UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Staša Rašić

Rešavanje lokacijskog problema sa ograničenjem maksimalne udaljenosti

master rad

Beograd, 2025.

Mentor:

dr Stefan Mišković, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Filip Marić, redovan profesor
Univerzitet u Beogradu, Matematički fakultet

Vladimir Kuzmanović, asistent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: 8. 4. 2025.

Porodici i prijateljima

Naslov master rada: Rešavanje lokacijskog problema sa ograničenjem maksimalne udaljenosti

Rezime: Opšti pristup rešavanja lokacijskih problema se odnosi na odabir lokacija na kojima će biti uspostavljeni resursi i na alokaciju korisnika tj. na određivanje koji resursi će kojim korisnicima da isporučuju neke proizvode ili pružaju neku vrstu servisa. Rešavanje lokacijskih problema podrazumeva da ukupna suma alokacija korisnika i suma cena uspostavljanja resursa bude minimalna. Takođe ako korisnici imaju neka potraživanja od resursa ta potraživanja moraju biti u potpunosti ispunjena tj. svaki korisnik treba da dobije celokupnu količinu proizvoda ili bilo kakvih servisa koji su mu potrebni od svih resursa kojima je dodeljen. Pri tome ukupni zahtevi korisnika ne bi trebali da premaše kapacitete resursa kojima su dodeljeni tj. resursi treba da imaju jednake ili veće kapacitete nego što su ukupna potraživanja svih korisnika koji su alocirani da potražuju nešto od tih resursa. Rešavanje lokacijskih problema ima veoma praktičnu primenu. Ako su bolnice i ambulante resursi tada su korisnici stanovnici i veoma je bitno da mogu što pre da dođu do tih bolnica i ambulanti. Kada su banke resursi tada su vlasnici računa u tim bankama korisnici. Stanice na kojima se električni automobili pune strujom moroju biti locirane u skladu sa autonomijom vožnje električnih automobila sa napunjenom baterijom tj. u skladu sa kapacitetom baterija koje električni automobili koriste. Ako su korisnici supermarketi onda resursi mogu biti skladišta robe koja snabdevaju proizvodima te supermarkete. Ako su resursi vatrogasne stanice tada su korisnici svi objekti i površine gde može doći do požara. Kod ovog slučaja specijalno dolazi do izražaja lokacijski problem sa ograničenjem maksimalne udaljenosti jer vatrogasci moraju što pre da stignu na mesto gde je izbio požar i da ga ugase a za to je najbitniji faktor udaljenost između korisnika i resursa tj. lokacije na kojoj može da izbije požar i vatrogasne stanice. U ovom radu razmatran je lokacijski problem sa ograničenjem maksimalne udaljenosti (eng. Facility location problem with maximum distance constraint FLP-MD). Kod FLP-MD je cilj da ukupna suma cena uspostavljanja uspostavjenih resursa bude minimalna uz uslov da udaljenost između korisnika i resursa kome je dodeljen ne sme biti veća od unapred definisane udaljenosti. Egzaktne metode, kada su instance malih dimenzija, rešavaju lokacijske probleme tačno i brzo ali za veće instance može se desiti da zahtevaju jako veliku količinu vremena i zbog toga njihova upotreba, u slučaju velikih instanci, nije praktična. Zbog toga je za rešavanje problema korišćen metaheuristički pristup. Razvijen je algoritam koji implementira metodu promenljivih okolina (eng. Variable neighborhood search – VNS), tačnije varijantu VNS-a koja se zove redukovana metoda promenljivih okolina (eng. Reduced variable neighborhood search – RVNS). Za testiranje implementacije RVNS-a korišćene su ORLIB instance. Rezultati testiranja RVNS-a i rezultati CPLEX-a prikazani su i upoređeni. RVNS se po tačnosti a naročito po brzini izračunavanja rešenja pokazao kao efikasna metoda za rešavanje FLP-MD.

Ključne reči: lokacijski problemi, maksimalna udaljenost, optimizacioni problemi, metoda promenljivih okolina, metaheurističke metode.

SADRŽAJ

1. Uvod

1.1 Optimizacioni problemi

Optimizacija predstavlja proces dobijanja najboljeg rešenja razmatranog problema, imajući u vidu odgovarajuća ograničenja [2]. Neka je dat skup S i funkcija $f : S \to R$. Ako postoje neka specifična ograničenja za elemente skupa S, tada elementi koji zadovoljavaju ta ograničenja čine skup $X \subseteq S$. Cilj optimizacije je pronaći $x_0 \in X$ za koje važi

$$f(x_0) = \min_{x \in X} f(x) \qquad (1)$$

Skup S se često naziva pretraživački prostor. Elementi skupa X se nazivaju dopustivim rešenjima problema (1), a skup X dopustiv skup. Funkcija f se naziva funkcija cilja. Element x_0 u kome funkcija cilja dostiže minimum se naziva optimalno rešenje. Funkcija cilja može u više elemenata skupa X dostizati minimum, zbog čega može postojati i više optimalnih rešenja. Najčešće je dovoljno pronaći bar neko optimalno rešenje. U zavisnosti od problema koji se optimizuje, definisana su i ograničenja za elemente skupa S, kao i to da li se funkcija cilja minimizuje ili maksimizuje. Ako se funkcija cilja minimizuje, tada se radi o problemu minimizacije, a ako se funkcija cilja maksimizuje, radi se o problemu maksimizacije. Ako se radi o problemu maksimizacije, skupovi S, X i element x_0 su definisani na isti način kao i kod problema minimizacije, tako da je sve isto kao i kod problema minimizacije, osim toga što se ciljna funkcija maksimizuje. U tom slučaju se traže elementi $x \in \mathrm{X}$ za koji je

$$f(x_0) = \max_{x \in X} f(x)$$

Pošto važi jednakost

$$\max_{x \in X} f(x) = -\min_{x \in X}(-f(x))$$

problem maksimizacije se može svesti na problem minimizacije.

Optimizacioni problemi se mogu klasifikovati korišćenjem različitih kriterijuma. Neki od načina njihove klasifikacije su:

• Na osnovu toga da li postoje ograničenja za elemente skupa S, optimizacioni problemi se dele na probleme uslovne optimizacije, gde su prisutna ograničenja i tada je $X \neq S$, i na probleme bezuslovne optimizacije, gde elementi skupa S nisu međusobno uslovljeni odgovarajućim ograničenjima i tada je $X = S$.

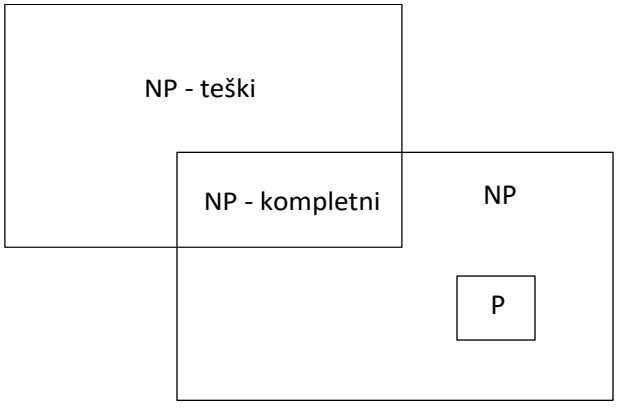
• Na osnovu toga da li su parametri u optimizacionom problemu poznati, slučajnog karaktera ili neodređeni, optimizacioni problemi se dele na determinističke, u sličaju kada su u optimizacionom problemu parametri poznati, zadatke stohastičke optimizacije, kada su parametri u optimizacionom problemu slučajnog karaktera i zadatke optimizacije u uslovima neodređenosti, kada su parametri u optimizacionom problemu neodređeni [3].

• Prema tipu skupa S, problemi optimizacije se mogu podeliti na diskretne (kombinatorne) probleme optimizacije kada je skup S konačan ili prebrojivo beskonačan, kontinualne probleme optimizacije, kada je skup S neprebrojiv i mešovite probleme optimizacije, koji sadrže elemente i diskretne i kontinualne optimizacije.

Primene optimizacije su izuzetno široke i pokrivaju različite domene. U industriji i proizvodnji, optimizacija je ključna za unapređenje efikasnosti i smanjenje troškova. Efikasno upravljanje zalihama omogućava održavanje njihovog optimalnog nivoa, kako bi se izbegao višak ili manjak materijala. Planiranje proizvodnje i logistike pomaže u preciznom usklađivanju proizvodnih kapaciteta s potražnjom, što doprinosi smanjenju zastoja i povećanju produktivnosti. Raspored radnih smena optimizuje raspodelu radnih sati među zaposlenima, čime se poboljšava iskorišćenost radne snage i resursa. Planiranje kapaciteta, s druge strane, omogućava prepoznavanje i rešavanje potencijalnih uskih grla u proizvodnom procesu, što vodi ka smanjenju troškova i poboljšanju ukupne efikasnosti. Više o primeni otimizacije u industriji može se videti u radu [16]. U finansijskom sektoru, optimizacija igra značajnu ulogu u upravljanju portfeljem investicija. Kroz različite modele optimizacije, investitori mogu maksimizovati profit svojih investicija uz minimizaciju rizika, što je ključno za ostvarivanje dugoročnih finansijskih ciljeva. Pored toga, upravljanje rizikom koristi optimizaciju za analizu potencijalnih finansijskih rizika i primenu strategija koje minimizuju te rizike. Ova praksa pomaže u očuvanju kapitala i stabilnosti finansijskog portfolija [17]. U logistici i transportu, optimizacija rute isporuke je ključna za smanjenje troškova goriva i vremena potrebnog za isporuku. Pomoću sofisticiranih algoritama i modela, moguće je odabrati najefikasnije rute za vozila, što doprinosi smanjenju operativnih troškova i poboljšanju usluga korisnicima. Takođe, upravljanje skladištima koristi optimizaciju za efikasno korišćenje skladišnog prostora i resursa, čime se smanjuje potreba za dodatnim prostorom i poboljšava brzina obrade narudžbina. U oblasti telekomunikacija, optimizacija mreže omogućava efikasno upravljanje mrežnim resursima, što doprinosi poboljšanju kvaliteta usluge i smanjenju operativnih troškova. Ova praksa uključuje analizu i prilagođavanje raspodele resursa kao što su kapaciteti mreže. Takođe, planiranje kapaciteta mreže pomaže u obezbeđivanju da mrežni sistemi mogu da ispune potrebe korisnika i podrže rastuću potražnju za uslugama. U zdravstvu,

optimizacija resursa igra ključnu ulogu u unapređenju efikasnosti pružanja zdravstvenih usluga. Ovo uključuje optimizaciju raspodele medicinskih aparata i osoblja, što pomaže u obezbeđivanju da svi potrebni resursi budu dostupni kada su najpotrebniji. Takođe, optimizacija procesa prijema i čekanja pacijenata doprinosi smanjenju vremena čekanja i poboljšanju ukupnog iskustva pacijenata. U sektoru energetike, optimizacija upravljanja potrošnjom energije pomaže u smanjenju troškova i povećanju efikasnosti kroz optimizaciju potrošnje u zgradama i industrijskim procesima. Takođe, planiranje proizvodnje energije iz različitih izvora omogućava ravnotežu između proizvodnje i potražnje, što doprinosi stabilnosti i efikasnosti energetske mreže. U računarstvu, optimizacija algoritama poboljšava efikasnost u smislu brzine obrade podataka i korišćenja memorije. Efikasni algoritmi mogu značajno ubrzati procese i smanjiti potrebne resurse. Takođe, upravljanje resursima u operativnim sistemima koristi optimizaciju za efikasno raspoređivanje procesorskog vremena i memorije, što poboljšava performanse i stabilnost sistema. U marketingu i prodaji, optimizacija segmentacije tržišta omogućava prepoznavanje i ciljanje specifičnih segmenata potrošača, što poboljšava efikasnost marketinških kampanja i povećava njihovu uspešnost. Takođe, optimizacija cena proizvoda pomaže u određivanju najpovoljnijih cena koje maksimizuju profit, uzimajući u obzir različite tržišne uslove i reakcije potrošača. Sve ove primene optimizacije imaju zajednički cilj: da poboljšaju efikasnost, smanje troškove i maksimizuju rezultate u svojim respektivnim oblastima.

Optimizacioni problem pripada P klasi ako postoji algoritam polinomske složenosti sa kojim taj problem može da se reši. Problem pripada NP klasi ako se u polinomskom vremenu može odrediti da li je neka vrednost rešenje tog problema. Problem je NP-težak ukoliko se na njega može svesti svaki problem iz NP klase. Problem je NP-kompletan ako je istovremeno NP-težak i pripada NP klasi [5, 6]. P klasa je podskup NP klase, ali nije poznato da li su P klasa i NP klasa jednake. Odnos klasa složenosti P i NP, kao i grupa NP-teških i NP-kompletnih problema, prikazan je na slici 1.



Slika 1. Prikaz klasa složenosti

1.2 Lokacijski problemi

Lokacijski problemi čine posebnu grupu problema kod kojih se određuje lokacija različitih tipova objekata [7, 8, 9, 10]. Objekti koji se lociraju najčešće se nazivaju resursi, a objekti ili osobe kojima oni pružaju neku vrstu usluge, servisa ili isporučuju neke proizvode najčešće se nazivaju korisnici. Prilikom rešavanja lokacijskih problema, bitno je odrediti koliko resursa i na kojim lokacijama ih treba uspostaviti. Takođe je bitno odrediti koji resursi će kojim korisnicima pružati usluge. Tada se govori o alokaciji korisnika. U zavisnosti od vrste problema postoje i različiti uslovi koji moraju biti zadovoljeni. Cena alokacije je najčešće srazmerna rastojanju korisnika od resursa ili troškovima transporta nekih proizvoda od resursa do korisnika. Postoje razni kriterijumi prema kojim se lokacijski problemi mogu klasifikovati. Neki od načina podele su sledeći:

• Podela lokacijskih problema na statičke i dinamičke. Kod statičkih lokacijskih problema ulazni podaci se ne menjaju tokom vremena, dok se kod dinamičkih lokacijskih problema tokom vremena ulazni podaci menjaju. Kako se tokom vremena kod dinamičkih lokacijskih problema ulazni podaci menjaju, tako se uspostavljeni resursi mogu uklanjati ili premeštati i mogu se uspostavljati novi resursi.

• Podela na endogene i egzogene lokacijske probleme. Ako je broj resursa koje treba uspostaviti unapred poznat, tada se radi o endogenim lokacijskim problemima, a ako nije unapred poznat, tada se radi o egzogenim lokacijskim problemima. Broj resursa koje treba uspostaviti se u tom slučaju dobija kao rezultat optimizacije. Dodatno, u zavisnosti od traženog broja resursa koji se uspostavljaju, Endogeni lokacijski problemi se mogu podeliti na jednoobjektne i višeobjektne. Kada je pored određivanja lokacija na kojima je potrebno uspostaviti resurse potrebno i alocirati korisnike, tj. odrediti i koji resursi će pružati usluge kojim korisnicima, tada se radi o lokacijsko-alokacijskom problemu.

