Union Univerzitet
Računarski fakultet

Dalibor V. Ristić

IMPLEMENTACIJA METODA PROMENLJIVIH OKOLINA ZA REŠAVANJE *P*-SLEDEĆI PROBLEMA NAD SKUPOM ČVOROVA GRAFA

Doktorska disertacija

Beograd, 2023.

Union University
Faculty of Computing

Dalibor V. Ristić

VARIABLE NEIGHBORHOOD SEARCH METHOD IMPLEMENTATION FOR SOLVING THE VERTEX-RESTRICTED *P*-NEXT PROBLEMS

Doctoral dissertation

Belgrade, 2023.

Podaci o mentoru i članovima komisije

Mentor:

dr Dragan Urošević
redovni profesor
Union Univerzitet, Računarski fakultet

Članovi komisije:

dr Dragan Urošević
redovni profesor
Union Univerzitet, Računarski fakultet

dr Filip Marić
redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Milanka Gardašević Filipović
vanredni profesor
Union Univerzitet, Računarski fakultet

Implementacija metoda promenljivih okolina za rešavanje *p*-sledeći problema nad skupom čvorova grafa

Sažetak

Problem *p*-centra je već dugo predmet interesovanja operacionih istraživanja. Poznat je još od sredine prošlog veka i predstavlja identifikaciju lokacija za postavljanje *p* centara, kao i njihovu dodelu korisnicima, na takav način da je maksimalna udaljenost korisnika od dodeljenog centra minimizovana. Postoji mnogo, kako egzaktnih matematičkih modela, tako i heurističkih algoritama koji uspešno rešavaju problem *p*-centra. Međutim, vremenom se postavilo pitanje šta u slučaju otkaza dodeljenog centra. U poslednjoj dekadi, kao potencijalni odgovor na ovo pitanje, predstavljeno je proširenje problema *p*-centra poznato kao problem *p*-sledećeg centra. Problem *p*-sledećeg centra predstavlja identifikaciju lokacija za postavljanje i dodelu *p* centara, ali na takav način da je minimizovana maksimalna suma rastojanja korisnika do najbližeg centra i rastojanja između tog i njemu najbližeg centra.

Poseban tip, poznat kao problemi nad skupom čvorova grafa, čine problemi optimalne selekcije centara iz skupa čvorova grafa koji predstavljaju sve korisnike i potencijalne centre. Postoji mali broj članaka i algoritama koji se bave problemom otkaza centra i stoga je u disertaciji definisano nekoliko problema i rešenja upravo iz klase problema ograničenih izborom centara iz skupa čvorova grafa. Posmatrani su problemi *p*-sledećeg, kao i *p*-drugog centra koji predstavlja identifikaciju *p* od potencijalnih *n* centara na takav način da je minimizovana maksimalna suma rastojanja korisnika do najbližeg i drugog najbližeg centra.

Na mogućnost otkaza centara, ne mora se posmatrati samo u kontekstu *p*-centar problema. U disertaciji je definisan i *p*-sledeći medijan problem koji se bavi otkazom centara u kontekstu poznatog *p*-medijan problema. Problem *p*-sledećeg medijana se definiše kao identifikacija lokacija *p* centara sa ciljem minimizacije ukupne sume rastojanja od svih *n* korisnika do najbližih centara plus rastojanja između tih centara i njima najbližih centara. Problem *p*-medijana se definiše kao izbor *p* centara sa ciljem minimizacije sume rastojanja svih korisnika do najbližih centara.

Na problem otkaza centra se može posmatrati i kao na mogućnost otkaza ne samo jednog dodeljenog centra, već i više njih. Stoga su u disertaciji definisani i problemi *p*-*α*-sledećih centara, *p*-*α*-najbližih centara i *p*-*α*-sledeći medijan kao uopštenja prethodno pomenutih problema, tako da rešenja problema svakom od *n* korisnika dodeljuju više zamenskih centara, tj. jedan primarni i $\alpha - 1$ zamenskih sa ciljem minimizacije rastojanja među njima u kontekstu proširenja *p*-centar, odnosno *p*-medijan problema.

