drugi pristup je da se početna populacija generiše tako da sve jedinke budu korektne, a zatim da se primenjuju genetski operatori koji čuvaju korektnost. Za neke probleme, bolji rezultati se mogu dobiti evolucijom dve populacije, jedne sa korektnim jedinkama, a druge sa nekorektnim [30].

5.4 Genetski operatori

Genetski operatori predstavljaju osnovni deo genetskog algoritma. Izbor konkretnog tipa genetskog operatora direktno zavisi od problema koji se rešava. Međutim, neki tipovi operatora mogu da se primene na različite probleme i njihove karakateristike su predstavljene u nastavku.

5.4.1 Selekcija

Selekcijom se vrši izbor jedinki populacije koje se koriste za generisanje nove populacije. Osnovni cilj je da bolje prilagođene jedinke dobiju veću šansu za prenos svog genetskog materijala u narednu generaciju. Uloga selekcije je da usmerava pretragu ka boljim rešenjima, ali sa druge strane veliki pritisak selekcije (engl. *selection pressure*) može da dovede do preuranjene konvergencije. Postoji više tipova operatora selekcije.

Kod rulet selekcije (engl. *roulette wheel selection*), verovatnoća izbora neke jedinke je proporcionalna njenoj prilagođenosti. Verovatnoća izbora jedinke se može izračunati na osnovu jednačine (16), gde je f_i prilagođenost jedinke i, a P_{size} veličina populacije. Nedostatak ovakve selekcije jeste mogućnost čestog izbora visoko prilagođenih jedinki, što može dovesti do preuranjene konvergencije

$$p_i = \frac{f_i}{\sum_{i=1}^{P_{size}} f_i} \tag{16}$$

Selekcija zasnovana na rangiranju (engl. *rank based selection*) uređuje (rangira) jedinke po prilagođenosti, a zatim se verovatnoća izbora jedinke određuje na osnovu njenog ranga. Kod linearnog rangiranja verovatnoća izbora jedinke sa rangom k se određuje na osnovu jednačine (17). Podešavanjem parametara α i β može se kontrolisati pritisak selekcije i tako prevazići problem dominacije visoko prilagođenih jedinki koji se javlja kod rulet selekcije.

$$p_k = \alpha + \beta k, \qquad \sum_{k=1}^{P_{size}} (\alpha + \beta k) = 1 \tag{17}$$

Turnirska selekcija (engl. *tournament selection*) predstavlja još jednu alternativu rulet selekciji. Bira se skup od T jedinki koje se međusobno porede. Jedinka sa najboljom prilagođenošću

predstavlja pobednika turnira koji prolazi dalje i učestvuje u procesu kreiranja naredne generacije jedinki. Parametar T ovakve selekcije predstavlja veličinu turnira i njime se određuje pritisak selekcije. Često je teško odrediti vrednost parametra T, stoga može biti povoljno da on uzima realnu vrednost. Fino gradirana turnirska selekcija (engl. *fine-grained tournament selection*) uvodi realan parametar F_{tour} koji predstavlja željenu prosečnu veličinu turnira. Ideja je da se u okviru jednog koraka selekcije koriste turniri različitih veličina, a da prosečna veličina turnira bude što bliža vrednosti F_{tour} [19].

5.4.2 Ukrštanje

Ukrštanjem se sa određenom verovatnoćom razmenjuju delovi genetskog koda dve ili više jedinki. Rezultat ukrštanja je genetski kod nove jedinke ili više njih. Jedinke na koje se primenjuje operator ukrštanja se nazivaju roditelji, a rezultujuće jedinke potomci. Ukrštanjem se dobijaju jedinke koje sadrže delove genetskih kodova svojih roditelja, a koje mogu biti i bolje prilagođene.

Kada genetski kod predstavlja nisku simbola, najjednostavniji tip ukrštanja jeste jednopoziciono ukrštanje (engl. *one-point crossover*) dve jedinke. Na slučajan način se bira jedna pozicija genetskog koda, a zatim jedinke razmenjuju sav genetski materijal desno od te tačke i na taj način se dobijaju dve nove jedinke. Kod dvopozicionog ukrštanja (engl. *two-point crossover*) se na slučajan način biraju dve pozicije, a jedinke razmenjuju materijal između te dve pozicije. Po istom principu se dalje mogu dobiti i višepoziciona ukrštanja (engl. *multi-point crossover*).

Više slučajnosti pri razmenjivanju genetskog materijala dve jedinke se može postići uniformnim ukrštanjem (engl. *uniform crossover*), što može dovesti do veće raznovrsnosti populacije. Prvo se generiše binarna niska (maska) na slučajan način, na svakoj poziciji verovatnoća simbola 1 jeste p, a simbola 0 je $1 - p$. Jedna nova jedinka uzima vrednosti genetskog koda sa svih pozicija za koje maska ima vrednost 1 od prvog roditelja, a ostale vrednosti od drugog. Za drugu jedinku važi obrnuto, 1 označava uzimanje od drugog, a 0 od prvog roditelja. Obično je parametar $p = 0.5$, ali je moguće dati naklonost nekom od roditelja.

5.4.3 Mutacija

Mutacijom se vrši izmena jednog malog dela genetskog koda. Operator mutacije održava raznovrsnost genetskog materijala populacije, jer se mutacijom mogu dobiti jedinke koje nije moguće dobiti samo primenom selekcije i ukrštanja. Mutacija igra ključnu ulogu u sprečavanju preuranjene konvergencije, tj. u izbegavanju lokalnih optimuma.

Način implementacije ovog operatora zavisi od kodiranja rešenja. Kod binarnog kodiranja mutacija se svodi na invertovanje jednog ili više slučajno izabranih bitova sa izvesnom verovatnoćom. Ako je rešenje kodirano celim ili realnim brojevima, vrednost na nekoj poziciji se može zameniti slučajnim brojem, može se dodati ili oduzeti neka vrednost i slično. Za specifične načine kodiranja, koji ne spadaju u standardne, razvijaju se posebni operatori mutacije.

Pored vrste same mutacije, bitno je odrediti i koliko se često ona dešava. Obično se uvodi parametar p_{mutate} koji označava verovatnoću da dođe do mutacije na nekoj poziciji genetskog koda.

6 Predloženi genetski algoritam za optimalno planiranje BMM

U ovom poglavlju opisane su karakteristike predloženog genetskog algoritma za rešavanje problema optimalnog planiranja BMM. Pseudokod predloženog genetskog algoritma je prikazan u algoritmu 4.

Kriterijum zaustavljanja je određen kombinacijom maksimalnog broja generacija I_{max} i maksimalnog broja uzastopnih generacija bez poboljšanja rešenja S_{max}.

Algoritam 4: *Osnovna struktura predloženog genetskog algoritma - GA*

```
1: I_count ← 0; S_count ← 0; best ← 0;
2: generiši inicijalnu populaciju;
3: while I_count ≤ I_max and S_count ≤ S_max do
4:    izračunaj prilagođenost jedinki populacije;
5:    izvrši turnirsku selekciju;
6:    izvrši jednopoziciono ukrštanje;
7:    izvrši mutaciju;
8:    I_count ← I_count + 1;
9:    current ← prilagođenost najbolje jedinke;
10:   if best < current
11:      best ← current; S_count ← 0;
12:   else
13:      S_count ← S_count + 1;
14: rešenje ← najprilagođenija jedinka;
```

U narednim sekcijama biće detaljno opisani korišćeni genetski operatori i način realizacije osnovnih koraka predloženog genetskog algoritma.

6.1 Generisanje inicijalne populacije

Inicijalna populacija se generiše slučajno i sadrži P_{size} jedinki. Način generisanja je određen sa dva parametra, p_{router} i $p_{gateway}$. Prvi označava verovatnoću da se na nekoj poziciji postavi ruter, a drugi da se postavi prolaz.

6.2 Funkcija prilagođenosti

Reprezentacija rešenja i računanje funkcije cilja su isti kao kod metode promenljivih okolina. Prilagođenost jedinke je recipročna vrednost funkcije cilja rešenja kojem odgavara ta jedinka.