• Podela na min-sum i min-max lokacijske probleme. Kod min-sum lokacijskih problema funkcija cilja minimizuje ukupnu sumu rastojanja od korisnika do resursa, dok kod min-max lokacijskih problema funkcija cilja minimizuje maksimalno rastojanje od korisnika do resursa. Ako je kod min-sum problema potrebno uspostaviti tačno p resursa, tada se oni nazivaju problemi p-medijane, a kada je kod min-max problema potrebno uspostaviti tačno p resursa, oni se nazivaju problemi p-centra.

• Podeala na diskretne, mrežne i kontinualne lokacijske probleme. Ova podela je izvedena na osnovu mesta na kojima je dozvoljeno uspostaviti resurse. Ako se lokacije za uspostavljanje resursa mogu birati samo iz unapred određenog konačnog skupa lokacija, radi se o diskretnim lokacijskim problemima. Ako je resurse dozvoljeno locirati bilo gde u posmatranoj oblasti, tada

su to kontinualni lokacijski problemi. Ako se resursi uspostavljaju na zadatoj meži tada su to mrežni lokacijski problemi.

• Podela na lokacijske probleme ograničenih i neograničenih kapaciteta. Ako resursi imaju ograničenja kapaciteta za usluge koje pružaju korisnicima, radi se o lokacijskim problemima ograničenih kapaciteta, a ako resursi nemaju takva ograničenja, reč je o lokacijskim problemima neograničenih kapaciteta.

1.3. Prost lokacijski problem neograničenih kapaciteta

Kod prostog lokacijskog problema neograničenih kapaciteta (eng. Uncapacitated facility location problem – UFLP) definisan je skup korisnika *D* i skup resursa F. Za svaki par korisnika i resursa poznata je cena dodeljivanja c_{ij}, $i \in$ F, $j \in$ D, a za svaki resurs njegova cena uspostavljanja f_i, $i \in$ F. Svakog korisnika je potrebno alocirati, odnosno dodeliti tačno jednom resursu, tako da ukupna suma cena alokacija korisnika i cena uspostavljanja resursa bude minimalna. Pri dodeljivanju neki resursi ne moraju biti uspostavljeni, ali svaki korisnik mora biti dodeljen tačno jednom resursu. Neka su x_{ij}, $i \in$ F , $j \in$ D binarne promenljive koje uzimaju vrednost 1 ukoliko je korisnik j dodeljen resursu i , a inače 0, a y_i, $i \in$ F binarne promenljive koje uzimaju vrednost 1 ukoliko je resurs i uspostavljen, a inače 0. Imajući prethodno u vidu, matematički model za UFLP se može formulisati kao

$$\min \sum_{i \in F} \sum_{j \in D} x_{ij}\, c_{ij} + \sum_{i \in F} f_i\, y_i \qquad (2)$$

uz sledeća ograničenja:

$$\sum_{i \in F} x_{ij} = 1, \qquad \forall j \in D\ , \qquad (3)$$

$$x_{ij} \le y_i\ , \qquad \forall i \in F\ ,\ \forall j \in D\ , \qquad (4)$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in F\ ,\ \forall j \in D\ , \qquad (5)$$

$$y_i \in \{0,1\}, \qquad \forall i \in F\ . \qquad (6)$$

Funkcija cilja (2) minimizuje ukupnu sumu cena alokacija korisnika i cena uspostavljanja resursa. Uslovima (3) je određeno da se svaki korisnik može dodeliti tačno jednom resursu. Uslovima (4) je obezbeđeno da se svaki korisnik može dodeliti nekom resursu samo ako je taj resurs uspostavljen. Uslovi (5) i (6) se odnose na to da su promenljive x_{ij} i y_i , $i \in F$, $j \in D$ binarne.

Jedan od najjednostavnijih načina za rešavanje prostog lokacijskog problema neograničenih kapaciteta predstavlja proveravanje svih kombinacija među uspostavljenim resursima, kojih ima $2^{|D|} - 1$. Ukoliko je odabran skup $P \subseteq F$ uspostavljenih resursa, za svakog korisnika $j \in D$ je dovoljno pronaći najmanju vrednost c_{ij} među svim $i \in P$. Time se jednostavno može pronaći rešenje potproblema sa uspostavljenim resursima, a optimalno rešenje problema je ono čija je ukupna vrednost cena, među svim podskupovima uspostavljenih resursa, najmanja. Složenost ovog algoritma je $O(2^{|D|}|D||P|)$.

Od šezdesetih godina do danas razvijeno je mnogo egzaktnih algoritama koji rešavaju UFLP [4]. UFLP je jedan od najčešće razmatranih NP-teških problema. U [11] i [12] se može pronaći pregled različitih pristupa za njegovo rešavanje. U literaturi su razmatrane i njegove različite varijante.

Kod prostog lokacijskog problema ograničenih kapaciteta sa jednostrukim alokacijama (eng. Single source capacitated facility location problem – SSCFLP), za razliku od UFLP, svaki Korisnik j ima svoja potraživanja d_j, a svaki resurs i ima kapacitet s_i. Svakog korisnika je potrebno dodeliti tačno jednom resursu, tako da ukupna suma cena dodeljivanja i cena uspostavljanja resursa bude minimalna. Suma potražnji svih korisnika koji su dodeljeni nekom resursu ne sme biti veća od kapaciteta tog resursa [13, 14].

Prost lokacijski problem neograničenih kapaciteta sa višestrukim alokacijama (eng. Multi-level uncapacitated facility location problem – MLUFLP). Kod MLUFLP je dozvoljeno da jedan korisnik bude dodeljen neograničenom broju resursa. Celokupno potraživanje korisnika treba da bude ostvareno iz svih resursa kojima je korisnik dodeljen [15].

U ovom radu će biti razmatran lokacijski problem sa ograničenjem maksimalne udaljenosti (eng. Facility location problem with maximum distance constraint – FLP-MD), koji je prvi put uveden u [2]. Kod njega se razmatra skup korisnika i potencijalnih lokacija za resurse kojima korisnici mogu biti dodeljeni. Kod FLP-MD potrebno je minimizovati samo ukupnu sumu cena uspostavljanja resursa, za razliku od UFLP, gde se minimizuje ukupna suma cena alokacija korisnika i cena uspostavljanja resursa. Kod FLP-MD postoji dodatno ograničenje maksimalnog rastojanja između korisnika i resursa tako da korisnik može biti dodeljen nekom resursu samo ako udaljenost između tog korisnika i tog resursa nije veća od ograničenja za maksimalno rastojanje između korisnika i resursa. Kod FLP-MD korisnici nemaju nikakva potraživanja od resursa koja moraju biti zadovoljena kao što je slučaj kod SSCFLP i MLUFLP. Uvođenje maksimalnog dozvoljenog rastojanja ima praktični značaj. Na primer, potencijalna lokacija za bolnicu ili policijsku stanicu treba da bude blizu svim građanima tj. ne bi trebala da se nalazi na velikoj udaljenosti od bilo kog građanina. Još evidentniji primer je odabir lokacija za vatrogasne stanice. Vatrogasna stanica treba da bude postavljena na lokaciji od koje može brzo da se stigne do svih mesta gde može da izbije požar a naročito do mesta gde ima živih bića.

U sekciji 2 je detaljnije opisana matematička formulacija FLP-MD. Predložena metoda promenljivih okolina za njegovo rešavanje je predstavljena u sekciji 3. Dobijeni rezultati predloženog algoritma i njegovo poređene sa CPLEX rešavačem je prikazano u sekciji 4. U sekciji 5. je definisan algoritam za redukciju dimenzija instanci za FLP-MD i prikazane su brzine

izračunavanja CPLEX-a na originalnim instancama i na redukovanim instancama koje su dobijene primenom algoritma za redukciju dimenzija instanci kod FLP-MD. Konačno, u sekciji 6. je dat kratak zaključak sa predlozima za budući rad.

2. Lokacijski problem sa ograničenjem maksimalne udaljenosti

2.1 Opis problema

U nastavku će detaljnije biti opisan lokacijski problem sa ograničenjem maksimalne udaljenosti FLP-MD. Razlika između FLP-MD i UFLP je u tome što se kod UFLP minimizuje ukupna suma cena alokacija korisnika i cena uspostavljanja resura, a kod FLP-MD se minimizuje samo ukupna suma cena uspostavljanja resursa. Dodatno, kod FLP-MD postoji uslov da, udaljenost korisnika od resursa ne bude veća od neke unapred određene maksimalne udaljenosti.
Kod FLP-MD, postoji skup potencijalnih lokacija za uspostavljanje resursa (npr. skladišta, servisnih centara, bolnica, policijskih stanica, vatrogasnih stanica, supermarketa, banaka itd.) i skup korisnika (npr. kupaca, pacijenata, korisnika nekih usluga itd.). Cilj je da se odredi koje lokacije treba odabrati za uspostavljanje resursa, tako da se minimizuje ukupna cena njihovog uspostavljanja, uz uslov da udaljenost nijednog korisnika ne prelazi unapred definisanu maksimalnu udaljenost do resursa kome je dodeljen. FLP-MD je koristan u različitim scenarijima, gde je važno, ne samo optimizovati troškove uspostavljanja resursa, već i obezbediti da rastojanje između korisnika i resursa ne bude preveliko. To je posebno važno u planiranju infrastrukture kao što su bolnice, vatrogasne stanice, policijske stanice, pošte, banke, supermarketi ili bilo kojih sistema gde je bitnija pristupačnost resursima od troškova uspostavljanja resursa. Razmatrani problem može biti koristan i pri postavljanju centara za testiranje stanovništva u pandemijskim uslovima. Potrebno je uspostaviti odgovarajući broj stanica za testiranje stanovništva ali tako da svi stanovnici moraju biti u mogućnosti da stignu do najbliže stanice za testiranje u kratkom vremenskom periodu. Zbog toga je potrebno odrediti maksimalnu udaljenost između izabranih lokacija za postavljanje centara za testiranje i stanovnika. Pošto cene postavljanja centara za testiranje na različitim lokacijama variraju, neophodan je optimalan odabir lokacija za njihovo postavljanje u cilju minimizacije ukupnih troškova.

2.2 Matematička formulacija

Lokacijski problem sa ograničenjem maksimalne udaljenosti je predložen u [2].
Neka su dati:

- F - skup potencijalnih lokacija za resurse.
- D - skup korisnika.

- d_{ij} - Udaljenost resursa i od korisnika j.
- y_i - Promenljiva koja će imati vrednost 1 ako je resurs i otvoren ili vrednost 0 ako nije otvoren.
- x_{ij} - Promenljiva koja će imati vrednost 1 ako je kosrisnik j dodeljen resursu i ili 0 ako nije.
- L - Pozitivan broj koji predstavlja ograničenje za maksimalnu udaljenost korisnika od resursa.

Imajući prethodne oznake u vidu, FLP-MD se može formulisati na sledeći način:

$$\min \sum_{i \in F} f_i \, y_i \ , \tag{7}$$

uz sledeće uslove:

$$\sum_{i \in F} x_{ij} = 1 \ , \qquad \forall j \in D \ , \tag{8}$$

$$\sum_{i \in F} x_{ij} \, c_{ij} \leq L \ , \qquad \forall j \in D \ , \tag{9}$$

$$x_{ij} \ \leq \ y_i \ , \qquad \forall i \in \ F, \ \forall j \in D \ , \tag{10}$$

$$x_{ij} \ \in \ \{0,1\}, \qquad \forall i \in \ F, \ \forall j \in D \ , \tag{11}$$

$$y_i \ \in \ \{0,1\}, \qquad \forall i \in D \ . \tag{12}$$

Funkcija cilja (7) minimizuje ukupnu sumu cena uspostavljanja resursa. Uslovima (8) je određeno da se svaki korisnik može dodeliti tačno jednom resursu. Uslovima (9) obezbeđuje se da udaljenost korisnika od resursa nije veća od unapred određene maksimalne udaljenosti. Uslovima (10) je obezbeđeno da se svaki korisnik može dodeliti nekom resursu samo ako je taj resurs uspostavljen. Uslovi (11) i (12) se odnose na binarnu prirodu promenljivih $\boldsymbol{x_{ij}}$ i $\boldsymbol{y_i}$, $i \in \boldsymbol{F}$, $j \in \boldsymbol{D}$.

2.3 Primer problema

U sledećem primeru data je matrica M dimenzije 5×4 koja sadrži udaljenosti korisnika od resursa, gde svaka vrsta predstavlja korisnika a svaka kolona resurs. Polje M_{ij} gde je i index vrste a j index kolone, predstavlja udaljenost i-tog korisnika od j-tog resursa. Dat je niz f dimenzije 4 koji sadrži cene uspostavljanja resursa. U primeru će biti razmotreno nekoliko različitih vrednosti maksimalnog ograničenja udaljenosti korisnika od resursa. Za svaku datu vrednost parametra L će biti prikazana matrica koja se sastoji od 0 i 1, tako da na mestima gde je 1, udaljenost koisnika od

resursa nije veća od L, a na mestima gde je 0, udaljenost korisnika od resursa je veća od L. Takođe, biće prikazana vrednost rešenja koja predstavlja minimalnu sumu cena uspostavljanja neophodnih resursa tako da svaki korisnik bude dodeljen jednom resursu od koga se nalazi na udaljenosti manjoj ili jednakoj od L. Biće prikazan i niz y koji se sastoji od elemenata koji imaju vrednost 1 ili 0 koji prikazuje koji su resursi uspostavljeni, tako što će na pozicijama gde je resurs uspostavljen biti vrednost 1, a 0 inače. Rešenje će sadržati i matricu X koja će na mestima gde je korisnik alociran imati vrednost 1 a na ostalim mestima vrednost 0.

Neka je M matrica udaljenosti korisnika od resursa

$$M = \begin{bmatrix} 7 & 2 & 8 & 9 \\ 1 & 3 & 3 & 4 \\ 2 & 2 & 4 & 3 \\ 8 & 7 & 7 & 1 \\ 1 & 2 & 7 & 2 \end{bmatrix}$$

Neka su cene uspostavljanja resursa $f = [3, \quad 5, \quad 4, \quad 7]$

• Prvo ćemo postaviti ograničenje za maksimalnu udaljenost korisnika od resursa koje je jednako najvećoj udaljenosti u matrici M, a to je 9. Za takvu vrednost $L = 9$, razmatrani problem se svodi na UFLP, jer nijedna udaljenost između bilo kog korisinka i bilo kog resursa nije veća od ograničenja za maksimalnu udaljenost i svaki korisnik može biti dodeljen svakom resursu. Matrica koja se sastoji od elemenata koji imaju vrednost 0 ili 1 koja prikazuje gde su uslovi maksimalne udaljenosti korisnika od resusa zadovoljeni izgleda

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Pošto svaki korisnik može biti dodeljen svakom resursu, minimalna suma cena uspostavljanja neophodnih resursa, tako da svaki korisnik bude alociran, se dobija kada se svi korisnici dodele resursu čija je cena uspostavljanja najjeftinija. Tako da se rešenje dobija kada se svi korisnici dodele resursu sa cenom uspostavljanja 3.

Prikaz uspostavljenih resursa $y = [1, \quad 0, \quad 0, \quad 0]$

Prikaz alokacija korisnika

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Pošto je uspostavljen samo resurs čija je cena uspostavljanja 3, rešenje je 3.

• Ako je L = 5, matrica koja se sastoji od elemenata koji imaju vrednost 0 ili 1 koja prikazuje gde udaljenosti korisnika od resursa nisu veće od L izgleda

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Prvog korisnika možemo dodeliti samo drugom resursu, zato što se samo od njega ne nalazi na udaljenosti većoj od L. Zbog toga je neophodno uspostaviti drugi resurs čija je cena uspostavljanja 7. Drugi i treći korisnik mogu biti dodeljeni bilo kom resursu jer se od svih resursa nalaze na udaljenosti manjoj ili jednakoj od $\mathrm{L} = 5$, ali pošto je drugi resurs već uspostavljen, drugi i treći korisnik će biti dodeljeni drugom resursu, zato što to neće povećati ukupne troškove uspostavljanja resursa. Četvrti korisnik se nalazi na udaljenosti manjoj ili jednakoj od $L = 5$ samo od četvrtog resursa, pa može biti dodeljen samo četvrtom resursu čija je cena uspostavljanja 9. Zbog toga će četvrti resurs biti uspostavljen. Peti korisnik može biti dodeljen prvom, drugom i četvrtom resursu. Prvi resurs ima najmanju cenu uspostavljanja, ali pošto bi dodeljivanje petog korisnika prvom resursu izazvalo uspostavljanje prvog resursa koji do sada nije uspostavljen, to bi povećalo ukupne troškove uspostavljanja resursa, pa će peti korisnik biti dodeljen ili drugom ili četvrtom resursu, bez obzira što je cena njihovog uspostavljanja veća od cene uspostavljanja prvog resursa. Pošto su drugi i četvrti resurs već uspostavljeni i dodela petog korisnika nekom od ta dva resursa neće povećeti ukupnu cenu otvaranja neophodnih resursa. Rešenje će biti zbir cena uspostavljanja drugog i četvrtog resursa što iznosi 12.

Prikaz uspostavljenih resursa

$$y = [0, \quad 1, \quad 0, \quad 1]$$

Prikaz alokacija korisnika

$$X = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Dakle, rešenje za $L = 5$ iznosi 12.

• Neka je ograničenje maksimalne udaljenosti korisnika od resursa sada $L = 2$. Matrica koja prikazuje gde udaljenosti korisnika od resusa nisu veće od L izgleda:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Prvog korisnika možemo dodeliti samo drugom resursu zato što se samo od drugog resursa nalazi na udaljenosti manjoj ili jednakoj od L. Zbog toga je neophodno uspostaviti drugi resurs čija je cena uspostavljanja 7. Drugog korisnika možemo dodeliti samo prvom resursu zato što se samo od prvog resursa nalazi na udaljenosti manjoj ili jednakoj od L. Zbog toga je neophodno uspostaviti prvi resurs čija je cena uspostavljanja 3. Trećeg korisnika možemo dodeliti prvom ili drugom resursu, jer se od oba nalazi na udaljenosti manjoj ili jednakoj od L i nije bitno kom resursu će biti dodeljen, budući da su oba resursa već uspostavljena. Četvrti korisnik se nalazi na udaljenosti manjoj ili jednakoj od L samo od četvrtog resursa, pa zato može biti dodeljen samo četvrtom resursu čija je cena uspostavljanja 9 i zbog toga se uspostavlja i četvrti resurs. Peti korisnik može biti dodeljen prvom, drugom i četvrtom resursu. Svejedno je kom resursu će biti dodeljen jer su i prvi i drugi i četvrti resurs već uspostavljeni i dodela bilo kom od ta tri resursa neće povećeti ukupnu cenu otvaranja neophodnih resursa. Tako da će rešenje za $L = 2$ biti zbir cena uspostavljanja prvog, drugog i četvrtog resursa što iznosi 15.

Prikaz uspostavljenih resursa

$$y = [1, \quad 1, \quad 0, \quad 1]$$

Prikaz alokacija korisnika

$$X = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

• Za ograničenje maksimalne udaljenosti korisnika od resursa $L = 1$, matrica koja prikazuje gde udaljenosti korisnika od resusa nisu veće od L izgleda

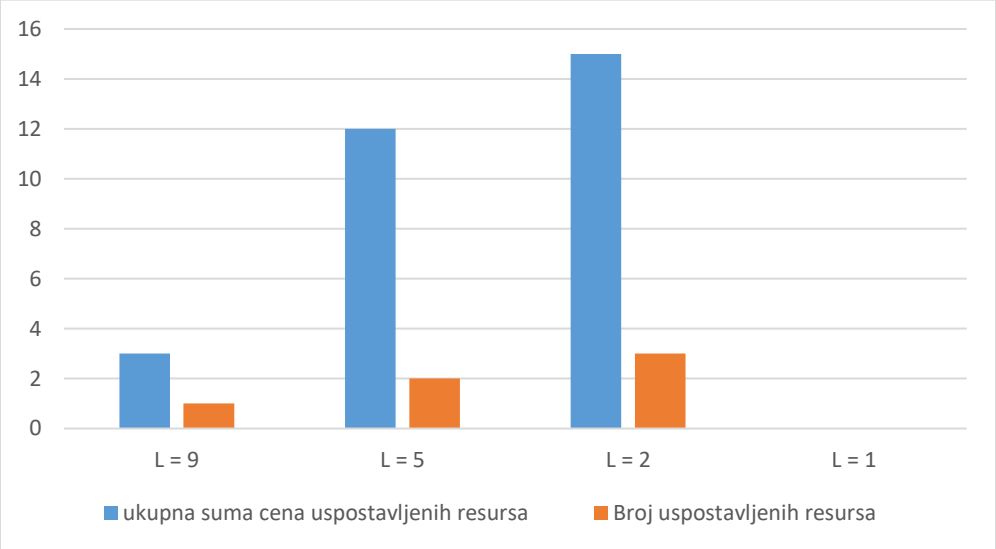
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Prvi i treći korisnik ne mogu biti dodeljeni ni jednom resursu, zato što se od svih resursa nalaze na udaljenosti većoj od L i zbog toga za $L = 1$ ne postoji rešenje.

Može se uočiti da se sa smanjenjem ograničenja za maksimalnu udaljenost korisnika od resursa, ukupna suma cena uspostavljanja resursa povećava, zato što se smanjuje broj mogućnosti dodeljivanja korisnika resursima, što izaziva potrebu za uspostavljanjem većeg broja resursa. Odnos između smanjenja ograničenja za maksimalnu udaljenost korisnika od resursa L, ukupne sume cena uspostavljanja neophodnih resursa i broja uspostavljenih resursa, za navedeni primer je prikazan u tabeli 1 i na slici 2.

L	Suma cena uspostavljenih resursa	Broj uspostavljenih resursa
9	3	1
5	12	2
2	15	3
1	Ne postoji rešenje	

Tabela 1. Odnos između ograničenja za maksimalnu udaljenost korisnika od resursa, ukupne sume cena uspostavljanja neophodnih resursa i broja uspostavljenih resursa



Slika 2. Odnos između ograničenja za maksimalnu udaljenost korisnika od resursa, ukupne sume cena uspostavljanja neophodnih resursa i broja uspostavljenih resursa

3. Algoritam za rešavanje problema

3.1 Metoda promenljivih okolina

Metoda promenljivih okolina (eng. Variable neighborhood search – VNS) je prvi put predložena u [1]. VNS se fokusira na iterativno unapređivanje rešenja x tako što kontinuirano pretražuje različite, unapred definisane okoline tog rešenja $U_l(x)$, $1 \leq l \leq l_{max}$ i na taj način pronalazi najbolje rešenje. Kako bi se sprečilo da algoritam zaglavi u lokalnom optimumu, definisane okoline koje se pretražuju se postupno menjaju. Metoda promenljivih okolina se fokusira na sledeće činjenice:

- Lokalni minimum jedne okoline ne mora da bude i lokalni minimum druge okoline.
- Globalni minimum je lokalni minimum svih okolina.
- Za mnoge probleme u praksi, lokalni minimumi su relativno blizu jedni drugih.

Postoji više verzija inicijalnog algoritma, od kojih se uglavnom pojavljuju redukovana metoda promenljivih okolina, osnovna metoda promenljivih okolina, uopštena metoda promenljivih okolina i metoda promenljivog spusta. Generalno kada se pronađe bolje rešenje, algoritam se vraća na prvu pretraživanu okolinu. Ako nije pronađeno bolje rešenje, pretraga se nastavlja u narednoj okolini. Redukovana metoda promenljivih okolina (Reduced variable neighborhood search – RVNS) je specifična po tome što se kod nje ne radi lokalna pretraga, već se uzima neko proizvoljno rešenje iz okoline trenutnog u cilju traženja boljeg rešenja. Ako se pronađe bolje rešenje iz okoline trenutnog rešenja, tada se to bolje rešenje postavi da bude trenutno najbolje rešenje i nastavlja se pretraživanje po polaznoj okolini. Ako se ne pronađe bolje rešenje od trenutnog prelazi se u sledeću okolinu, gde se broj resursa, čija stanja menja funkcija invert, povećava za jedan. Takođe se poveća za jedan broj parova resursa, koji imaju različita stanja, čija stanja funkcija swap promeni. Pseudokod za RVNS je prikazan sledećim algoritmom.

Algoritam: Redukovana metoda promenljivih okolina – RVNS

Definisati okoline U_l, $1 \leq l \leq l_{max}$, koje će se koristiti za pretragu
Generisati početno rešenje x
repeat
  $l \leftarrow 1$
  while $l \leq l_{max}$ **do**
    Izabrati proizvoljno rešenje x' u okolini $U_l(x)$,
    if rešenje x' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x'$

$l \leftarrow 1$
else
$l \leftarrow l + 1$
end if
end while
Ako je pronađeno bolje rešenje ažurirati vrednost najboljeg rešenja
until nije zadovoljen kriterijum zaustavljanja

Osnovna metoda promenljivih okolina (Basic variable neighborhood search – BVNS) je karakteristična po tome što se kod nje primenjuje neki tip lokalne pretrage na slučajno izabrano početno rešenje x iz tekuće okoline. Lokalna pretraga se primenjuje na slučajno izabrano rešenje iz tekuće okoline da bi se pronašlo bolje rešenje sa najmanjom vrednošću funkcije cilja. Ako se pronađe bolje rešenje sa manjom vrednošću funkcije cilja, tada to rešenje postaje trenutno najbolje rešenje i pretrga se nastavlja u polaznoj okolini. Ako se ne pronađe bolje rešenje, tada se broj resursa čija stanja menja funkcije invert i broj parova resursa sa različitim stanjima čija stanja menja funkcija swap povećava za jedan i tako se prelazi u sledeću okolinu po kojoj se vrši pretraga u cilju pronalaska boljeg rešenja, za koje funkcija cilja ima manju vrednost. Ako se u toj novoj okolini pronađe bolje rešenje, pretraga se opet nastavlja u polaznoj okolini, a ako se ne pronađe bolje rešenje prelazi se u narednu okolinu i tako sve do l_{max}. Pseudokod za BVNS je prikazan sledećim algoritmom.

Algoritam: Osnovna metoda promenljivih okolina – BVNS

Definisati okoline U_l, $1 \leq l \leq l_{max}$, koje će se koristiti za pretragu
Generisati početno rešenje x
repeat
$l \leftarrow 1$
while $l \leq l_{max}$, **do**
Izabrati proizvoljno rešenje x' u okolini $U_l(x)$
Primenom nekog tipa lokalne pretrage na početno rešenje x' naći rešenje x'' sa najmanjom vrednošću funkcije cilja.
if rešenje x'' ima manju vrednost funkcije cilja od rešenja x **then**
$x \leftarrow x''$,
$l \leftarrow 1$
else
$l \leftarrow l + 1$
end if
end while
Ako je pronađeno bolje rešenje ažurirati vrednost najboljeg rešenja
until nije zadovoljen kriterijum zaustavljanja

Uopštena metoda promenljivih okolina (eng. General variable neighborhood search – GVNS) primenjuje dve varijante lokalne pretrage, spoljašnju i unutrašnju varijantu lokalne pretrage. Pri tome se za unutrašnju varijantu lokalne pretrage koristi jedan tip okoline a za spoljašnju varijantu lokalne pretrage drugi tip okoline. Zbog toga imamo za svaki tip okolina maksimalan broj okolina l_{max} i l'_{max} i jedan tip okoline ide do l_{max} a drugi do l'_{max}. Pseudokod za GVNS je prikazan sledećim algoritmom.

Algoritam: Uopštena metoda promenljivih okolina – GVNS

Definisati okoline U_l, $1 \leq l \leq l_{max}$, i U'_l, $1 \leq l \leq l'_{max}$
Izabrati početno rešenje x
Poboljšati ga korišćenjem redukovane metode promenljivih okolina
repeat
  $l \leftarrow 1$
  while $l \leq l_{max}$ **do**
    Izabrati proizvoljno rešenje x' u okolini $U_l(x)$
    $l' \leftarrow 1$
    while $l' \leq l'_{max}$ **do**
      Naći rešenje x'' u okolini $U_{l'}$ (x') sa najmanjom vrednošću funkcije cilja
      if rešenje x'' ima manju vrednost funkcije cilja od rešenja x' **then**
        $x' \leftarrow x''$,
        $l' \leftarrow 1$
      else
        $l' \leftarrow l' + 1$
      end if
    end while
    if rešenje x'' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x''$,
      $l \leftarrow 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
    Ako je pronađeno bolje rešenje ažurirati vrednost najboljeg rešenja
until nije zadovoljen kriterijum zaustavljanja

Metoda promenljivog spusta (eng. Variable neighborhood descent – VND) iz okoline trenutnog rešenja, u svakoj iteraciji analizira najbolje rešenje. Maksimalan broj okolina koje će biti pretraživane je l_{max}. Polazi se od $l = 1$ i zatim se l povećava za 1 i na taj način se okoline menjaju sve do l_{max}. U slučaju da je $l_{max} = 1$, metoda promenljivog spusta se svodi na običnu lokalnu pretragu. Postupak rada metode je prikazan sledećim pseudokodom.