Moguće je da neka rešenja ne budu dopustiva, jer nisu svi klijenti pokriveni. Takva rešenja se popravljaju pohlepnim algoritmom. Algoritam u svakom koraku postavlja ruter na poziciju koja ima minimalan odnos cene instalacije rutera i broja novih pokrivenih klijenata [6]. Na kraju se još jednom prolazi kroz rešenje i uklanjaju se nepotrebni ruteri, ali kako se na ovaj način može ukloniti ruter koji je neophodan za protok, ovaj korak se vrši sa određenom verovatnoćom p_{remove}. Međutim, rešenje i dalje može biti nedopustivo zbog neostvarenog protoka. Da bi se prevazišao ovaj problem, količina neostvarenog protoka se dodaje kao kazna (engl. *penalty function*) [48] funkciji cilja, na osnovu sledeće formule:

$$F(x) = f(x) + t \cdot flow(x)$$

$F(x)$ je nova funkcija cilja, $f(x)$ stara funkcija cilja, $flow(x)$ količina neostvarenog protoka, a t parametar koji određuje jačinu kazne.

6.3 Genetski operatori

U narednim podsekcijama opisanu su korišćeni genetski operatori. Opisani operatori ne čuvaju dopustivost rešenja, a taj problem se rešava na isti način kao i kod generisanja inicijalne populacije.

6.3.1 Selekcija

Implementiran je operator turnirske selekcije. Iz populacije se na slučajan način bira T_{size} jedinki, zatim se od izabranih se uzima najbolja u smislu vrednosti funkcije prilagođenosti. Pobednik turnira prolazi dalje i učestvuje u stvaranju naredne populacije jedinki.

6.3.2 Ukrštanje

Dve jedinke roditelji se ukrštaju jednopozicionim ukrštanjem i daju dve nove jedinke. Na slučajan način se bira jedna tačka ukrštanja. Prva jedinka potomak uzima deo koda rešenja prvog roditelja do tačke ukrštanja, a drugi deo od drugog roditelja počevši od tačke ukrštanja do kraja koda. Druga jedinka potomak se dobija spajanjem preostalih delova kodova rešenja jedinki roditelja (prvi deo od drugog roditelja, drugi od prvog). Ukrštanje se vrši sa verovatnoćom $p_{crossover}$, dok sa verovatnoćom $1 - p_{crossover}$ jedinke roditelji nepromenjeni prelaze u narednu fazu.

6.3.3 Mutacija

Do mutacije nekog gena dolazi sa verovatnoćom p_{mutate}, a vrednost parametra mutacije zavisi od vrednosti gena. Na praznu lokaciju se postavlja ruter. Ako je na poziciji bio ruter, sa verovatnoćom m_{router} se uklanja, a sa verovatnoćom $1 - m_{router}$ se postavlja prolaz. Ako je na poziciji bio prolaz, sa verovatnoćom $m_{gateway}$ se premešta na drugu slučajno izabranu poziciju, a sa verovatnoćom $1 - m_{gateway}$ se uklanja.

Konvergencijom algoritma može doći do smanjivanja raznovrsnosti genetskog materijala populacije. Može se desiti da neke pozicije gena jedinki imaju iste vrednosti kod velikog dela populacije, takve pozicije se nazivaju zaleđenim (engl. *frozen bits*) [42]. Zato se uvodi nova verovatnoća mutacije p_{frozen} za ovakve gene, uz parametar $f_{threshold}$ koji označava koliko procentualno jedinki populacije mora da ima istu vrednost pozicije kako bi ona bila označena zaleđenom.

6.4 Ostali aspekti algoritma

Kako bi se najbolje jedinke sačuvale za sledeću generaciju uvodi se elitizam. E_{count} najboljih jedinki se prenosi u narednu generaciju bez promena, takve jedinke se nazivaju elitnim. Pored

toga, duplikati jedinki se uklanjaju iz populacije. Detaljna struktura predloženog genetskog algoritma prikazana je kao algoritam 5.

Algoritam 5: *Detaljna struktura predloženog genetskog algoritma - GA*

```
1: I_count ← 0; S_count ← 0; best ← 0;
2: generiši_inicijalnu_populaciju(P_size, p_router, p_gateway);
3: while I_count ≤ I_max and S_count ≤ S_max do
4:    for svaka jedinka populacije do
5:       izračunaj_vr_funkcije_cilja();
6:       proveri_korektnost();
7:       if jedinka nekorektna zbog nepokrivenih klijenata
8:          popravi_pohlepno(p_remove);
9:       if jedinka nekorektna zbog neostvarenog protoka
10:         dodaj_kaznu(t);
11:      izračunaj_prilagođenost();
12:   ukloni_duplikate();
13:   izdvoj_elitne_jedinke(E_count);
14:   turnirska_selekcija(T_size);
15:   jednopoziciono_ukrštanje(p_crossover);
16:   nađi_zaleđene_pozicije(f_threshold);
17:   mutacija(p_mutate, p_frozen, m_router, m_gateway);
18:   I_count ← I_count + 1;
19:   current ← prilagođenost_najbolje();
20:   if best < current
21:      best ← current; S_count ← 0;
22:   else
23:      S_count ← S_count + 1;
24: rešenje ← najprilagođenija_jedinka();
```

7 Hibridizacija metode promenljivih okolina i genetskog algoritma za optimalno planiranje BMM

Jedan od načina da se izvrši hibridizacija dve opisane metaheuristike jeste da se rešenja genetskog algoritma iskoriste kao početna rešenja metode promenljivih okolina. Na kraju izvršavanja genetskog algoritma određeni broj jedinki poslednje populacije se poboljšava. Pseudokod hibridnog GA-VNS algoritma prikazan je kao algoritam 6.

Kako izvršavanje ne bi postalo previše vremenski zahtevno, poboljšavaju se samo dve jedinke. Prva je najbolja jedinka populacije, a druga se bira iz prvih 20 jedinki, kao ona jedinka koja je na najvećem rastojanju od prve. Rastojanje se računa kao broj pozicija na kojima se vrednosti rešenja razlikuju. Pre pokretanja metode promenljivih okolina, prolazi se uklanjaju iz rešenja. Kako se na taj način ne bi izgubilo najbolje rešenje genetskog algoritma, ono se prvo sačuva. Kao rezultat se vraća najbolje od ova tri rešenja.

Algoritam 6: *Struktura hibridnog GA-VNS algoritma*

```
1:  populacija ← pokreni genetski algoritam;
2:  x_1 ← najbolja jedinka populacije;
3:  x_2 ← x_1;
4:  x_3 ← x_1;
5:  for prvih 20 jedinki populacije do
6:      if rastojanje(x_1, x_3) < rastojanje(x_1, jedinka)
7:          x_3 ← jedinka;
8:  ukloni prolaze iz x_2;
9:  ukloni prolaze iz x_3;
10: x'_2 ← pokreni VNS algoritam sa početnim rešenjem x_2;
11: x'_3 ← pokreni VNS algoritam sa početnim rešenjem x_3;
12: return najbolje rešenje od rešenja x_1, x'_2 i x'_3
```

8 Eksperimentalni rezultati

Opisane metaheuristike su implementirane u programskom jeziku *C++*. Za problem određivanja maksimalnog protoka korišćena je *Boost* biblioteka za rad sa grafovima i Bojkov-Kolmogorov algoritam [9, 10]. Kao egzaktni rešavač korišćen je *IBM ILOG CPLEX Teaching Edition 12.1*, čije je vreme izvršavanja ograničeno na 4 sata. Sva testiranja izvršena su na računaru sa procesorom AMD A6 2.70 GHz i 8 GB RAM memorije.

8.1 Instance

Instance problema su generisane po uzoru na situacije iz prakse [3]. Kreiran je i generator instanci u programskom jeziku *C#*. Aplikacija omogućava zadavanje različitih parametara na osnovu kojih se generišu instance, kao i grafički prikaz generisanih instanci i rešenja problema. Parametri za generisanje instanci su dati tabelom 2. Vrednosti parametara su izražene u odgovarajućim jedinicama za konkretnu situaciju iz prakse. Na primer vrednost parametara a, d i s može biti izražena u metrima, a vrednost parametara t, c_1, c_2, c_3 u Mbps. Vrednost funkcije cilja može se računati u stotinama dolara.

Tabela 2: *Opis parametara za generisanje instanci*

N	Broj potencijalnih lokacija
M	Broj klijenata
a	Dužina stranice kvadrata na kojem će biti raspoređene kandidatske lokacije i klijenti
d	Domet veze između dve lokacije
s	Domet pristupne veze lokacija
r	Odnos između cene instaliranja rutera i prolaza
t	Saobraćaj koji zahteva svaki klijent
c_1	Kapacitet pristupne veze svake lokacije
c_2	Kapacitet veze između svake dve lokacije
c_3	Kapacitet veze između svake lokacije i osnove.

Instance problema se generišu na sledeći način. Najpre se generiše N lokacija, odnosno njihove Dekartove koordinate u ravni. Za svaku koordinatu se uzima slučajan ceo broj iz intervala $[0, a]$. Klijenti se postavljaju tako da instanca bude dopustiva. Za svakog klijenta prvo se na slučajan način bira neka lokacija, zatim se klijent na slučajan način postavlja u domet te lokacije, tj. u krug prečnika s sa centrom u toj lokaciji. Tako je svaki klijent pokriven bar jednom lokacijom. Za svakog klijenta, nalazimo lokacije koje su na rastojanju manjem od s i sortiramo ih neopadajuće po tom rastojanju (kod realnih instanci, sortiranje bi se vršilo po jačini signala). Dalje, za svake dve lokacije koje su na rastojanju manjem od d pravimo jednu vezu. Cene instalacije rutera i prolaza se lako određuju na osnovu parametra r.

Prethodno opisanim načinom generisana je prva grupa od ukupno 60 instanci, podeljenih u tri podgrupe od po 20 instanci. Podgrupe su formirane prema veličini instanci, redom male, srednje i velike. Pri generisanju instanci menjani su parametri N, M, a, r, t i njihove vrednosti su prikazane u tabelama 3, 4, 5. Parametri d, s, c_1, c_2, c_3 su fiksirani za sve instance i uzimaju

Tabela 3: *Instance malih dimenzija prve grupe*

Instanca	N	M	a	r	t
small1.txt	20	50	600	1/10	3
small2.txt	20	50	600	1/10	3
small3.txt	20	50	600	1/10	6
small4.txt	20	50	600	1/10	9
small5.txt	20	50	600	1/10	12
small6.txt	20	50	600	1/10	1
small7.txt	40	100	800	1/10	3
small8.txt	40	100	800	1/10	3
small9.txt	40	100	1000	1/10	3
small10.txt	40	100	700	1/10	3
small11.txt	40	100	600	1/10	3
small12.txt	40	100	500	1/10	3
small13.txt	60	150	1000	1/10	3
small14.txt	60	150	1000	1/10	3
small15.txt	60	150	1000	1/4	3
small16.txt	60	150	1000	1/2	3
small17.txt	60	150	1000	3/4	3
small18.txt	60	150	1000	10/11	3
small19.txt	80	180	1100	1/10	3
small20.txt	100	250	1200	1/10	3

Tabela 4: *Instance srednjih dimenzija prve grupe*

Instanca	N	M	a	r	t
medium1.txt	150	400	1300	1/10	3
medium2.txt	150	400	1300	1/10	3
medium3.txt	150	400	1200	1/10	3
medium4.txt	150	400	1100	1/10	3
medium5.txt	200	500	1500	1/10	3
medium6.txt	250	600	1700	1/10	3
medium7.txt	250	600	1700	1/10	1
medium8.txt	250	600	1700	1/10	3
medium9.txt	250	600	1700	1/10	6
medium10.txt	250	600	1700	1/10	9
medium11.txt	250	600	1700	1/10	12
medium12.txt	300	700	1800	1/10	3
medium13.txt	300	700	1800	1/10	3
medium14.txt	300	700	1800	1/4	3
medium15.txt	300	700	1800	1/2	3
medium16.txt	300	700	1800	4/5	3
medium17.txt	300	700	1800	10/11	3
medium18.txt	350	900	1900	1/10	3
medium19.txt	350	900	2000	1/10	3
medium20.txt	350	900	2400	1/10	3

vrednosti redom 250, 100, 54, 54, 128. Ovako generisane instance imaju iste cene uređaja po svim lokacijama (engl. *unicost*), jedino se razlikuje odnos cena rutera i prolaza.

Druga grupa instanci je genirasana na isti način, ali sa manje fiksiranih parametara. Instance iz ove grupe imaju uniformnu raspodelu cena instalacije uređaja po lokacijama, za rutere iz intervala [0.1, 5], a za prolaze iz [7, 11], sa korakom 0.1. Vrednosti parametara t, c_2, c_3 su određene uniformnom raspodelom iz skupova vrednosti redom {1, 2, 3}, {18, 36, 54, 72, 90}, {32, 64, 96, 128, 160}. Vrednosti parametara N, M i a za ovu grupu instanci date su u tabeli 6.

Tabela 5: *Instance velikih dimenzija prve grupe*

Instanca	N	M	a	r	t
large1.txt	500	1000	2000	1/10	3
large2.txt	600	1400	2100	1/10	3
large3.txt	800	2000	2300	1/10	3
large4.txt	900	2300	2400	1/10	3
large5.txt	1000	2500	2500	1/10	3
large6.txt	1000	2500	2600	1/10	3
large7.txt	1000	2500	2700	1/10	3
large8.txt	1000	2500	2800	1/10	3
large9.txt	900	2300	2400	4/5	3
large10.txt	900	2300	2400	1/2	3
large11.txt	900	2300	2400	1/6	3
large12.txt	700	1600	2050	1/10	3
large13.txt	700	1600	2100	1/10	3
large14.txt	700	1600	2150	1/10	3
large15.txt	700	1600	2200	1/10	3
large16.txt	800	2000	2300	1/10	3
large17.txt	800	2000	2300	1/10	4
large18.txt	800	2000	2300	1/10	5
large19.txt	800	2000	2300	1/10	6
large20.txt	800	2000	2300	1/10	8

Tabela 6: *Druga grupa instanci*

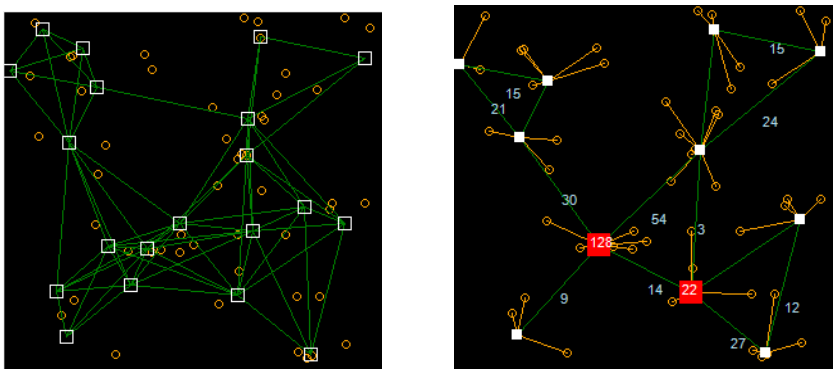
Instanca	N	M	a
random1.txt	40	100	500
random2.txt	40	120	500
random3.txt	100	250	1000
random4.txt	100	250	1000
random5.txt	250	600	1700
random6.txt	350	900	2000
random7.txt	350	900	1600
random8.txt	500	1200	1800
random9.txt	800	2000	2200
random10.txt	800	2000	2200

Primer generisane instance dat je na slici 4. U prvom redu su redom parametri N i M. Zatim sledi $2M$ redova koji se odnose na pokrivenost klijenata. Za svakog klijenta postoje dva reda: u prvom redu je broj lokacija kojima je klijent pokriven, a u drugom je sortiran niz rednih brojeva tih lokacija. Niz je sortiran neopadajuće po rastojanju od klijenta. Sledeći red sadrži broj veza između lokacija. Zatim sledi po jedan red za svaku vezu koji sadrži redne brojeve lokacija koje su povezane i parametar c_2. Zatim sledi red za svakog klijenta sa koordinatama njegove lokacije i parametrom t. Na kraju, za svaku lokaciju ide red sa koordinatama, cenom instalacije rutera, dodatnom cenom instalacije prolaza i parametrima c_1 i c_3.

Slika 4: *Primer generisane instance*

```
2  4
2
1  0
2
1  0
2
0  1
2
0  1
1
0  1  54
44  93  3
53  109  3
76  156  3
44  146  3
43  157  1  9  54  128
42  84  1  9  54  128
```

Na levoj strani slike 5 je dat grafički prikaz ulaznih podataka instace *small*2.*txt*, dok je optimalno rešenje za datu instancu prikazano na desnoj strani slike 5. Na slici (a) kvadrati predstavljaju kandidatske lokacije, krugovi klijente, a zelene linije moguće veze. Na slici (b) beli kvadrati predstavljaju instalirane rutere, a crveni kvadrati prolaze.