Algoritam: Metoda promenljivog spusta – VND

Definisati okoline U_l, $1 \leq l \leq l_{max}$, koje će se koristiti za pretragu
Generisati početno rešenje x
repeat
  $l \leftarrow 1$
  while $l \leq l_{max}$, **do**
    Nađi najbolje rešenje x' u okolini $U_l(x)$
    if rešenje x' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x'$,
      $l \leftarrow 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
  Ako je pronađeno bolje rešenje ažurirati vrednost najboljeg rešenja
until nije zadovoljen kriterijum zaustavljanja

3.2 Kodiranje rešenja i funkcija cilja

U ovom delu je opisano kodiranje rešenja za algoritam RVNS sa kojim je rešavan FLP-MD. Zatim je detaljno opisano kako se računaju vrednosti funkcije cilja i prikazan je pseudokod funkcije cilja.

Rešenje je predstavljeno binarnim nizom dužine $|F|$, gde je F skup potencijalnih lokacija za resurse. Ukoliko je na i-toj lokaciji resurs uspostavljen, na i-toj poziciji niza se nalazi vrednost 1, a inače 0. Na primer, niz 000001101 označava da ima devet potencijalnih lokacija za resurse, a resursi koji su uspostavljeni se nalaze na šestoj, sedmoj i devetoj lokaciji. Uspostavljeni resursi u rešenju su resursi koje je bilo neophodno uspostaviti da bi svaki korisnik bio alociran, ali tako da udaljenost svakog korisnika od resursa kome je dodeljen, ne bude veća od ograničenja za maksimalnu udaljenost korisnika od resursa. Kada je određen niz koji predstavlja rešenje, tada se saberu cene uspostavljanja uspostavljenih resursa i dobija se ukupna suma cena uspostavljanja uspostavljenih resursa koja se takođe smatra rešenjem. Kompletno rešenje je niz koji pokazuje koji su resursi uspostavljeni i ukupna suma cena uspostavljanja uspostavljenih resursa. Kod CPLEX-a kompletno rešenje sadrži i matricu koja pokazuje koji korisnik je dodeljen kom resursu. Matrica se sastoji od nula i jedinica i ako element matrice X_{ij} ima vrednost 1 tada je korisnik i dodeljen resursu j. Ako element matrice X_{ij} ima vrednost 0 tada korisnik i nije dodeljen resursu j.

Računanje vrednosti funkcije cilja se radi na sledeći način. Funkcija cilja koristi binarni niz koji predstavlja rešenje, niz cena uspostavljanja resursa, matricu udaljenosti svih korisnika od

svih resursa, lokalni pomoćni niz koji predstavlja iskorišćene resurse, ograničenje za maksimalnu udaljenost korisnika od resursa označenu sa L i promenljivu koja predstavlja ukupnu sumu cena uspostavljanja iskorišćenih resursa. Ukoliko lokalni pomoćni niz na i-toj poziciji sadrži vrednost 1, resurs na i-toj lokaciji je iskorišćen a ako na i-toj poziciji ima vrednost 0, resurs na i-toj lokaciji nije iskorišćen. Resurs je iskorišćen ako je uspostavljen i ako je na njega alociran bar jedan korisnik. Niz koji predstavlja rešenje je inicijalizovan pre početka izvršavanja funkcije cilja. Njegovi elementi su sa određenom verovatnoćom postavljeni na 1, što znači da su resursi na tim lokacijama uspostavljeni. Na početku se sve vrednosti u pomoćnom nizu postave na 0, što znači da ispočetka nema iskorišćenih resursa. Funkcija cilja se sastoji od dve for petlje. U prvoj, dvostrukoj for petlji, se za svakog korisnika iz skupa D, pronalazi bilo koji uspostavljen resurs iz skupa F, takav da rastojanje između tog korisnika i resursa nije veće od L. Kada se naiđe na prvi takav resurs, pretraga se prekida, a u pomoćnom nizu koji predstavlja iskorišćene resurse se na poziciji koja odgovara tom resursu postavi 1. Ukoliko nije pronađen nijedan takav resurs, vrednost funkcije cilja se postavlja na beskonačno tj. tada nema rešenja za uspostavljene resurse iz vektora koji predstavlja rešenje. U drugoj for petlji se elementi iz pomoćnog niza prepišu u niz koji predstavlja rešenje i izračunava se ukupna suma cena uspostavljanja resursa koji su iskorišćeni. Primetimo da u inicijalnom nizu koji predstavlja rešenje na početku rada funkcije cilja, mogu postojati uspostavljeni resursi koji nisu dodeljeni nijednom korisniku. Takvi resursi se zatvaraju tj. cena njihovog uspostavljanja ne ulazi u ukupnu sumu cena uspostavljanja resursa koja predstavlja rešenje. Prostije objašnjenje rada funkcije cilja bi bilo da ona prolazi kroz matricu udaljenosti korisnika od resursa i proverava za svako polje M_{ij} da li je resurs j uspostavljen. Ako resurs j nije uspostavljen prelazi se na sledeći resurs a ako je resurs uspostavljen proverava se da li udaljenost M_{ij} nije veća od ograničenja za maksimalnu udaljenost. Ako je resurs j uspostavljen i udaljenost M_{ij} nije veća od ograničenja za maksimalnu udaljenost, tada se resurs j označi kao iskorišćen. Na kraju ako je svaki korisnik alociran, saberu se cene uspostavljanja svih iskorišćenih resursa i ta suma je rešenje koje funkcija cilja vraća.

Algoritam: Funkcija cilja

Ulazni podaci: x - niz koji predstavlja rešenje, niz cena uspostavljanja resursa, cost - matrica udaljenosti svih korisnika od svih resursa, niz iskorišćenih resursa, ograničenje za maksimalnu udaljenost korisnika od resursa, skup potencijalnih lokacija za resurse, skup korisnika

niz iskorišćenih resursa ← sve vrednosti postaviti na 0
$v \leftarrow 0$
for i = 0 **to** |skup korisnika| **do**
    bestJ ← -1
    for j = 0 **to** |skup potencijalnih lokacija za resurse| **do**
        if $x[j]$ = 0 **then**
            $j \leftarrow j + 1$

```
        else if cost[i][j] <= ograničenje za maksimalnu udaljenost korisnika od resursa then
                bestJ ← j
                break
            end if
        end if
    end for
    if  bestJ = −1
        ne postoji rešenje za uspostavljene resurse
    else
        niz iskorišćenih resursa [bestJ] ← 1          // resurs se označava kao iskorišćen
    end if
end for
for j = 0 to |skup potencijalnih lokacija za resurse| do
    x[j] ← niz iskorišćenih resursa[j]
    if niz iskorišćenih resursa[j] = 1                // iskorišćen resurs
        v ← v + niz cena uspostavljanja resursa[j]
    end if
end for
Izlazni podaci: v
```

3.3 Odabir okolina

Koristi se k_max okolina koje se pretražuju. Broj k_max je unapred definisan. Oznaka $1 \leq k \leq k\_max$ zači da se na početku pretraga radi po okolini $k = 1$, zatim se k povećava za 1 i tako se pretraga po okolinama nastavlja sve do k_max okoline. Razlika između okolina je u tome što u k-toj okolini funkcija invert menja stanja k rasursa a funkcija swap menja stanja k parova resursa sa različitim stanjima. Sa verovatnoćom p_i se aktivira funkcija invert a s verovatnoćom $1 - p_i$ se aktivira funkcija swap. Sledi detaljan opis rada funkcija shake, invert i swap i njihovi pseudokodovi.

Neka je niz koji predstavlja rešenje označen sa x. Fukcija invert slučajnim izborom odabere k resursa i promeni im stanja. Uspostavljeni resursi se zatvaraju a zatvoreni resursi se uspostavljaju. Može se desiti da se više puta slučajno odaberu isti resursi, tako da ako se resurs odabere paran broj puta njegovo stanje će ostati isto a ako se odabere neparan broj puta, tada će njegovo stanje biti promenjeno. Proces se ponavlja sve dok rešenje ne bude dopustivo. U ovom kontekstu, rešenje se smatra dopustivim ukoliko mu je bar jedan resurs otvoren. Na primer, slučaj da rešenje postane nedopustivo se može desiti ako u nizu koji predstavlja rešenje, postoji tačno k uspostavljenih resursa i funkcija invert slučajnim izborom odabere baš tih k uspostavljenih resursa. Tada će ih zatvoriti i neće postojati ni jedan uspostavljen resurs tj. svi resursi će biti zatvoreni. Zbog toga se u while petlji proverava da li je, posle promene stanja slučajno izabranih k resursa, bar jedan resurs uspostavljen i ako ni jedan resurs nije uspostavljen, svi resursi kojima su promenjena stanja se vrate u prethodna stanja, a zatim se ponovo slučajnim izborom odabere k

resursa i njihova stanja se promene. While petlja prestaje sa radom kada posle invertovanja stanja *k* resursa postoji bar jedan uspostavljen resurs. Tada i cela funkcija invert završava sa radom.

Algoritam: Funkcija invert

```
Ulazni podaci: x, k
for i = 0 to k do
    invertIndex[i] ← izabrati proizvoljnu vrednost između 0 i dužine niza x
    x[invertIndex[i]] ← promeniti stanje resursa
end for
while dokle god  ni jedan resurs nije uspostavljen do
    for i = 0 to k do            // izmenjeni resursi se vrate u prethodno stanje
        x[invertIndex[i]] ← vratiti resurs u prethodno stanje
    end for
    for i = 0 to k do
        invertIndex[i] ← izabrati proizvoljnu vrednost između 0 i dužine niza x
        x[invertIndex[i]] ← promeniti stanje resursa
    end for
end while
```

Fukcija swap slučajnim izborom odabere *k* parova resursa iz vektora *x* koji predstavlja rešenje tj. sadrži informacije o tome koji su resursi, iz skupa svih resursa *F*, uspostavljeni a koji nisu. Ako su stanja oba resursa ista otvorena ili zatvorena, bira se novi par resursa, sve dok se ne izabere par resursa sa različitim stanjima. Pritom je regularno da se unutar tih k iteracija dva ili više puta izabere isti par. Kada se izabere par resursa čija su stanja različita, njihova stanja se promene. Uspostavljeni resurs se zatvori a zatvoreni resurs se uspostavi. Kod funkcije swap se ne može desiti slučaj da posle promene stanja resursa, ne bude ni jedan uspostavljen resurs, kao što je bio slučaj kod funkcije invert, zato što se pre promene stanja resursa bira par resursa sa različitim stanjima, tako da jedan od resursa u paru mora biti zatvoren, pa će posle promene stanja tog resursa koji je bio zatvoren taj resurs biti uspostavljen.

Algoritam: Funkcija swap

```
Ulazni podaci: x, k
for i = 0 to k do
        n1 ← izabrati proizvoljnu vrednost između 0 i dužine niza x
        n2 ← izabrati proizvoljnu vrednost između 0 i dužine niza x
        while x[n1] = x[n2] do
                n1 ← izabrati proizvoljnu vrednost između 0 i dužine niza x
                n2 ← izabrati proizvoljnu vrednost između 0 i dužine niza x
        end while
        x[n1] ← promeniti stanje resursa
        x[n2] ← promeniti stanje resursa
end for
```

3.4 Ostali aspekti algoritma

Funkcija RVNS implementira redukovanu metodu promenljivih okolina (Reduced Variable Neighborhood Search – RVNS) u cilju pronalaženja najboljeg rešenja za FLP-MD. Na početku se inicijalizuje početno rešenje u vidu niza čiji su elementi binarne vrednosti i predstavljaju resurse. Svaki element početnog rešenja sa definisanom verovatnoćom dobija vrednost 1. Ako element niza početnog rešenja ima vrednost 1 tada je resurs na toj poziciji uspostavljen a ako element ima vrednost 0 tada resurs na toj poziciji nije uspostavljen. U slučaju da se desi da posle inicijalizacije početnog rešenja nema uspostavljenih resursa, slučajno se odabere jedan resurs iz početnog rešenja i uspostavi se. Vrednost početnog rešenja se postavi kao najbolje rešenje. Kako je RVNS iterativni algoritam, glavni deo metode je petlja (u daljem tekstu glavna petlja) koja se izvršava unapred definisani broj iteracija. Taj unapred defnisani broj iteracija je kriterijum zaustavljanja algoritma i algoritam se izvršava sve dok god nije ispunjen kriterijum zaustavljanja tj. dok nije izvršen unapred definisani broj iteracija. Glavna petlja pronalazi najbolje rešenje i posle završetka glavne petlje, metoda RVNS vreća najbolje pronađeno rešenje. U glavnoj petlji se prvo sa unapred definisanom verovatnoćom poziva funkcija shake, koja u trenutnom rešenju uspostavlja unapred definisani broj resursa. Funkcija shake će slučajno odabrati unapred definisani broj resursa, gde neki od odabranih resursa mogu biti već uspostavljeni a neki ne, tako da će funkcija shake od slučajno odabranih resursa uspostaviti samo one koji do tada nisu bili uspostavljeni. Nakon shake operacije, poziva se funkcija cilja i izračunava vrednost trenutnog rešenja. Zatim se promenljivoj k dodeli vrednost 1. Promenljiva k će biti korištena za određivanje tipova okolina tj. broja resursa kojima funkcija invert menja stanja tako što uspostavljene resurse zatvori a zatvorene resurse uspostavi, ili funkcija swap koja menja stanja k parova resursa, sa različitim stanjima. Ona takođe uspostavljene resurse zatvori a zatvorene resurse uspostavi. Sledi while petlja za pretragu susednih okolina. While petlja će raditi dok se ne dostigne okolina k_max, gde funkcija invert menja stanja k_max resursa a funkcija swap menja stanja k_max parova resursa koji imaju različita stanja. Kada je $k = 1$ funkcija invert menja stanje jednog resursa a funkcija swap menja stanja jednog para resursa sa različitim stanjima. Ukoliko u trenutnoj okolini nije pronađeno bolje rešenje, prelazi se na narednu okolinu tako što se k poveća za jedan. Kako je sada $k = 2$, funkcije invert i swap menjaju stanja kod dva resursa, odnosno dva para resursa i tako sve dok k ne dostigne vrednost k_max. U while petlji prvo se vrednosti iz trenutnog rešenja sačuvaju u vektoru koji predstavlja prethodno rešenje. Zatim se odabere slučajni broj i ako je manji od unapred definisanog broja, poziva se funkcija invert a ako je slučajno odabrani broj jednak ili veći od unapred definisanog broja, poziva se finkcija swap. Zatim se poziva funkcija cilja i izračuna vrednost novog rešenja. Ako je vrednost novog rešenja manja od vrednosti trenutnog rešenja, pretraga se resetuje na prvi tip okoline i ažurira se vrednost trenutnog i najboljeg rešenja a ako nije, k se poveća za 1 da bi se pretraživala naredna okolina. Kada k dostigne k_max while petlja prestaje sa radom. Tada ako je trenutno rešenje manje od do tada pronađenog najboljeg rešenja, ono se postavi da bude najbolje do sada pronađeno rešenje. Kada se izvrše sve iteracije, glavna petlja prestaje sa radom i metod RVNS vraća najbolje pronađeno rešenje. Pseudokod za RVNS je prikazan sledećim algoritmom.