(a) *Instanca* **(b)** *Rešenje*

Slika 5: *Grafički prikaz instance small*6.*txt i odgovarajućeg optimalnog rešenja*

8.2 Vrednosti parametara predloženih metoda

Vrednosti parametara metode promenljivih okolina date su u tabeli 7. Vrednosti nekih parametara genetskog algoritma su prikazane tabelom 8. Različite vrednosti parametra I_{max} su postavljene za male, srednje i velike instance, redom 200, 500, 2000. Slično je urađeno i za parametar S_{max}, redom 30, 100, 250. Parametri pojedinačnih metoda hibridnog algoritma su isti kao prikazani u tabelama 7 i 8, s tim što je za GA parametar I_{max} postavljen na 500, a S_{max} na 50.

Parametri metode promenljivih okolina za drugu grupu instanci su prikazani u tabeli 7. Parametri genetskog algoritma za ovu grupu instanci su isti kao u tabeli 8, dok su I_{max} i S_{max} postavljeni redom na 2000 i 200. Hibridna metaheuristika za ove instance ima iste vrednosti parametara kao i pojedinačne metode, a za GA parametri I_{max} i S_{max} su postavljeni redom na 1000 i 50.

Sve vrednosti parametara određene su izvođenjem preliminarnih eksperimenata na podskupu instanci. Implementacija svake metode je pokretana 10 puta za instance malih i srednjih dimenzija, a 5 puta za instance velikih dimenzija prve grupe. Za instance $random1$ - $random5$ druge grupe, implementacije su pokretane 10 puta, dok su za instance $random6$ - $random10$, implementacije pokretane 5 puta.

Tabela 7: *Vrednosti parametara metode promenljivih okolina*

Parametar	Prva grupa instanci				Druga grupa instanci	
	Male		Srednje i velike			
	VNS1	VNS2	VNS1	VNS2	VNS1	VNS2
I_{max}	250	100	250	100	100	250
S_{max}	15	5	30	15	30	50
k_{max}	5	5	7	5	5	5
p	0.6	0.4	0.6	0.4	0.6	0.6
c_{max}	10	5	10	5	10	5

Tabela 8: *Vrednosti parametara genetskog algoritma.*

Parametar	Vrednost	Parametar	Vrednost
P_{size}	100	p_{frozen}	$2/N$
T_{size}	3	$f_{threshold}$	0.9
$p_{crossover}$	0.8	p_{remove}	0.5
p_{mutate}	$1/N$	E_{count}	12
m_{router}	0.8	p_{router}	0.3
$m_{gateway}$	0.8	$p_{gateway}$	0.1
t	10		

8.3 Pregled i analiza rezultata

Rezultati su prikazani tabelarno. Za svaku instancu, u koloni *sol* je prikazano rešenje egzaktnog rešavača CPLEX, sa zvezdicom su označena rešenja za koje nije dokazana optimalnost posle isteka vremenskog ograničenja. Svaka od vrednosti sa zvezdicom u koloni *sol* predstavlja gornju granicu (engl. *upper bound*) funkcije cilja optimalnog rešenja, koju je CPLEX uspeo da pronađe u zadatom vremenu izvršavanja. Crtica u koloni *sol* označava da rešavač nije uspeo da nađe rešenje usled nedostatka memorije. U koloni $t(s)$, prikazano je vreme izvršavanja CPLEX rešavača. Na osnovu više pokretanja implementiranih metoda izračunate su sledeće vrednosti:

- *best* - najbolje rešenje metode, sa oznakom *opt* ukoliko se poklapa sa poznatim optimalnim rešenjem,
- t_{tot} - prosečno ukupno vreme izvršavanja (u sekundama),
- t_{best} - prosečno vreme u kojem metoda prvi put dobije svoje najbolje rešenje (u sekundama),
- *agap* - prosečno procentualno odstupanje rešenja metode od najboljeg rešenja,
- σ - standardna devijacija (u procentima),
- gap_{bk} - procentualno odstupanje najboljeg rešenja metode od najboljeg poznatog rešenja. Najbolje poznato (engl. *best known*) rešenje se poklapa sa optimalnim, ukoliko je ono poznato. U suprotom, za najbolje poznato rešenje se uzima najmanja od sledećih vrednosti: GA *best*, VNS *best*, GA-VNS *best* i gornje granice dobijene CPLEX-om.

Dodatno, za genetski algoritam je naveden i prosečan broj generacija gen_{avg}. Rešenja *best* i *sol* su istaknuta ako predstavljaju najbolje poznato rešenje. Rešenje *best* dobijeno nekim od predloženih algoritama je podvučeno ako je bolje od gornje granice dobijene egzaktnim rešavačem.

8.3.1 Rezultati na instancama prve grupe

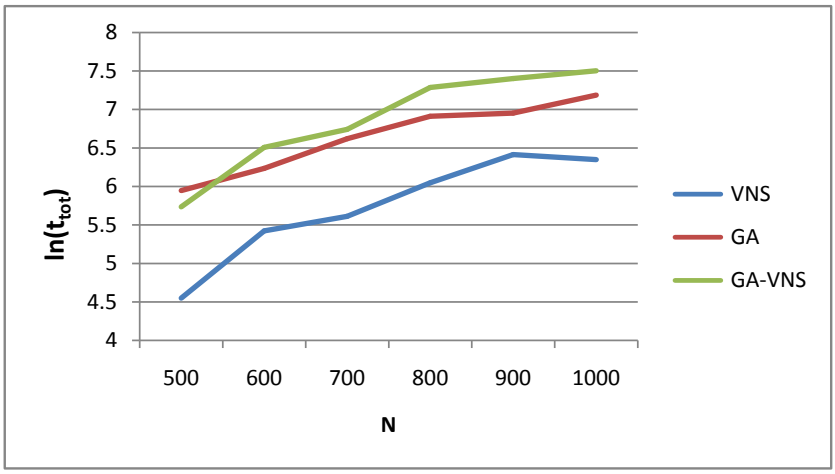
Rezultati metode promenljivih okolina su prikazani u tabelama (9)-(11). Kod instanci prve grupe, kada su ruteri postavljeni i svi klijenti pokriveni, postavljanje uređaja za ostvarivanje protoka je relativno jednostavno. Iz tog razloga se adekvatnim izborom parametara pojačava prva faza algoritma, dok se druga skraćuje. Na instancama malih i srednjih dimenzija, algoritam pronalazi optimalno rešenje na svim instancama u svih 10 pokretanja. Kod skoro svih instanci srednjih dimenzija vreme izvršavanja je kraće u odnosu na vreme izvršavanja CPLEX-a. U slučaju instanci velikih dimenzija, VNS dostiže optimalna rešenja ili gornje granice dobijene CPLEX rešavačem, a u nekm slučajevima i popravlja gornju granicu dobijenu CPLEX-om. Pri tome vreme izvršavanja ne prelazi 15 minuta, a vrednost *agap* je uvek manja od 1%, dok je σ uvek manja od 0.5.

Rezultati genetskog algoritma su prikazani u tabelama (12)-(14). Algoritam pronalazi optimalna rešenja kod svih instanci malih dimenzija, ali rezultati nisu stabilni kao kod VNS-a, što se može videti iz vrednosti u kolonama *agap* i σ. Kod 4 instance srednjih dimenzija, ne pronalazi se optimalno rešenje. Mogući uzrok tome je povećan zahtevani protok kod tih instanci, što daje veću verovatnoću generisanja nedopustivih rešenja, tj. rešenja kod kojih zahtevani protok nije ostvaren, te je algoritmu potrebno više vremena da dođe do kvalitetnih dopustivih

rešenja. Rezultati na instancama velikih dimenzija su lošiji od rezultata VNS-a, a kod tri instance dobijeni su rezultati bolji od CPLEX-a.

Hibridna metaheuristika ima za cilj poboljšanje rešenja VNS-a, stoga je testirana samo na instancama velikih dimenzija. Rezultati su prikazani u tabeli 15. Na 6 instanci su postignuti bolji rezultati u odnosu na VNS, a na dve instance lošiji. Cena ovog poboljšanja jeste duže vreme izvršavanja, koje sada ide i do 35 minuta.

Odnos prirodnog logaritma ukupnog vremena izvršavanja implementiranih metoda $ln(t_{tot})$ i parametra N instanci velikih dimenzija je prikazan na slici 6. Kada više instanci ima istu vrednost parametra N prikazana je srednja vrednost $ln(t_{tot})$ na tim instancama.

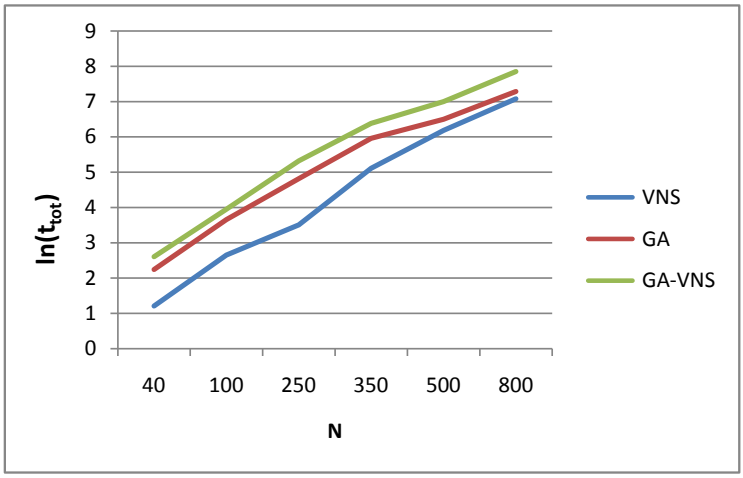


Slika 6: *Odnos $ln(t_{tot})$ implementiranih metoda i parametra N instanci velikih dimenzija.*

8.3.2 Rezultati na instancama druge grupe

Instance iz druge grupe su generisane sa više slučajnosti i pokazale su se težim za rešavanje. Rezultati metode promenljivih okolina prikazani su u tabeli 16. Kod instanci druge grupe potrebno je pojačati drugu fazu algoritma VNS, što dovodi do dužeg vremena izvršavanja. Poređenjem vremena t_{tot} i t_{best}, ispostavlja se da je parametar S_{max} moguće postaviti na manju vrednost. U nekim slučajevima početno rešenje, dobijeno u prvoj fazi, predstavlja lokalni optimum iz kojeg algoritam druge faze ne uspeva da izađe. To može biti uzrok lošijim rezultatima VNS metode u odnosu na prvu grupu instanci. Od prvih 5 instanci, algoritam ne uspeva da pronađe optimalno rešenje za dve instance, iako su one malih dimenzija. Kod preostalih 5 većih instanci, algoritam VNS za samo jednu instancu daje isto rešenje kao CPLEX, a za ostale približno rešenje. Mogući uzrok ovakvom ponašanju jeste razmrdavanje u okolini N_2, koje je ograničeno samo na dopustivi deo okoline tekućeg rešenja. Jedna od mogućih strategija za rešavanje ovog problema bi bila primena razmrdavanja u celoj okolini uz popravljanje rešenja, ukoliko se razmrdavanjem dobije nekorektno rešenje. Međutim, prethodno opisani pohlepni algoritam za popravljanje rešenja je vremenski zahtevan i njegova upotreba bi značajno povećala vreme izvršavanja metode.

Rezulati genetskog algoritma, prikazani u tabeli 17, su na većini instanci lošiji u odnosu na rezultate VNS-a. Međutim, za instancu 3, GA postiže optimalno rešenje, za razliku od VNS-a.



Slika 7: *Odnos $ln(t_{tot})$ implementiranih metoda i parametra N instanci druge grupe.*

Mogući razlog je to što genetski algoritam dopušta i pretragu nedopustivog dela prostora rešenja, jer se jedinke sa neostvarenim protokom ne odbacuju.

Rezultati hibridne metaheuristike su prikazani u tabeli 18. Hibridni algoritam je postigao jednaka ili bolja rešenja od pojedinačnih metaheuristika na svim instancama ove grupe uz očekivano duže vreme izvršavanja.

Odnos prirodnog logaritma ukupnog vremena izvršavanja implementiranih metoda $ln(t_{tot})$ i parametra N instanci druge grupe je prikazan na slici 7. Kada više instanci ima istu vrednost parametra N prikazana je srednja vrednost $ln(t_{tot})$ na tim instancama.

Tabela 9: *Rezultati metode promenljivih okolina na instancama malih dimenzija prve grupe*

Instanca	CPLEX		VNS					
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
small1.txt	**31.000**	0.044	**opt**	0.071	0.028	0.000	0.000	0.000
small2.txt	**29.000**	0.053	**opt**	0.139	0.091	0.000	0.000	0.000
small3.txt	**38.000**	0.109	**opt**	0.139	0.091	0.000	0.000	0.000
small4.txt	**47.000**	0.112	**opt**	0.153	0.099	0.000	0.000	0.000
small5.txt	**56.000**	0.141	**opt**	0.143	0.091	0.000	0.000	0.000
small6.txt	**20.000**	0.038	**opt**	0.127	0.089	0.000	0.000	0.000
small7.txt	**50.000**	0.071	**opt**	0.319	0.210	0.000	0.000	0.000
small8.txt	**49.000**	0.071	**opt**	0.370	0.264	0.000	0.000	0.000
small9.txt	**51.000**	0.095	**opt**	0.247	0.171	0.000	0.000	0.000
small10.txt	**47.000**	0.088	**opt**	0.569	0.478	0.000	0.000	0.000
small11.txt	**41.000**	2.358	**opt**	0.903	0.825	0.000	0.000	0.000
small12.txt	**38.000**	6.263	**opt**	0.830	0.760	0.000	0.000	0.000
small13.txt	**65.000**	0.390	**opt**	0.863	0.574	0.000	0.000	0.000
small14.txt	**68.000**	1.585	**opt**	0.589	0.450	0.000	0.000	0.000
small15.txt	**44.000**	0.421	**opt**	0.529	0.389	0.000	0.000	0.000
small16.txt	**36.000**	0.247	**opt**	0.511	0.377	0.000	0.000	0.000
small17.txt	**33.333**	0.278	**opt**	0.583	0.440	0.000	0.000	0.000
small18.txt	**32.400**	0.194	**opt**	0.615	0.422	0.000	0.000	0.000
small19.txt	**89.000**	2.887	**opt**	1.021	0.773	0.000	0.000	0.000
small20.txt	**99.000**	1.933	**opt**	1.843	1.452	0.000	0.000	0.000

Tabela 10: *Rezultati metode promenljivih okolina na instancama srednjih dimenzija prve grupe*

Instanca	CPLEX		VNS					
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
medium1.txt	**154.000**	24.322	**opt**	5.965	5.279	0.000	0.000	0.000
medium2.txt	**150.000**	37.116	**opt**	5.917	5.319	0.000	0.000	0.000
medium3.txt	**143.000**	74.970	**opt**	6.497	5.936	0.000	0.000	0.000
medium4.txt	**143.000**	114.607	**opt**	10.925	10.248	0.000	0.000	0.000
medium5.txt	**192.000**	39.584	**opt**	8.625	7.473	0.000	0.000	0.000
medium6.txt	**236.000**	169.704	**opt**	10.505	8.742	0.000	0.000	0.000
medium7.txt	**143.000**	457.158	**opt**	10.675	10.034	0.000	0.000	0.000
medium8.txt	**233.000**	74.857	**opt**	11.876	10.572	0.000	0.000	0.000
medium9.txt	**359.000**	147.344	**opt**	14.839	11.832	0.000	0.000	0.000
medium10.txt	**485.000**	62.819	**opt**	15.390	11.849	0.000	0.000	0.000
medium11.txt	**611.000**	91.045	**opt**	19.125	14.307	0.000	0.000	0.000
medium12.txt	**270.000**	174.961	**opt**	17.369	15.375	0.000	0.000	0.000
medium13.txt	**270.000**	142.905	**opt**	15.258	13.249	0.000	0.000	0.000
medium14.txt	**168.000**	143.992	**opt**	14.595	12.600	0.000	0.000	0.000
medium15.txt	**134.000**	9.182	**opt**	14.462	12.585	0.000	0.000	0.000
medium16.txt	**121.250**	7.638	**opt**	15.130	13.153	0.000	0.000	0.000
medium17.txt	**118.700**	11.352	**opt**	14.363	12.253	0.000	0.000	0.000
medium18.txt	**328.000**	243.509	**opt**	22.588	19.594	0.000	0.000	0.000
medium19.txt	**345.000**	408.515	**opt**	22.830	19.632	0.000	0.000	0.000
medium20.txt	**382.000**	120.088	**opt**	17.156	12.863	0.000	0.000	0.000

Tabela 11: *Rezultati metode promenljivih okolina na instancama velikih dimenzija prve grupe*