Algoritam: Funkcija RVNS

Ulazni podaci: x, maksimalan broj iteracija, v_shake, k, k_max

```
trenutno rešenje ← funkcija cilja(niz iskorišćenih resursa)    // početna vrednost rešenja
najbolje rešenje ← trenutno rešenje                            // početna najbolja vrednost rešenja
for i = 0 to maksimalan broj iteracija do
    if izabere se slučajan broj iz itervala [0, 1000) < 1 then
        shake(v_shake)
        trenutno rešenje ← funkcija cilja(niz iskorišćenih resursa)
        k ← 1
        while k ≤ k_max do
            prethodno rešenje ← x                              // čuvanje prethodnog rešenja
            if izabere se slučajan broj iz itervala [0, 100) < invertProbability then
                invert(k)
            else
                swap(k)
            end if
            novo rešenje ← funkcija cilja(niz iskorišćenih resursa)    // nova vrednost rešenja
            if novo rešenje < trenutno rešenje then
                trenutno rešenje ← novo rešenje
                k ← 1
            else
                x ← prethodno rešenje
                k ← k + 1
            end if
        end while
        if trenutno rešenje < najbolje rešenje then
            najbolje rešenje ← trenutno rešenje
        end if
end for
```

Izlazni podaci: najbolje rešenje

Funkcija shake – razmrdavanje slučajnim izborom odabira *v_shake* resursa iz vektora *x* koji predstavlja rešenje tj. čiji elemeti predstavljaju resurse i uspostavlja ih tj. dodeljuje im vrednost 1. U slučaju da je slučajnim izborom izabran resurs koji je već uspostavljen, ništa se ne menja, taj resurs i dalje ostaje uspostavljen, što se posebo ne razmatra. Ako izabrani resurs nije uspostavljen tj. u vektoru *x* ima vrednost 0, njemu se dodeljuje vrednost 1 i on se uspostavlja. Tako da posle izvršavanja funkcije shake od slučajno izabranih *k* resursa nije poznato koliko će neuspostavljenih resursa biti uspostavljeno. Može se desiti da ni jedan novi resurs ne bude uspostavljen, jer su svi slučajno odabrani resursi već uspostavljeni a može se desiti da svih slučajno izabranih *k* resursa budu upostavljeni, jer prethodno nisu bili uspostavljeni. Najverovatnije je da će neki broj resursa između 0 i *k* koji nisu bili uspostavljeni, biti uspostavljen. Funkcija shake omogućava da se rešenje ne zaglavi u lokalnom minimumu, a često se dešava da taj lokalni minimum bude takav da vrlo malo resursa bude u njemu uspostavljeno.

Algoritam: Funkcija shake – razmrdavanje

Ulazni podaci: *x*, *v_shake*

for *i* = 0 **to** *v_shake* **do**

    n ← izabrati proizvoljnu vrednost između 0 i dužine niza *x*

    x[*n*] ← 1

end for

4. Eksperimentalni rezultati

4.1 Test instance

Eksperimentalni rezultati su dobijeni korišćenjem UFLP ORLIB instanci preuzetih sa [18] koje se sastoje od:

- Dimenzija matrice udaljenosti svih korisnika od svih resursa.
- Cena uspostavljanja svih resursa.
- Matrice udaljenosti svih korisnika od svih resursa.

Instance su modifikovane, tako da su cene uspostavljanja resursa koje iznose 0, zamenjene sa cenama uspostavljanja resursa koje su iste kao za ostale resurse. Modifikovane su sve instance osim instanci capa, capb i capc, jer kod tih instanci nije bilo resursa čija je cena uspostavljanja bila 0. Element M_{ij} ulazne matrice udaljenosti svih korisnika od svih resursa predstavlja udaljenost i-tog korisnika od j-tog resursa. Korišćene su instance koje sadrže podatke za: 50 korisnika i 16 resursa, 50 korisnika i 25 resursa, 50 korisnika i 50 resursa i 1000 korisnika i 100 resursa. Za rešavanje FLP-MD su korišćena i različita ograničenja za maksimalne udaljenost korisnika od resursa označena sa L. Ti podaci se ne nalaze u originalnim ulaznim instancama već su odabrani proizvoljno i dodati u instance. Sa promenom vrednosti L dolazi i do promene rešenja. Detaljni rezultati rešavanja FLP-MD dobijeni korišćenjem CPLEX-a i RVNS-a su prikazani u tabeli 2 a poređenje brzine rada CPLEX-a i RVNS-a je prikazano na slikama 3, 4, 5 i 6. Kod svake instance postoji najveća udaljenost korisnika od resursa i kada se za L postavi najveća vrednost iz matrice udaljenosti svih korisnika od svih resursa, dobija se da je rešenje cena najjeftinijeg resursa jer se svi korisnici mogu dodeliti resursu čija je cena uspostavljanja najjeftinija. Takođe kod svake instance postoji minimalna vrednost za ograničenje maksimalne udaljenosti korisnika od resursa za koju postoji rešenje. Ta vrednost je najveća vrednost od svih minimalnih udaljenosti korisnika od resursa za svakog korisnika. Kada je L jednako toj minimalnoj vrednosti za koju postoji rešenje, dobija se najveća vrednost rešenja i uspostavlja se najveći broj resursa. U slučaju da se za L postavi vrednost manja od te najmanje udaljenosti za koju postoji rešenje, tada ne postiji rešenje, jer tada postoji korisnik ili korisnici koji se ne mogu dodeliti ni jednom resursu, zato što se od svih resursa nalaze na većoj udaljenosti od L tj. od ograničenja za maksimalnu udaljenost korisnika od resursa.

Promenama vrednosti ograničenja maksimalne udaljenosti svih korisnika od svih resursa, između maksimalne i minimalne udaljenosti, dešavaju se promene broja uspostavljanja resursa i samim tim i ukupne sume cena uspostavljanja uspostavljenih resursa. Da bi se te promene videle, za ograničenja maksimalne udaljenosti korisnika od resursa su postavljene odgovarajuće vrednosti. Tako se za različite vrednosti ograničenja maksimalne udaljenosti korisnika od resursa dobijaju različita rešenja. Za instance sa dimenzijom 50×16 za maksimalnu vrednost ograničenja maksimalne udaljenosti korisnika od resursa se uspostavlja jedan resurs a za minimalnu vrednosti ograničenja maksimalne udaljenosti korisnika od resursa se uspostavljaju dva resursa. Kod instanci sa dimenzijom 50×25 i 50×50 za maksimalnu vrednost ograničenja

udaljenosti korisnika od resursa, rešenje je cena uspostavljanja jednog resursa a za minimalnu vrednosti ograničenja maksimalne udaljenosti korisnika od resursa kao rešenje se dobija suma cena uspostavljanja tri uspostavljena resursa. Postavljanjem odgovarajuće vrednosti za L, dobija se da je rešenje suma cena uspostavljanja dva resursa.

4.2 Poređenje rezultata CPLEX-a i RVNS-a

FLP-MD je rešavan korišćenjem CPLEX rešavača i programom koji implementira redukovanu metodu promenljivih okolina – RVNS. Program za CPLEX je napisan korišćenjem programskog jezika Optimization Programming Language – OPL a program koji implementira RVNS je napisan korišćenjem programskog jezika C++. Za vrednosti parametara RVNS-a su uzete sledeće vrednosti do kojih se došlo preliminarnim testiranjima:

• k_max – Maksimalna veličina susednih okolina = 3,
• initProbability – Početna verovatnoća sa kojom se uspostavlja resurs = 0.25,
• invertProbability – Verovatnoća da se koristi invert operacija naspram swap operacije = 0.20,
• shakeLength – broj resursa koje uspostavlja shake operacije = 20,
• iterations – broj iteracija programa = 10000.

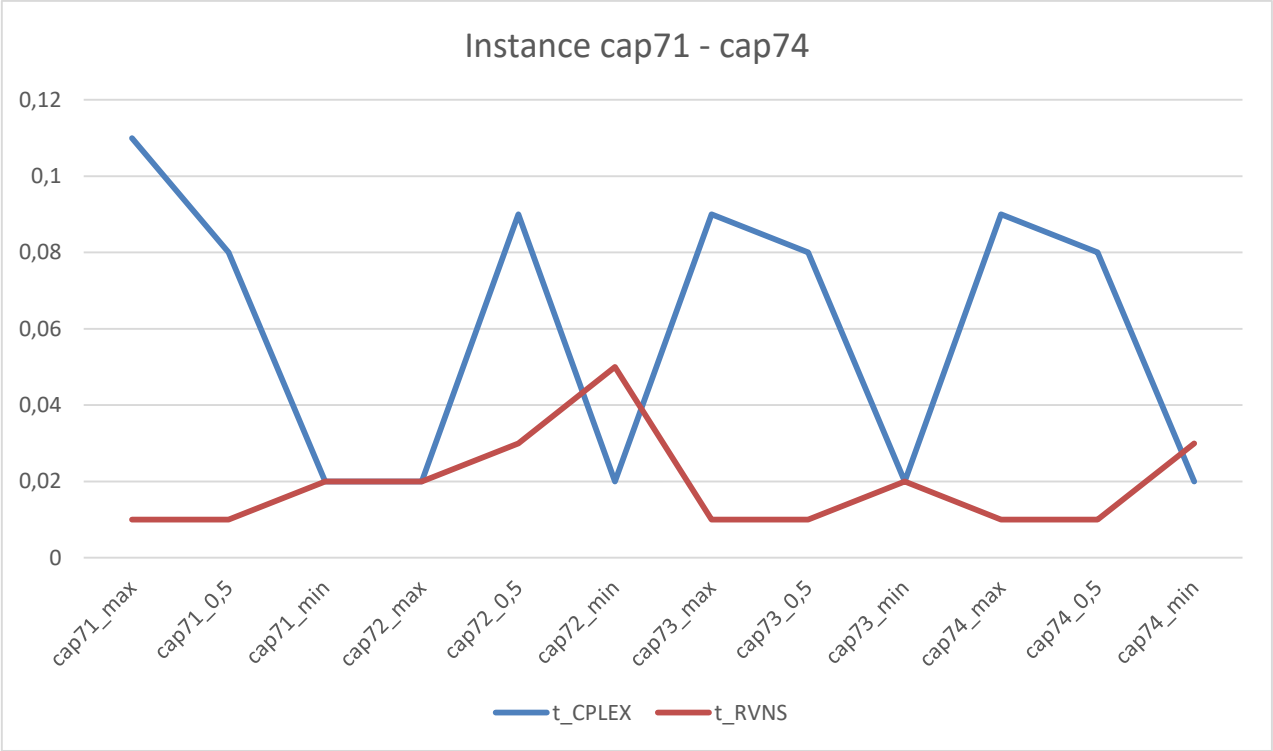
Tabela 2 koja prikazuje rezultate dobijene korišćenjem CPLEX-a i RVNS-a sadrži sledeće kolone:

• Instanca – Nazivi instanci.
• L – Ograničenje za maksimalne udaljenosti svih korisnika od svih resursa.
• Rešenje – Tačna rešenja.
• CPLEX – Rešenja dobijena korišćenjem CPLEX-a.
• t_{CPLEX} – vremena za koja je CPLEX izračunao rešenja.
• RVNS – Rešenja dobijena korišćenjem RVNS-a.
• t_{RVNS} – vremena za koja je RVNS algoritam izračunao rešenja.

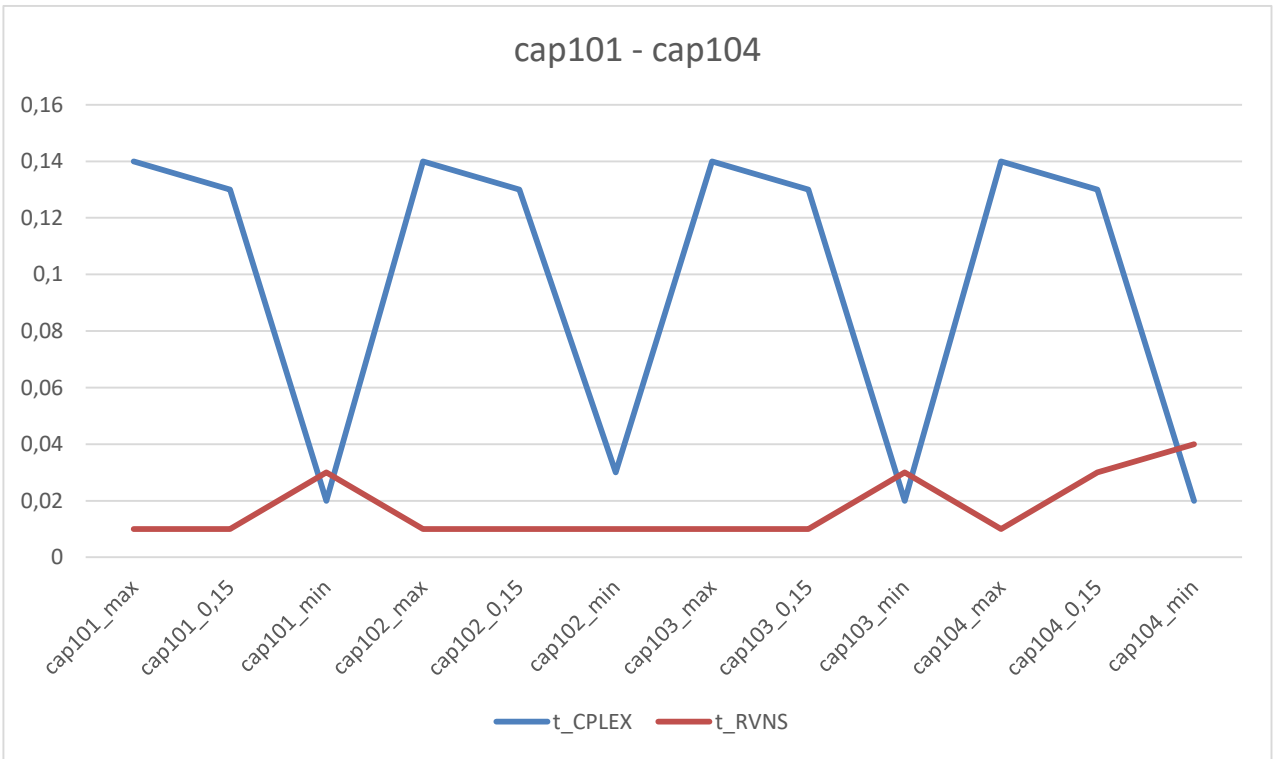
RVNS algoritam je pokretan deset puta za svaku instancu i pri svakom pokretanju dobijeno je optimalno rešenje. Optimalna rešenja su u tabeli označena sa opt. Rešenje je optimalno ako GAP za to rešenje ima vrednost 0.00 %. Vremena za koja CPLEX i RVNS izračunavaju rešenja su prikazana u sekundama. U tabeli 2 se može videti da i CPLEX i RVNS daju optimalna rešenja na svim instancama na kojima su testirani. Na vremena potrebna za izračunavanja rešenja najviše utiču dimenzije test instanci i ograničenja za maksimalne udaljenosti korisnika od resursa. Iz dobijenih rezultata može se primetiti da promene ograničenja za maksimalne udaljenosti korisnika od resursa mnogo više utiču na vremena potrebna za izračunavanja rešenja kod CPLEX-a dok na vremena potrebna za izračunavanja rešenja primenom RVNS-a ne utiču u tolikoj meri. RVNS algoritmu je skoro na svim instancama potrebno mnogo manje vremena da izračuna tačno rešenje za FLP-MD nego CPLEX-u osim u slučajevima kada su ograničenja za maksimalne udaljenosti korisnika od resursa minimalna. U tim slučajevima kada su ograničenja za maksimalne udaljenosti