Instanca	CPLEX		VNS					
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
large1.txt	**359.000**	6012.964	**opt**	94.530	87.781	0.000	0.000	0.000
large2.txt	**465.000**	9952.312	**opt**	226.452	215.104	0.086	0.105	0.000
large3.txt	501.000*	-	**500.000**	281.666	269.374	0.280	0.204	0.000
large4.txt	704.000*	-	**506.000**	249.410	237.473	0.119	0.097	0.000
large5.txt	513.000*	-	513.000	337.730	325.550	0.039	0.078	0.195
large6.txt	**518.000***	-	**518.000**	237.257	220.979	0.039	0.077	0.000
large7.txt	**625.000***	-	**625.000**	454.991	433.618	0.192	0.120	0.000
large8.txt	627.000*	-	**624.000**	407.723	386.494	0.096	0.128	0.000
large9.txt	951.000*	-	768.000	370.840	345.131	0.052	0.104	0.130
large10.txt	961.000*	-	915.000	403.681	370.718	0.087	0.082	0.109
large11.txt	1051.000*	-	**1049.000**	478.239	437.208	0.095	0.060	0.000
large12.txt	1329.000*	-	**1328.000**	434.526	380.457	0.286	0.324	0.000
large13.txt	-	-	**704.000**	639.130	608.615	0.114	0.057	0.000
large14.txt	-	-	**236.500**	623.553	597.078	0.761	0.493	0.000
large15.txt	-	-	278.000	650.314	622.114	0.576	0.288	0.361
large16.txt	-	-	494.000	537.452	506.680	0.324	0.243	0.203
large17.txt	-	-	**771.000**	700.594	665.905	0.130	0.116	0.000
large18.txt	-	-	**787.000**	542.581	510.255	0.280	0.203	0.000
large19.txt	-	-	806.000	563.140	527.070	0.149	0.093	0.124
large20.txt	-	-	**824.000**	501.352	465.740	0.073	0.059	0.000

Tabela 12: *Rezultati genetskog algoritma na instancama malih dimenzija prve grupe*

Instanca	CPLEX		GA						
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	gen_{avg}	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
small1.txt	**31.000**	0.044	**opt**	1.400	0.242	10.100	0.000	0.000	0.000
small2.txt	**29.000**	0.053	**opt**	2.130	0.252	20.600	0.000	0.000	0.000
small3.txt	**38.000**	0.109	**opt**	2.250	0.414	22.200	0.000	0.000	0.000
small4.txt	**47.000**	0.112	**opt**	2.180	0.338	21.400	0.213	0.638	0.000
small5.txt	**56.000**	0.141	**opt**	0.587	0.130	23.800	0.536	0.818	0.000
small6.txt	**20.000**	0.038	**opt**	0.505	0.051	20.600	0.000	0.000	0.000
small7.txt	**50.000**	0.071	**opt**	4.780	0.686	21.200	0.000	0.000	0.000
small8.txt	**49.000**	0.071	**opt**	0.980	0.137	21.300	0.000	0.000	0.000
small9.txt	**51.000**	0.095	**opt**	1.130	0.326	26.200	0.000	0.000	0.000
small10.txt	**47.000**	0.088	**opt**	0.999	0.163	21.900	0.000	0.000	0.000
small11.txt	**41.000**	2.358	**opt**	1.170	0.338	26.000	0.000	0.000	0.000
small12.txt	**38.000**	6.263	**opt**	1.390	0.532	30.100	0.263	0.789	0.000
small13.txt	**65.000**	0.390	**opt**	8.390	2.740	27.000	1.230	3.690	0.000
small14.txt	**68.000**	1.585	**opt**	1.850	0.631	28.200	0.588	0.720	0.000
small15.txt	**44.000**	0.421	**opt**	1.690	0.498	26.100	0.909	1.110	0.000
small16.txt	**36.000**	0.247	**opt**	1.760	0.599	28.100	0.556	1.110	0.000
small17.txt	**33.333**	0.278	**opt**	1.790	0.641	28.900	0.300	0.900	0.000
small18.txt	**32.400**	0.194	**opt**	1.760	0.598	28.100	0.617	1.230	0.000
small19.txt	**89.000**	2.887	**opt**	12.700	4.260	27.300	0.112	0.337	0.000
small20.txt	**99.000**	1.933	**opt**	17.100	7.140	31.200	0.606	0.670	0.000

Tabela 13: *Rezultati genetskog algoritma na instancama srednjih dimenzija prve grupe*

Instanca	CPLEX		GA						
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	gen_{avg}	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
medium1.txt	**154.000**	24.322	**opt**	24.702	10.050	164.500	0.195	0.298	0.000
medium2.txt	**150.000**	37.116	**opt**	23.923	9.020	157.800	0.933	0.611	0.000
medium3.txt	**143.000**	74.970	**opt**	26.877	12.199	175.100	0.559	0.685	0.000
medium4.txt	**143.000**	114.607	**opt**	25.592	10.509	164.800	0.839	0.523	0.000
medium5.txt	**192.000**	39.584	**opt**	33.281	14.464	173.500	0.573	0.491	0.000
medium6.txt	**236.000**	169.704	**opt**	45.175	22.212	192.500	0.297	0.271	0.000
medium7.txt	**143.000**	457.158	**opt**	48.444	24.320	198.000	0.629	0.660	0.000
medium8.txt	**233.000**	74.857	**opt**	51.279	27.815	214.100	0.644	0.346	0.000
medium9.txt	**359.000**	147.344	360.000	45.805	22.013	188.700	0.472	0.374	0.278
medium10.txt	**485.000**	62.819	486.000	65.757	42.048	273.900	0.453	0.288	0.206
medium11.txt	**611.000**	91.045	613.000	71.926	47.964	294.500	0.522	0.290	0.326
medium12.txt	**270.000**	174.961	271.000	74.833	47.172	262.700	0.480	0.406	0.369
medium13.txt	**270.000**	142.905	**opt**	66.424	37.290	222.100	0.296	0.277	0.000
medium14.txt	**168.000**	143.992	**opt**	69.023	38.565	221.900	0.655	0.494	0.000
medium15.txt	**134.000**	9.182	**opt**	60.087	29.925	195.900	1.343	0.870	0.000
medium16.txt	**121.250**	7.638	**opt**	61.572	31.370	197.900	0.825	0.738	0.000
medium17.txt	**118.700**	11.352	**opt**	70.318	39.171	226.200	0.758	0.454	0.000
medium18.txt	**328.000**	243.509	**opt**	81.509	47.700	236.800	0.183	0.149	0.000
medium19.txt	**345.000**	408.515	**opt**	94.244	59.137	258.000	0.432	0.503	0.000
medium20.txt	**382.000**	120.088	**opt**	70.653	35.734	197.900	0.445	0.236	0.000

Tabela 14: *Rezultati genetskog algoritma na instancama velikih dimenzija prve grupe*

Instanca	CPLEX		GA						
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	gen_{avg}	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
large1.txt	**359.000**	6012.964	360.000	382.326	263.695	788.300	0.611	0.461	0.279
large2.txt	**465.000**	9952.312	469.000	510.028	346.417	763.000	0.832	0.375	0.860
large3.txt	501.000*	-	509.000	898.144	687.588	1083.800	0.511	0.423	1.800
large4.txt	704.000*	-	511.000	691.615	482.882	814.400	0.626	0.624	0.988
large5.txt	513.000*	-	522.000	708.980	500.936	843.600	0.153	0.307	1.953
large6.txt	**518.000***	-	524.000	718.174	512.804	876.600	0.382	0.209	1.158
large7.txt	**625.000***	-	632.000	892.636	655.803	935.600	0.538	0.348	1.120
large8.txt	627.000*	-	629.000	998.311	749.889	999.200	0.636	0.389	0.801
large9.txt	951.000*	-	778.000	948.458	709.640	965.800	0.463	0.450	1.434
large10.txt	961.000*	-	928.000	949.716	707.929	950.600	0.129	0.081	1.532
large11.txt	1051.000*	-	1064.000	1052.618	804.030	1050.200	0.244	0.401	1.430
large12.txt	1329.000*	-	1346.000	1218.922	988.261	1284.200	0.282	0.202	1.355
large13.txt	-	-	712.000	1138.423	861.208	1013.500	0.801	0.491	1.136
large14.txt	-	-	245.500	974.505	701.623	883.800	2.933	1.489	3.805
large15.txt	-	-	290.000	1017.717	749.269	934.600	1.069	0.823	4.693
large16.txt	-	-	505.000	1067.584	802.770	998.300	0.871	0.640	2.434
large17.txt	-	-	782.000	1550.538	1241.743	1242.000	0.435	0.304	1.427
large18.txt	-	-	799.000	1175.852	879.573	978.300	0.526	0.284	1.396
large19.txt	-	-	812.000	1417.300	1116.767	1165.400	0.936	0.474	0.870
large20.txt	-	-	831.000	1187.710	890.574	985.800	0.710	0.435	0.850

Tabela 15: *Rezultati hibridne metaheuristike na instancama velikih dimenzija prve grupe*

Instanca	CPLEX		GA-VNS					
	sol	t(s)	best	$t_{tot}(s)$	gen_{avg}	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
large1.txt	**359.000**	6012.964	**opt**	310.030	164.600	0.000	0.000	0.000
large2.txt	**465.000**	9952.312	**opt**	670.423	217.800	0.043	0.086	0.000
large3.txt	501.000*	-	**500.000**	1020.031	252.200	0.080	0.098	0.000
large4.txt	704.000*	-	**506.000**	821.802	241.600	0.079	0.097	0.000
large5.txt	513.000*	-	**512.000**	845.555	244.600	0.117	0.096	0.000
large6.txt	**518.000***	-	**518.000**	730.099	247.000	0.000	0.000	0.000
large7.txt	**625.000***	-	**625.000**	1222.939	259.400	0.224	0.128	0.000
large8.txt	627.000*	-	**624.000**	1142.016	279.200	0.000	0.000	0.000
large9.txt	951.000*	-	**767.000**	1389.184	362.200	0.078	0.104	0.000
large10.txt	961.000*	-	**914.000**	1488.821	368.400	0.131	0.082	0.000
large11.txt	1051.000*	-	**1049.000**	1640.994	406.000	0.095	0.085	0.000
large12.txt	1329.000*	-	**1328.000**	2062.596	461.600	0.045	0.060	0.000
large13.txt	-	-	**704.000**	1599.697	271.200	0.114	0.057	0.000
large14.txt	-	-	237.500	1478.879	253.000	0.253	0.206	0.421
large15.txt	-	-	**277.000**	1826.134	280.600	0.433	0.270	0.000
large16.txt	-	-	**493.000**	1676.221	255.200	0.243	0.298	0.000
large17.txt	-	-	**771.000**	2020.885	311.000	0.104	0.052	0.000
large18.txt	-	-	788.000	2037.459	366.000	0.102	0.095	0.127
large19.txt	-	-	**805.000**	1660.845	273.800	0.124	0.192	0.000
large20.txt	-	-	**824.000**	1593.227	280.600	0.024	0.049	0.000

Tabela 16: *Rezultati metode promenljivih okolina na drugoj grupi instanci*

Instanca	CPLEX		VNS					
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
random1.txt	**25.400**	0.398	**opt**	2.162	0.470	0.000	0.000	0.000
random2.txt	**37.000**	5.505	**opt**	5.207	1.863	0.000	0.000	0.000
random3.txt	**110.600**	9.519	111.000	29.039	6.055	0.216	0.177	0.362
random4.txt	**96.400**	0.634	**opt**	6.940	1.613	0.000	0.000	0.000
random5.txt	**190.400**	559.960	190.600	33.377	6.669	0.000	0.000	0.105
random6.txt	**345.300**	133.431	345.600	116.535	57.003	0.081	0.069	0.087
random7.txt	**237.200***	-	**237.200**	236.779	73.435	0.118	0.126	0.000
random8.txt	**314.100***	-	314.700	485.559	270.217	0.680	0.770	0.191
random9.txt	**431.500**	10920.205	432.300	959.904	416.527	0.083	0.054	0.185
random10.txt	**343.000***	-	343.400	1474.888	712.264	0.012	0.023	0.117

Tabela 17: *Rezultati genetskog algoritma na drugoj grupi instanci*

Instanca	CPLEX		GA						
	sol	t(s)	best	$t_{tot}(s)$	$t_{best}(s)$	gen_{avg}	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
random1.txt	**25.400**	0.398	**opt**	7.767	0.881	222.900	0.157	0.315	0.000
random2.txt	**37.000**	5.505	**opt**	11.403	3.378	280.700	0.703	0.422	0.000
random3.txt	**110.600**	9.519	**opt**	41.018	22.454	439.200	0.687	1.120	0.000
random4.txt	**96.400**	0.634	**opt**	36.631	18.771	408.100	1.535	0.947	0.000
random5.txt	**190.400**	559.960	191.000	123.843	79.026	549.800	1.199	1.587	0.315
random6.txt	**345.300**	133.431	350.000	416.259	348.577	1230.200	0.566	0.630	1.361
random7.txt	**237.200***	-	239.100	360.614	293.944	1078.800	2.083	2.323	0.801
random8.txt	**314.100***	-	337.100	666.859	569.209	1353.800	1.115	0.710	7.323
random9.txt	**431.500**	10920.205	453.900	1478.799	1339.013	1675.400	3.886	2.621	5.191
random10.txt	**343.000***	-	361.800	1447.554	1303.507	1680.000	2.742	5.481	5.481

Tabela 18: *Rezultati hibridne metaheuristike na drugoj grupi instanci*

Instanca	CPLEX		GA-VNS					
	sol	t(s)	best	$t_{tot}(s)$	gen_{avg}	$agap(\%)$	$\sigma(\%)$	$gap_{bk}(\%)$
random1.txt	**25.400**	0.398	**opt**	9.132	80.700	0.000	0.000	0.000
random2.txt	**37.000**	5.505	**opt**	20.102	91.400	0.000	0.000	0.000
random3.txt	**110.600**	9.519	**opt**	77.105	179.100	0.145	0.177	0.000
random4.txt	**96.400**	0.634	**opt**	35.093	149.700	0.000	0.000	0.000
random5.txt	**190.400**	559.960	190.600	205.084	322.100	0.000	0.000	0.105
random6.txt	**345.300**	133.431	345.500	560.709	460.000	0.081	0.085	0.058
random7.txt	**237.200***	-	**237.200**	628.581	583.600	0.067	0.068	0.000
random8.txt	**314.100***	-	314.300	1100.054	667.800	0.458	0.567	0.064
random9.txt	**431.500**	10920.205	432.200	2405.870	917.200	0.130	0.094	0.162
random10.txt	**343.000***	-	**343.000**	2763.478	889.000	0.082	0.079	0.000

9 Zaključak

U radu je razmatran problem planiranja bežičnih meš mreža koji za cilj ima minimizaciju ukupnih troškova instalacije mreže pri zadatim uslovima. Problem je rešavan pomoću dve metaheurističke metode, metodom promenljivih okolina i genetskim algoritmom. Elementi obe heurističke metode su prilagođeni karakteristikama razmatranog problema. Korišćen je adekvatan način kodiranja rešenja i implementiran je efikasan algoritam za računanje funkcije cilja. U slučaju metode promenljivih okolina definisane su odgovarajuće strukture okolina, dok su u slučaju genetskog algoritma korišćeni adekvatni genetski operatori. Predloženi genetski algoritam i metoda promenljivih okolina su kombinovane na osnovu njihovih komplementarnih karakteristika.

U cilju testiranja implementiranih metoda generisane su dve grupe instanci. Prva grupa je generisana na osnovu situacija iz prakse, dok je prilikom generisanja druge grupe instanci uključeno više slučajnosti. Za generisanje instanci je kreirana posebna aplikacija.

Na osnovu prikazanih eksperimentalnih rezultata, dolazi se do zaključka da su implementirane metode pogodne za rešavanje predstavljenog problema. Za instance koje nisu rešene egzaktnim rešavačem, pronađena su rešenja u razumnom vremenu. Neka rešenja za koja nije dokazana optimalnost su poboljšana. Metoda promenljivih okolina se pokazala kao bolji pristup rešavanju razmatranog problema u odnosu na genetski algoritam, dok su neka rešenja dodatno poboljšana hibridizacijom ove dve metode.

Razmatrani problem je prvi put rešavan predloženim metodama, što je osnovni doprinos ovog rada. Dalje istraživanje može da obuhvati sledeće:

- Definisanje novih okolina za razmrdavanje kod metode promenljivih okolina i njihovo testiranje,
- Dodatno eksperimentalno podešavanje parametara genetskog algoritma,
- Paralelizacija predstavljenih metoda u cilju poboljšanja vremena izvršavanja,
- Primena predstavljenih metoda na slične formulacije problema planiranja bežičnih meš mreža.