Instanca	L	Rešenje	CPLEX	t_{CPLEX}	VNS	t_{VNS}
cap71_max	1361570.4	7500	opt	0.11	opt	0.01
cap71_0.5	680785.2	7500	opt	0.08	opt	0.01
cap71_min	203364.0	15000	opt	0.02	opt	0.02
cap72_max	1361570.4	12500	opt	0.02	opt	0.02
cap72_0.5	680785.2	12500	opt	0.09	opt	0.03
cap72_min	203364.0	25000	opt	0.02	opt	0.05
cap73_max	1361570.4	17500	opt	0.09	opt	0.01
cap73_0.5	680785.2	17500	opt	0.08	opt	0.01
cap73_min	203364.0	35000	opt	0.02	opt	0.02
cap74_max	1361570.4	25000	opt	0.09	opt	0.01
cap74_0.5	680785.2	25000	opt	0.08	opt	0.01
cap74_min	203364.0	50000	opt	0.02	opt	0.03
cap101_max	1415639.4	7500	opt	0.14	opt	0.01
cap101_0.15	212345.9	15000	opt	0.13	opt	0.01
cap101_min	130895.4	22500	opt	0.02	opt	0.03
cap102_max	1415639.4	12500	opt	0.14	opt	0.01
cap102_0.15	212345.9	25000	opt	0.13	opt	0.01
cap102_min	130895.4	37500	opt	0.03	opt	0.01
cap103_max	1415639.4	17500	opt	0.14	opt	0.01
cap103_0.15	212345.9	35000	opt	0.13	opt	0.01
cap103_min	130895.4	52500	opt	0.02	opt	0.03
cap104_max	1415639.4	25000	opt	0.14	opt	0.01
cap104_0.15	212345.9	50000	opt	0.13	opt	0.03
cap104_min	130895.4	75000	opt	0.02	opt	0.04
cap131_max	1415639.4	7500	opt	0.34	opt	0.01
cap131_0.15	212345.9	15000	opt	0.28	opt	0.01
cap131_min	130895.4	22500	opt	0.05	opt	0.07
cap132_max	1415639.4	12500	opt	0.36	opt	0.01
cap132_0.15	212345.9	25000	opt	0.28	opt	0.01
cap132_min	130895.4	37500	opt	0.03	opt	0.08
cap133_max	1415639.4	17500	opt	0.34	opt	0.01
cap133_0.15	212345.9	35000	opt	0.28	opt	0.01
cap133_min	130895.4	52500	opt	0.03	opt	0.07
cap134_max	1415639.4	25000	opt	0.36	opt	0.01
cap134_0.15	212345.9	50000	opt	0.27	opt	0.01
cap134_min	130895.4	75000	opt	0.03	opt	0.06
capa_max	147306.1	1365939	opt	70.63	opt	7.10
capa_0.5	73653.1	2801642	opt	261.30	opt	35.34
capa_0.25	36826.5	7284724	opt	40.63	opt	70.02
capb_max	149989.9	558666	opt	73.06	opt	6.11
capb_0.5	74995.0	654736	opt	70.53	opt	2.88
capb_0.25	37497.4	2937073	opt	67.05	opt	124.1
capc_max	151047.4	397560	opt	70.33	opt	5.66
capc_0.5	75523.7	503443	opt	70.23	opt	5.59
capc_0.25	37761.8	1884390	opt	37.50	opt	23.99

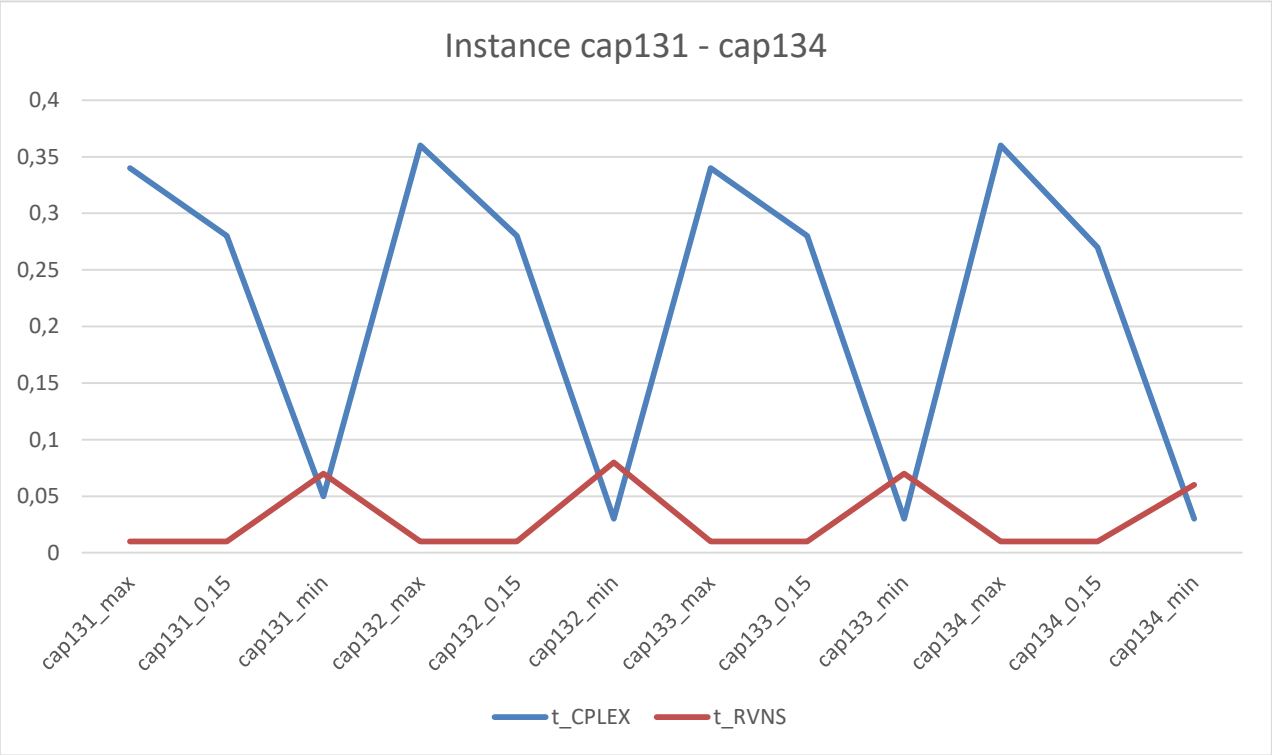
Tabela 2. Rezultati na modifikovanim ORLIB instancama



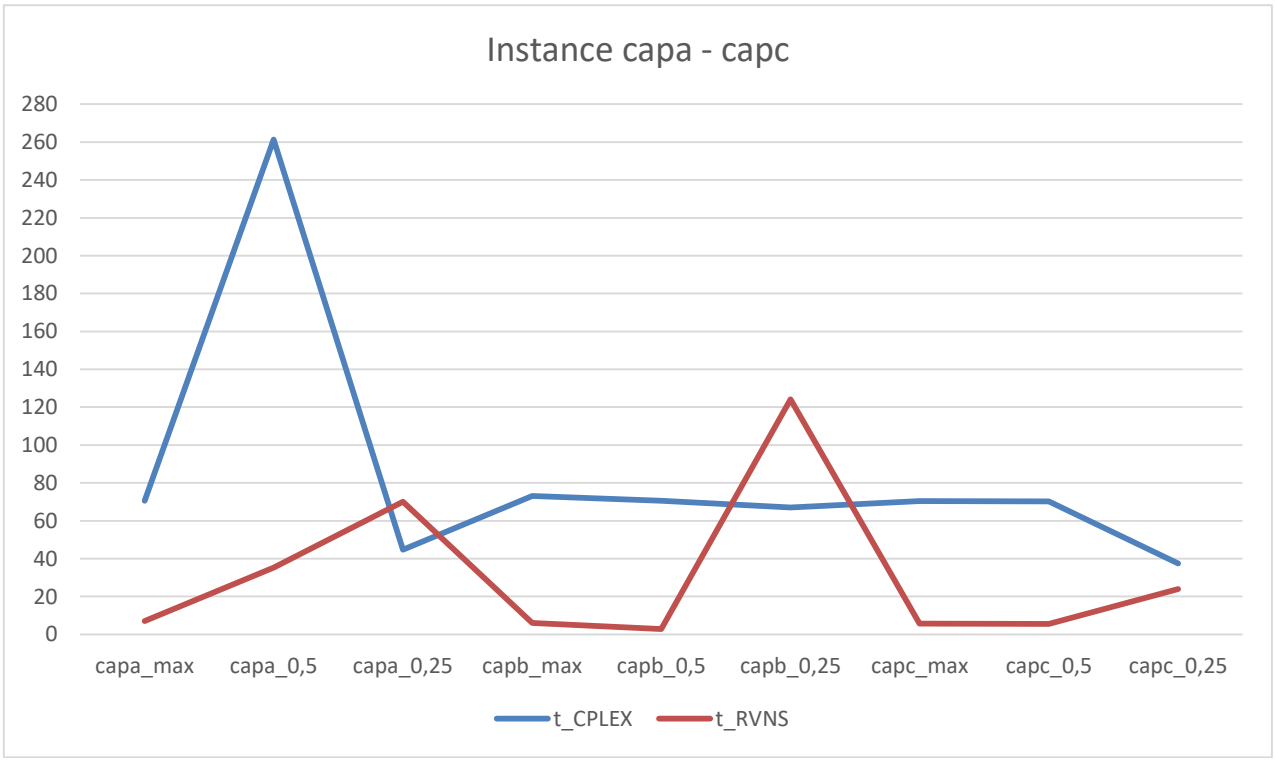
Slika 3. Poređenje brzine rada CPLEX-a i RVNS-a na instancama cap71 – cap74



Slika 4. Poređenje brzine rada CPLEX-a i RVNS-a na instancama cap101 – cap104



Slika 5. Poređenje brzine rada CPLEX-a i RVNS-a na instancama cap131 – cap134



Slika 6. Poređenje brzine rada CPLEX-a i RVNS-a na instancama capa – capc

korisnika od resursa minimalna, CPLEX reševaču i RVNS-u je približno potrebno isto vremena za izračunavanje tačnog rešenja. Što je veća vrednost ograničenja za maksimalne udaljenosti korisnika od resursa CPLEX-u je potrebno više vremena za izračunavanje tačnog rešenja dok na vreme koje je potrebno RVNS-u da izračuna tačnu vrednost rešenja, velika vrednost ograničenja za maksimalne udaljenosti korisnika od resursa nema nikakvog uticaja. Generalno, vreme potrebno za izračunavanje tačnog rešenja za FLP-MD primenom RVNS-a je mnogo brže, stabilnije, ravnomernije i manje podložno uticaju promena ograničenja za maksimalne udaljenosti korisnika od resursa nego vreme koje je potrebno za izračunavanje tačnog rešenja za FLP-MD primenom CPLEX-a.

5. Algoritam za redukciju dimenzija instanci za FLP-MD

5.1 Definicija algoritma

Osnovni razlog zbog koga egzaktni algoritmi nisu praktični je to što im je potrebno puno vremena za izračunavanje rešenja u slučaju instanci velikih dimenzija. U ovom radu je formulisan algoritam koji smanjuje dimezije instanci tj. smanjuje broj korisnika koje treba alocirati i broj resursa koje je neophodno uspostaviti i izračunava sumu cena uspostavljenih resursa za deo instance za koji je instanca smanjena. Zatim se na dobijene instance manjih dimenzija može primeniti neki egzaktni ili neki drugi metod za rešavanje FLP-MD, koji kada završi sa izračunavanjem, na dobijeni rezultat doda sumu cena uspostavljanja resursa koju je prethodno izračunao navedini algoritam i to je finalno rešenje za početnu instancu. Sa smanjenjem ograničenja za maksimalnu udaljenost korisnika od resursa, navedeni algoritam sve više smanjuje dimenzije instanci i samim tim sve više ubrzava rad metoda koji rešavaju FLP-MD, jer tada metode koje rešavaju FLP-MD koriste istance manjih dimenzija. Razlog za to, što sa smanjenjem ograničenja za maksimalnu udaljenost korisnika od resursa, navedeni algoritam sve više smanjuje dimenzije instanci, je taj što će udaljenost nekih korisnika od resursa samo na nekim mestima zadovoljavati uslov maksimalne udaljenosti, što će prouzrokovati uspostavljanje novih resursa gde je uslov maksimalne udaljenosti zadovoljen, a to će dalje prouzrokovati dodelu, svih korisnika koji zadovoljavaju uslov maksimalne udaljenosti kod tih novouspostavljenih resursa. U slučaju da se na uspostavljene resurse, koje algoritam za smanjenje dimenzija za FLP-MD uspostavi, mogu alocirati svi ostali korisnici iz instance, na primer kod instanci cap71_min, cap72_min, cap73_min i cap74_min, algoritam rešava FLP-MD u potpunosti. Sledi tekstualni opis algoritma.

Ako neki korisnici mogu biti dodeljeni samo jednom resursu, jer se samo od tih resursa ne nalaze na udaljenosti većoj od maksimalne definisane udaljenosti korisnika od resursa, ti resursi se uspostavljaju, a ti alocirani korisnici tj. vrste koje ih predstavljaju se neće nalaziti u novoj matrici. Tkođe u novoj matrici se neće nalaziti ni uspostavljeni resursi tj. kolone koje ih predstavljaju. Svi ostali korisnici koji mogu biti dodeljeni nekim od već uspostavljenih resursa u prethodnom koraku, se dodeljuju nekim od tih uspostavljenih resursa i vrste koje predstavljaju te korisnike se takođe neće nalaziti u novoj matrici. Zatim se izračuna ukupna suma cena uspostavljanja uspostavljenih resursa. Takođe se iz niza koji sadrži cene uspostavljanja resursa, brišu sve cene uspostavljenih resursa. Tako da će nova matrica imati isti broj kolona kao novi niz cena uspostavljanja resursa elemenata. Dobijena matrica je manjih dimenzija za broj svih alociranih korisnika i broj svih uspostavljenih resursa a niz cena uspostavljanja resursa je manji za broj svih uspostavljenih resursa. Na novodobijenu instancu manjih dimenzija, se primenjuje neki od metoda za rešavanje FLP-MD koji izračunava ukupnu sumu cena uspostavljanja resursa. Suma cena uspostavljanja uspostavljenih resursa, koju je izračunao metod za rešavanje FLP-MD, se sabere sa sumom cena uspostavljanja uspostavljenih resursa koju je prethodno izračunao algoritam

za smanjenje dimenzija instanci za FLP-MD i dobijeni zbir je konačno rešenje. Na primer ako imamo 100 korisnika i 30 resursa i primenom algoritma se ustanovi da 10 resursa mora biti uspostavljeno jer 20 korisnika zadovoljava uslov maksimalne udaljenosti samo kod tih 10 resursa, tada uspostavimo tih 10 resursa i dodelimo tih 20 korisnika tim uspostavljenim resursima. Zatim se ustanovi, da može da se dodeli još nekih drugih 30 korisnika tim uspostavljenim resursima, jer zadovoljavaju uslov maksimalne udaljenosti kod tih 10 uspostavljeih resursa i kod nekih drugih resursa i dodele se tih 30 korisnika tim već uspostavljenim resursima. Tada se iz originalne instance dimenzija 100×30 izbaci tih 50 alociranih korisnika i 10 uspostavljenih resursa i dobija se nova instanca dimenzija 50×20, što je značajno manja dimezija od dimenzije originalne instance. Kada se na novodobijenu instancu manjih dimenzija primeni neki od metoda koje rešavaju FLP-MD mnogo brže se dobije rešenje. U nastavku su prikazani svi koraci algoritma.

Ulazni podaci su: 1) Matrica udaljenosti svih korisnika od svih resursa 2) Niz cena uspostavljanja resursa. 3) Maksimalna udaljenost korisnika od resursa.
Izlazni podaci: 1) Nova matrica udaljenosti svih korisnika od svih resursa manjih dimenzija koja ne sadrži alocirane korisnike i uspostavljene resurse. 2) Suma cena uspostavljanja uspostavljenih resursa. 3) Niz cena uspostavljanja resursa koji ne sadrži cene uspostavljenih resursa.

1. Prolaskom kroz ulaznu matricu označiti vrste gde postoji samo jedan element manji ili jednak od ograničenja za maksimalnu udaljenost. Označiti i kolone koje predstavljaju resurse, gde se ti elementi nalaze. Ti resursi se uspostavljaju a ti korisnici tj. vrste koje ih predstavljaju se neće nalaziti u novoj matrici. Takođe u novoj matrici se neće nalaziti ni uspostavljeni resursi.

2. Sve korisnike koji zadovoljavaju uslov maksimalne udaljenosti,na bar jednoj poziciji, gde su resursi u prethodnom koraku uspostavljeni, dodeliti tim resursima. Svi takvi korisnici tj. vrste koje ih predstavljaju se neće nalaziti u novoj matrici.

3. Izračunati ukupnu sumu cena uspostavljanja uspostavljenih resursa.

4. U slučaju da su na uspostavljene resurse alocirani svi ostali korisnici iz instance, izračunata suma cena uspostavljanja uspostavljenih resursa je finalno rešenje tj. algoritam je rešio FLP-MD u potpunosti.

5. Izlaz iz algoritma je nova matrica manjih dimenzija koja ne sadrži alocirane korisnike i uspostavljene resurse, suma cena uspostavljanja uspostavljenih resursa i niz cena uspostavljanja reusrsa koji ne sadrži cene uspostavljenih resursa.

6. Na novo dobijenu instancu manjih dimenzija, se primenjuje neki od metoda za rešavanje FLP-MD koji izračunava ukupnu sumu cena uspostavljanja uspostavljenih resursa.

7. Sabrati sume cena uspostavljanja resursa dobijene u 3. i 6. koraku. Taj zbir je finalno rešenje.

5.2 Primer rada algoritma

Neka je ulazna matrica

$$\begin{bmatrix} 2 & 6 & 8 & 9 & 7 & 8 & 6 & 4 \\ 1 & 1 & 6 & 4 & 7 & 2 & 8 & 9 \\ 9 & 9 & 4 & 9 & 9 & 9 & 9 & 9 \\ 2 & 3 & 7 & 7 & 5 & 9 & 8 & 9 \\ 2 & 2 & 7 & 2 & 7 & 2 & 8 & 9 \\ 7 & 2 & 8 & 9 & 7 & 2 & 6 & 9 \\ 1 & 1 & 9 & 4 & 7 & 2 & 9 & 7 \\ 9 & 9 & 9 & 9 & 6 & 9 & 5 & 9 \\ 8 & 7 & 5 & 1 & 7 & 2 & 8 & 9 \\ 9 & 8 & 7 & 8 & 2 & 9 & 7 & 9 \end{bmatrix}$$

Neka su cene uspostavljanja resursa

$$[10, 75, 30, 90, 60, 50, 40, 20]$$

Neka je maksimalna distanca između korisnika i resursa $L = 5$.

• Identifikacija vrsta koje imaju samo jednu vrednost manju ili jednaku od L.
Treća vrsta ima samo u trećoj koloni vrednost koja nije veća od L. Uspostavlja se treći resurs i alocira se treći korisnik. Osma vrsta ima samo u sedmoj koloni vrednost koja nije veća od L. Uspostavlja se sedmi resurs i alocira se osmi korisnik. Deseta vrsta ima samo u petoj koloni vrednost koja nije veća od L. Uspostavlja se peti resurs i alocira se desti korisnik. Treća, osma i deseta vrsta se neće nalaziti u izlaznoj matrici. Takođe treća, peta i sedma kolona se neće nalaziti u izlaznoj matrici

$$\begin{bmatrix} 2 & 6 & 8 & 9 & 7 & 8 & 6 & 4 \\ 1 & 1 & 6 & 4 & 7 & 2 & 8 & 9 \\ 9 & 9 & \boxed{4} & 9 & 9 & 9 & 9 & 9 \\ 2 & 3 & 7 & 7 & 5 & 9 & 8 & 9 \\ 2 & 2 & 7 & 2 & 7 & 2 & 8 & 9 \\ 7 & 2 & 8 & 9 & 7 & 2 & 6 & 9 \\ 1 & 1 & 9 & 4 & 7 & 2 & 9 & 7 \\ 9 & 9 & 9 & 9 & 6 & 9 & \boxed{5} & 9 \\ 8 & 7 & 5 & 1 & 7 & 2 & 8 & 9 \\ 9 & 8 & 7 & 8 & \boxed{2} & 9 & 7 & 9 \end{bmatrix}$$

• Identifikacija vrsta koje imaju više od jedne vrednosti koja je manja ili jednaka od L i jedna od tih vredosti se nalazi u nekoj od kolona koje predstavljaju uspostavljene resurse. Četvrta vrsta ima nekoliko vrednosti manjih ili jednakih od L. Jedna od tih vrednosti se nalazi u petoj koloni koja predstavlja peti resurs koji je već uspostavljen. Deveta vrsta ima nekoliko vrednosti manjih ili

jednakih od L. Jedna od tih vrednosti se nalazi u trećoj koloni koja predstavlja treći resurs koji je već uspostavljen. Četvrta i deveta vrsta se neće nalaziti u izlaznoj matrici.

$$\begin{matrix} \\ \\ \\ \Rightarrow \\ \\ \\ \\ \Rightarrow \\ \\ \\ \end{matrix} \begin{bmatrix} 2 & 6 & 8 & 9 & 7 & 8 & 6 & 4 \\ 1 & 1 & 6 & 4 & 7 & 2 & 8 & 9 \\ 9 & 9 & 4 & 9 & 9 & 9 & 9 & 9 \\ \boxed{2} & \boxed{3} & 7 & 7 & \boxed{5} & 9 & 8 & 9 \\ 2 & 2 & 7 & 2 & 7 & 2 & 8 & 9 \\ 7 & 2 & 8 & 9 & 7 & 2 & 6 & 9 \\ 1 & 1 & 9 & 4 & 7 & 2 & 9 & 7 \\ 9 & 9 & 9 & 9 & 6 & 9 & 5 & 9 \\ 8 & 7 & \boxed{5} & \boxed{1} & 7 & \boxed{2} & 8 & 9 \\ 9 & 8 & 7 & 8 & 2 & 9 & 7 & 9 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}$$
$$\qquad\qquad\qquad\qquad\qquad\quad \Uparrow \qquad\quad \Uparrow \qquad\quad \Uparrow$$

• izračunava se zbir cena uspostavljanja uspostavljenih resursa tj. trećeg, petog i sedmog resursa i on iznosi 130.

Cene uspostavljenih resursa $= [10, 75, \boxed{30}, 90, \boxed{60}, 50, \boxed{40}, 20]$.

Izlaz iz algoritma je nova matrica dimenzija 5×5 koja ne sadrži treću, četvrtu, osmu, devetu i desetu vrstu i treću, petu i sedum kolonu. Niz cena uspostavljanja resursa koji ne sadrži cene uspostavljanja trećeg, petog i sedmog resursa i ukupan zbir cena uspostavljanja uspostavljenih resursa koji iznosi 130.

Izlazna matrica

$$\begin{bmatrix} 2 & 6 & 9 & 8 & 4 \\ 1 & 1 & 4 & 2 & 9 \\ 2 & 2 & 2 & 2 & 9 \\ 7 & 2 & 9 & 2 & 9 \\ 1 & 1 & 4 & 2 & 7 \end{bmatrix}$$

Cene uspostavljanja resursa

$$[10, 75, 90, 50, 20]$$

Zbir cena uspostavljanja resursa iznosi 130.

Sada se umesto na instancu dimenzije 10×8 primenjuje neki metod za rešavanje FLP-MD na instancu dimenzije 5×5 i kada izračuna rešenje, to rešenje se sabere sa zbirom cena uspostavljanja trećeg, petog i sedmog resursa koji iznosi 130, koji je izlaz iz algoritma i to je konačno rešenje. U primeru je korišćena matrica 10×8 zbog lakšeg opisa rada algoritma. U realnosti algoritam ima smisla koristiti na instancama velikih dimenzija, čija veličina dovodi do toga da je metodama za rešavanje FLP-MD potrebna velika količina vremena. Na takvim instancama jako velikih dimenzija smanjenje dimezija instanci bitno ubrzava rad metoda kojima se rešava FLP-MD.

5.3 Poređenje rezultata CPLEX-a na originalnim instancama i instancama redukovanim algoritmom

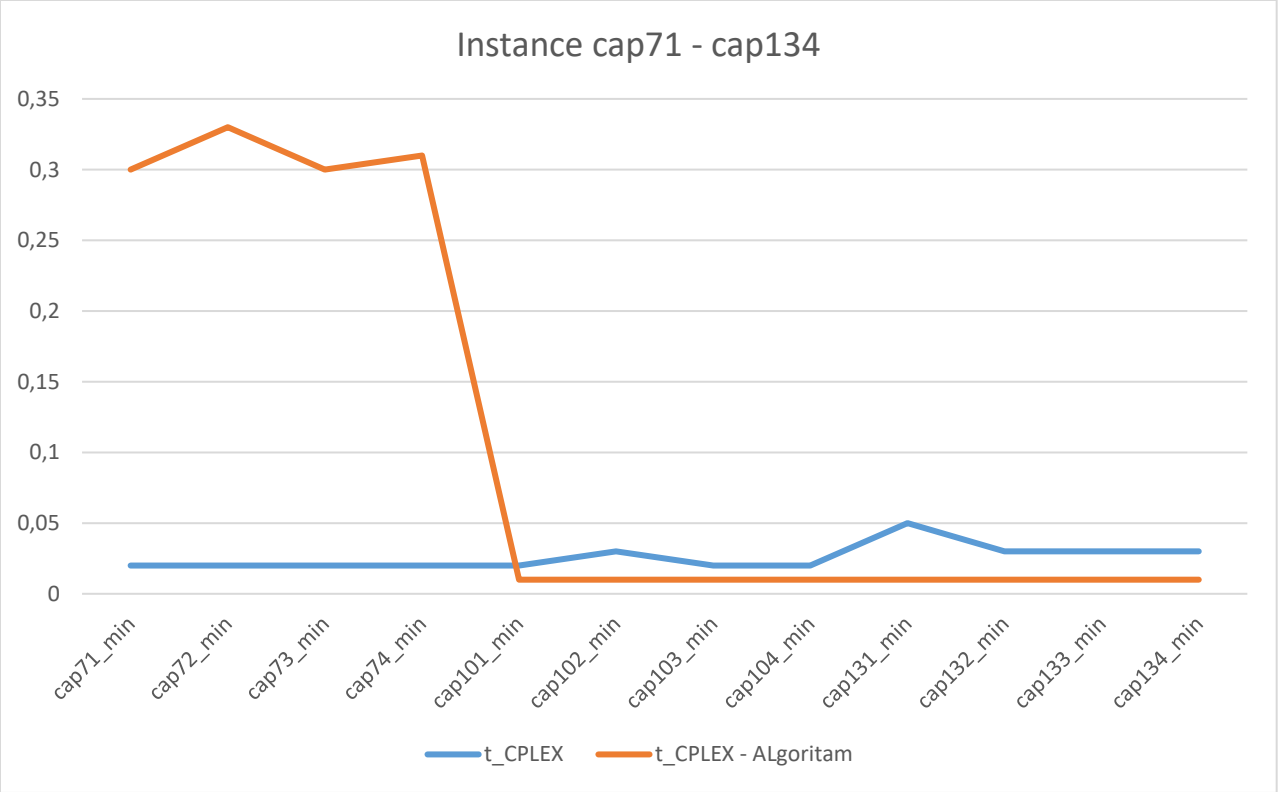
U tebeli 3 su prikazani rezultati CPLEX-a na originalnim instancama i na instancama čije su dimenzije redukovane primenom algoritma za redukciju dimenzija instanci za FLP-MD. Rešenja su optimalna i identična u oba slučaja i zbirovi vremena za koja algoritam redukuje instance i vremena za koje CPLEX izračunava rešenja na redukovanim instancama su mnogo manji nego vremena za koje CPLEX izračunava rešenja na originalnim instancama u slučaju kada su instance velikih dimenzija. U slučaju instanci manjih dimenzija, vremena su približno ista, ali za veće instance kao što su MT1 koja ima dimenzije 2000×2000, primena navedenog algoritma višestruko smanjuje vreme rada CPLEX-a potrebno za izračunavanje rešenja lokacijskog problema sa ograničenjem maksimalne udaljenosti. Vreme izvršavanja CPLEX-a na originalnoj instanci MT1 je 347.81 sekudi a vreme izvršavanja CPLEX-a na redukovanoj instanci primenom algoritma je 13.16 sekundi. Za instance cap71_min, cap72_min, cap73_min i cap74_min algoritam za redukciju dimezija instanci za FLP-MD u potpunosti rešava FLP-MD i dobija se potpuno rešenje. U tabeli 3 i na slici 7 za te instance je prikazano vreme rada algoritma. Za ostale instance je takođe u tabeli 3 i na slikama 7 i 8 prikazano vreme rada CPLEX-a na originalnim instancama i instancama koje su redukovane primenom algoritma za redukciju dimenzija instanci za FLP-MD. Originalna instanca MT1 je preuzeta sa sajta *max planck institute for informatics* sa linka https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/. Tabela 3 koja prikazuje rezultate CPLEX-a na originalnim instancama i na instancama čije su dimenzije redukovane primenom algoritma sadrži sledeće kolone:

- Instanca – Nazivi instanci.
- L – Ograničenja za maksimalne udaljenosti korisnika od resursa.
- Rešenje – Tačna rešenja.
- t_{CPLEX} – Vremena za koja je CPLEX izračunao rešenja na originalnim instancama.
- $t_{\text{CPLEX – Algoritam}}$ – Vremena za koja je CPLEX izračunao rešenja na instancama čije su dimenzije redukovane primenom algoritma za redukciju dimenzija instanci za FLP-MD.