Literatura

[1] Akyildiz, I. and Wang, X., (2009). *Wireless mesh networks*. John Wiley & Sons, pp. 1-13.

[2] Alp, O., Erkut, E. and Drezner, Z., (2003). An efficient genetic algorithm for the p-median problem. *Annals of Operations research*, 122(1-4), pp.21-42.

[3] Amaldi, E., Capone, A., Cesana, M., Filippini, I. and Malucelli, F., (2008). Optimization models and methods for planning wireless mesh networks. *Computer Networks*, 52(11), pp.2159-2171.

[4] Beasley D., Bull D.R. and Martin R.R., (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, 15(2), pp.58-69.

[5] Beasley, J.E., (1987). An algorithm for set covering problem. *European Journal of Operational Research*, 31(1), pp.85-93.

[6] Beasley, J.E. and Chu, P.C., (1996). A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2), pp.392-404.

[7] Benyamina, D., Hafid, A. and Gendreau, M., (2012). Wireless mesh networks design—A survey. *IEEE Communications surveys & tutorials*, 14(2), pp.299-310.

[8] Benyamina, D., Hafid, A., Hallam, N., Gendreau, M. and Maureira, J.C., (2012). A hybrid nature-inspired optimizer for wireless mesh networks design. *Computer Communications*, 35(10), pp.1231-1246.

[9] Boost.org. (2016). *Boost C++ Libraries*. [online] Available at: http://www.boost.org/ [Accessed 24 Jun. 2016].

[10] Boost.org. (2016). *Boost Graph Library: Boykov-Kolmogorov Maximum Flow - 1.60.0.* [online] Available at: http://www.boost.org/doc/libs/1_60_0/libs/graph/doc/boykov_kolmogorov_max_flow.html [Accessed 24 Jun. 2016].

[11] Brusco, M.J., Jacobs, L.W. and Thompson, G.M., (1999). A morphing procedure to supplement a simulated annealing heuristic for cost-and-coverage correlated set-covering problems. *Annals of Operations Research*, 86, pp.611-627.

[12] Caprara, A., Toth, P. and Fischetti, M., (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98(1-4), pp.353-371.

[13] Caserta, M., (2007). Tabu search-based metaheuristic algorithm for large-scale set covering problems. In *Metaheuristics*. Springer US, pp. 43-63

[14] Cormen, T., Leiserson, C., Rivest, R. and Stein, C. (2009). *Introduction to algorithms*. 3rd ed. Cambridge: Mit Press, pp.714-732.

[15] Correa, E.S., Steiner, M.T.A., Freitas, A.A. and Carnieri, C., (2004). A genetic algorithm for solving a capacitated p-median problem. *Numerical Algorithms*, 35(2-4), pp.373-388.

[16] Crawford, B., Soto, R., Cuesta, R. and Paredes, F., (2014). Using the bee colony optimization method to solve the weighted set covering problem. In *International Conference on Human-Computer Interaction*. Springer International Publishing, pp. 493-497.

[17] Dražic, Z., (2012). Variable Neighborhood Search for the File Transfer Scheduling Problem. *Serdica Journal of Computing*, 6(3), pp.333-348.

[18] Đenić, A., Marić, M., Stanimirović, Z. and Stanojević, P. (2016). A variable neighbourhood search method for solving the long-term care facility location problem. *IMA Journal of Management Mathematics*.

[19] Filipovic, V., (2003). Fine-grained Tournament Selection Operator in Genetic Algorithms. *Computing and Informatics*, 22(2), pp.143-161.

[20] Garey, M. and Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: Freeman.

[21] Goldberg D.E., (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Massachusetts: Addison-Wesley Publishing Company.

[22] Hansen, P., and Mladenović, N. (1997). Variable neighborhood search for the p-median. *Location Science*, 5(4), 207-226.

[23] Hansen, P. and Mladenovic, N., (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3), pp.449-467.

[24] Hansen, P. and Mladenovic, N. (2003). Variable Neighborhood Search. In: F. Glover and G. Kochenberger, ed., *Handbook of Metaheuristics*, Springer US, pp.145-184.

[25] Hansen, P., Mladenović, N., and Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1), 367-407

[26] Hansen, P. and Mladenovic, N. (2014). Variable Neighborhood Search. In: E. Burke and G. Kendall, ed., *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, 2nd ed. Springer US, pp.313-327.

[27] Holland J.H., (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

[28] Hoos, H.H. and Stützle, T., (2005). *Stochastic local search: Foundations & applications*. Elsevier, pp. 63-64.

[29] Ilić, A., Urošević, D., Brimberg, J., and Mladenović, N. (2010). A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. *European Journal of Operational Research*, 206(2), 289-300.

[30] Kimbrough, S.O., Koehler, G.J., Lu, M. and Wood, D.H., (2008). On a Feasible–Infeasible Two-Population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research*, 190(2), pp.310-327.

[31] Kleinberg, J. and Tardos, E. (2006). *Algorithm design*. Boston: Pearson/Addison-Wesley, pp.378-384.

[32] Lan, G., DePuy, G.W. and Whitehouse, G.E., (2007). An effective and simple heuristic for the set covering problem. *European journal of operational research*, 176(3), pp.1387-1403.

[33] Mladenovic, N. and Hansen, P., (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11), pp.1097-1100.

[34] Mladenović, N., Labbé, M., and Hansen, P. (2003). Solving the p - Center problem with Tabu Search and Variable Neighborhood Search. *Networks*, 42(1), 48-64.

[35] Mišković, S., Stanimirović, Z. and Grujičić, I., (2015). An efficient variable neighborhood search for solving a robust dynamic facility location problem in emergency service network. *Electronic Notes in Discrete Mathematics*, 47, pp.261-268.

[36] Nawaf, L., Mumford, C. and Allen, S., (2015), September. Optimizing the Placement of ITAPs in Wireless Mesh Networks by Implementing HC and SA Algorithms. In *International Conference on Ad Hoc Networks*. Springer International Publishing, pp. 29-41.

[37] Reeves C. (1993). Genetic Algorithms. In: C. Reeves, ed., *Modern Heuristic Techniques for Combinatorial Problems*. New York: John Willy and Sons, pp.151-196

[38] Reeves, C. (2003). Genetic Algorithms. In: F. Glover and G. Kochenberger, ed., *Handbook of Metaheuristics*, Springer US, pp.55-75.

[39] Ren, Z.G., Feng, Z.R., Ke, L.J. and Zhang, Z.J., (2010). New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58(4), pp.774-784.

[40] Sakamoto, S., Kulla, E., Oda, T., Ikeda, M., Barolli, L. and Xhafa, F., (2014). Performance evaluation considering iterations per phase and SA temperature in WMN-SA system. *Mobile Information Systems*, 10(3), pp.321-330.

[41] Schroeder, J., Guedes, A.L.P. and Duarte Jr, E.P., (2004). *Computing the minimum cut and maximum flow of undirected graphs.* Universidade Federal do Paraná.

[42] Stanimirović, Z., (2010). A genetic algorithm approach for the capacitated single allocation p-hub median problem. *Computing and Informatics*, 29(1), pp.117-132.

[43] Stanimirović, Z., Kratica, J., Filipović, V. and Tošić, D. (2011). *Evolutivni pristup za rešavanje hab lokacijskih problema*. Beograd: Zavod za udžbenike i nastavna sredstva, pp.25-38.

[44] Stanimirović, Z., Marić, M., Božović, S. and Stanojević, P. (2012). An efficient evolutionary algorithm for locating long-term care facilities. *Information Technology and Control*, 41(1), pp.77-89.

[45] Topcuoglu, H., Corut, F., Ermis, M., and Yilmaz, G. (2005). Solving the uncapacitated hub location problem using genetic algorithms. *Computers & Operations Research*, 32(4), 967-984.

[46] Xhafa, F., Sánchez, C. and Barolli, L., (2010), April. Genetic algorithms for efficient placement of router nodes in wireless mesh networks. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, pp. 465-472.

[47] Xhafa, F., Sánchez, C., Barolli, A. and Takizawa, M., (2015). Solving mesh router nodes placement problem in Wireless Mesh Networks by Tabu Search algorithm. *Journal of Computer and System Sciences*, 81(8), pp.1417-1428

[48] Yeniay, Ö., (2005). Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10(1), pp.45-56.