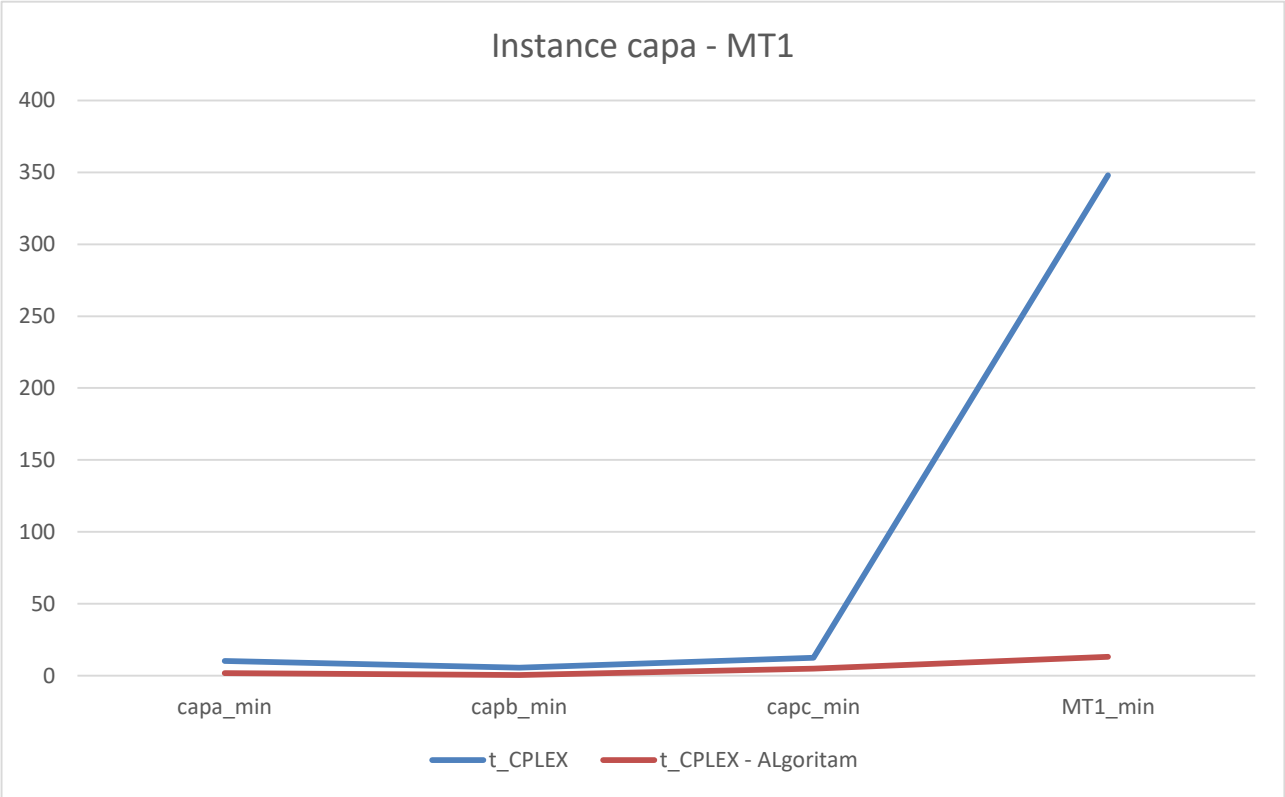
Vremena za koja je CPLEX izračunao rešenja na originalnim instancama i instancama čije su dimenzije redukovane primenom algoritma za redukciju dimenzija instanci za FLP-MD su prikazana u sekundama.

Instanca	*L*	Rešenje	t_{CPLEX}	$t_{CPLEX - ALgoritam}$
cap71_min	203364.0	15000	0.02	0.30
cap72_min	203364.0	25000	0.02	0.33
cap73_min	203364.0	35000	0.02	0.30
cap74_min	203364.0	50000	0.02	0.31
cap101_min	130895.4	22500	0.02	0.01
cap102_min	130895.4	37500	0.03	0.01
cap103_min	130895.4	52500	0.02	0.01
cap104_min	130895.4	75000	0.02	0.01
cap131_min	130895.4	22500	0.05	0.01
cap132_min	130895.4	37500	0.03	0.01
cap133_min	130895.4	52500	0.03	0.01
cap134_min	130895.4	75000	0.03	0.01
capa_min	18417.82875	23575975	10.3	1.95
capb_min	13441.68125	17864623	5.73	0.52
capc_min	16431.0248	9274625	12.53	4.92
MT1_min	2.548720	237682.026	347.81	13.16

Tabela 3. Rezultati primene algoritma za redukciju dimezija instanci za FLP-MD



Slika 7. Brzina rada CPLEX-a na originalnim instancama i na instancama redukovanim algoritmom



Slika 8. Brzina rada CPLEX-a na originalnim instancama i na instancama redukovanim algoritmom

6. Zaključak

U ovom master radu analiziran je lokacijski problem sa ograničenjem maksimalne udaljenosti (FLP-MD). Za njegovo rešavanje korišćen je CPLEX i program koji je dobijen implementacijom redukovane metode promenljivih okolina RVNS, koja je varijanta metode promenljivih okolina VNS. Za testiranje CPLEX-a i RVNS-a su korišćene instance dimezija 50×16, 50×25, 50×50 i 1000×100. Analizirajući tabelu u kojoj su prikazani rezultati dobijeni korišćenjem CPLEX-a i RVNS-a jasno se vidi da RVNS za manje instance daje optimalna rešenja za isto vreme kao i CPLEX, dok za veće instance RVNS daje takođe optimalna rešenja kao i CPLEX ali mnogo brže. Iz toga sledi da je prednost RVNS-a u tome što za instance velikih dimenzija ne zahteva puno vremena kao egzaktne metode a pri tome daje optimalna rešenja. Još jedna prednost koju ima RVNS se može videti analizirajući grafike koji prikazuju brzine rada CPLEX-a i RVNS-a. Može se izvesti zaključak da brzine rada implementacije RVNS-a ne osciluju toliko puno kao kod CPLEX-a. CPLEX ima velike oscilacije brzine izračunavanja rešenja koje su izazvane promenama vrednosti ograničenja za maksimalnu udaljenost korisnika od resursa, dok kod RVNS-a promena vrednosti ograničenja za maksimalnu udaljenost korisnika od resursa ne utiče puno na vreme potrebno za izračunavanje rešenja. Povećanje brzine kojom RVNS izračunava FLP-MD se može dodatno postići izborom odgovarajućih vrednosti parametara. U ovom master radu je takođe formulisan algoritam za redukciju dimenzija instanci za FLP-MD, koji redukuje dimenzije instanci i za deo instance koji je izbačen izračunava ukupnu sumu cena uspostavljanja uspostavljenih resursa. Zatim se na redukovane instance smanjenih dimenzija primenjuje CPLEX rešavač ili bilo koji drugi metod, koji bi izračunao rešenje i na to rešenje se dodaje rešenje koje je algoritam za redukciju instanci već izračunao i tako se dobija konačno rešenje. Ukupno vreme rada algoritma potrebno za redukciju instanci i CPLEX-a na redukovanim instancama potrebno za izračunavanje rešenja za FLP-MD je mnogo manje nego vreme rada CPLEX-a na originalnim instancama potrebno za izračunavanje rešenja za FLP-MD. U slučajevima kada se prilikom izvršavanja algoritma za redukciju dimenzija instanci dešavalo da se na uspostavljene resurse mogu alocirati i svi ostali korisnici, algoritam je rešavao FLP-MD u potpunosti. Algoritam za redukciju dimenzija instanci za FLP-MD je postizao značajne redukcije instanci prilikom smanjenja ograničenja maksimalne udaljenosti korisnika od resursa.

Pošto je lokacijski problem sa ograničenjem maksimalne udaljenosti FLP-MD jedna od mnogobrojnih varijanti prostog lokacijskog problema neograničenih kapaciteta UFLP, budući rad se može usmeriti ka primenama implementacije RVNS-a i implementacija drugih varijanti VNS-a za rešavanje sličnih vrsta lokacijskih problema. Može se ispitati da li su neke implementacije VNS-a specijalno pogodne za rešavanje određenih lokacijskih problema. Algoritam za redukciju dimenzija instanci za FLP-MD, se može inkorporirati u VNS i egzaktne metode. Algoritam se sa određenim modifikacijama može primeniti na slične lokacijske probleme. U razmatranom lokacijskom problemu FLP-MD, algoritam je redukovao instance na način koji je odgovarajući za FLP-MD. Modifikovani algoritam bi, u zavisnosti od lokacijskog problema, na odgovarajući način redukovao instance i za izbačene delove instanci bi izračunavao odgovarajuće rešenje za lokacijski

problem koji se rešava. Kao što je u slučaju FLP-MD smanjivanjem ograničenja za maksimalnu udaljenost, algoritam sve više smanjivao dimenzije instanci, analogno tome može se zaključiti da će kod nekih drugih lokacijskih problema, neki drugi različiti faktori uticati na to koliko značajnu redukciju instanci će algoritam da uradi.

Literatura

[1] Mladenović, Nenad, and Pierre Hansen. "Variable neighborhood search." *Computers & operations research* 24, no. 11 (1997): 1097-1100.

[2] Li, Xiaowei, and Xiwen Lu. "The facility location problem with maximum distance constraint." *Information Processing Letters* 184 (2024): 106447.

[3] Preuzeto sa https://laboi.fon.bg.ac.rs/wp-content/uploads/data/MO/Uvod.pdf Vujošević, M. (n.d.). *Uvod u optimizaciju*. Beograd.

[4] Sonuç, Emrullah, and Ender Özcan. "An adaptive parallel evolutionary algorithm for solving the uncapacitated facility location problem." *Expert Systems with Applications* 224 (2023): 119956.

[5] Goldreich, Oded. *P, NP, and NP-Completeness: The basics of computational complexity*. Cambridge University Press, 2010.

[6] Wigderson, Avi. "P, NP and mathematics–a computational complexity perspective." In *Proceedings of the ICM*, vol. 6, pp. 665-712. 2006.

[7] Nickel, Stefan, Justo Puerto, and Antonio M. Rodríguez-Chía. "Location problems with multiple criteria." *Location science* (2019): 215-260.

[8] Drezner, E. "Facility location: A survey of applications and methods." *Journal of the Operational Research Society* 47, no. 11 (1996): 1421-1421.

[9] Brandeau, Margaret L., and Samuel S. Chiu. "An overview of representative problems in location research." *Management science* 35, no. 6 (1989): 645-674.

[10] Farahani, Reza Zanjirani, Masoud Hekmatfar, Alireza Boloori Arabani, and Ehsan Nikbakhsh. "Hub location problems: A review of models, classification, solution techniques, and applications." *Computers & industrial engineering* 64, no. 4 (2013): 1096-1109.

[11] Drezner, Zvi, and Horst W. Hamacher, eds. *Facility location: applications and theory*. Springer Science & Business Media, 2004.

[12] Klose, Andreas, and Andreas Drexl. "Facility location models for distribution system design." *European journal of operational research* 162, no. 1 (2005): 4-29.

[13] Yang, Zhen, Feng Chu, and Haoxun Chen. "A cut-and-solve based algorithm for the single-source capacitated facility location problem." *European Journal of Operational Research* 221, no. 3 (2012): 521-532.

[14] Guastaroba, Gianfranco, and Maria Grazia Speranza. "A heuristic for BILP problems: the single source capacitated facility location problem." *European Journal of Operational Research* 238, no. 2 (2014): 438-450.

[15] Barros, Ana Isabel, and Martine Labbé. "The multi-level uncapacitated facility location problem is not submodular." *European Journal of Operational Research* 72, no. 3 (1994): 607-609.

[16] Yang, Xin-She, and Slawomir Koziel, eds. *Computational optimization and applications in engineering and industry*. Vol. 359. Springer Science & Business Media, 2011.

[17] Zenios, Stavros A., ed. *Financial optimization*. Cambridge university press, 1993.

[18] UFLP ORLIB instance preuzete sa https://people.brunel.ac.uk/~mastjjb/jeb/info.html

Biografija autora

Staša Rašić rođen je 19. jula 1971. godine u Zemunu. Osnovne akademske studije na Matematičkom fakultetu Univerziteta u Beogadu na smeru računarstvo i informatika završio je 2011. godine i stekao titulu Diplomirani matematičar. Master akdameske studije na Matematičkom fakultetu Univerziteta u Beogadu na Katedri za računarstvo i informatiku završio je 2013. godine i stekao titulu Master matematičar. Još jedne master akademske studije, druge po redu, upisao je 2023. godine na Matematičkom fakultetu Univerziteta u Beogradu na Katedri za računarstvo i informatiku i položio je sve ispite predviđene programom studija.