Svi pomenuti problemi su NP-teški problemi i njihovo egzaktno rešavanje, posebno u slučaju instanci sa velikim brojem čvorova, iziskuje gotovo neograničene memorijske i procesorske resurse. Stoga su u disertaciji rešenja svih problema predložena kao heuristički algoritmi oslonjeni na metod promenljivih okolina kao generički okvir za implementaciju algoritama pretrage. Predloženi algoritmi su testirani nad u literaturi već poznatim *OR-Library* test skupom podataka, ali i nad novo-generisanim skupovima koji predstavljaju veće instance problema sa različitim gustinama grafova nad kojima su definisani. Dobijena su optimalna ili skoro-optimalna rešenja u vremenskim okvirima srazmernim veličini tretiranog problema.

Ključne reči: metod promenljivih okolina, problem *p*-sledećeg centra, problem *p*-drugog centra, problem *p*-*α*-sledećih centara, problem *p*-*α*-najbližih centara, *p*-sledeći medijan problem, *p*-*α*-sledeći medijan problem, *OR-Library* test platforma, heuristički algoritmi, kombinatorna optimizacija.

Naučna oblast: algoritmi i optimizacija.
Uža naučna oblast: heuristički algoritmi.

Variable neighborhood search method implementation for solving the vertex-restricted *p*-next problems

Abstract

The *p*-center problem has been the subject of interest in the field of operational research for a long time. It has been known since the middle of the previous century and represents the identification of locations for placing *p* centers, as well as their allocation to users, in such a way as to minimize the maximum distance to the assigned center. There are numerous, both exact mathematical models and heuristic algorithms that are successfully solving the *p*-center problem. However, over time, the question arose of what should be done in the event of exclusion of the assigned center. In the previous decade, as a potential answer to this question, an extension of the *p*-center problem known as the *p*-next center problem was presented. The *p*-next center problem represents the identification of locations for placing and assigning *p* centers, but in such a way as to minimize the maximum sum of the user's distance to the nearest center and the distance between that center and the next nearest center.

A special problem type, known as the vertex-restricted problem, consists of optimal selection of centers only from a set of graph nodes that represent all users and potential centers. There are a small number of articles and algorithms that deal with the center exclusion problem, and therefore several problems and solutions are defined in the dissertation precisely from the class of problems limited by the selection of centers from the set of graph nodes. There are observed the problems of the *p*-next center, as well as the *p*-second center, which represents the identification of *p* out of *n* possible centers in such a way that the maximum sum of the user's distance to the nearest and the second nearest center is minimized.

The possibility of cancellation of centers does not have to be viewed only in the context of the *p*-center problem. The dissertation also defined the *p*-next median problem, which deals with center failure in the context of the well-known *p*-median problem. The *p*-next median problem is defined as the identification of the locations of *p* centers with the goal of minimizing the total sum of the distances from all *n* users to the nearest centers plus the distances between these centers and the centers closest to them. Contrary to that, the *p*-median problem is defined as the selection of *p* centers with the goal of minimizing the sum of the distances of all users to the nearest centers.

The problem of cancellation of the center can be viewed as the possibility of cancellation of not only one assigned center, but also several centers. Therefore, in the dissertation, the problems of *p*-α-next centers, *p*-α-closest centers and *p*-α-next median are defined as generalizations of the previously mentioned problems, so that the solutions of the problems assign several backup centers to each of the *n* users, i.e., one primary and $\alpha - 1$ backup centers with the aim of minimizing the distance between them in the context of the extended *p*-center, or *p*-median problem.

All the mentioned problems are NP-hard problems and their exact solutions, especially in the case of instances with large number of nodes, require almost unlimited memory and processing resources. Therefore, in the dissertation, solutions to all problems were proposed as heuristic algorithms based on variable neighborhood search method as a generic framework for implementing search algorithms. The proposed algorithms were tested on the OR-Library data set already known in the literature, but also on newly generated sets that represent larger instances of problems with different densities of the graphs over which they are defined. Optimal or near-optimal solutions were obtained in time intervals proportional to the size of the treated problems.

Keywords: variable neighborhood search, *p*-next center problem, *p*-second center problem, *p*-α-next centers problem, *p*-α-closest centers problem, *p*-next median problem, *p*-α-next median problem, OR-Library test set, heuristic algorithms, combinatorial optimization.

Scientific field: algorithms and optimization.
Scientific subfield: heuristic algorithms.

Predgovor

Hakimi je 1964. godine predstavio problem apsolutnog centra [13] sa željom da locira policijsku stanicu ili bolnicu tako da se maksimalna udaljenost stanice do grupisanih naselja povezanih sistemom autoputeva svede na minimum. Neka je dat graf $G = (V, E)$ sa skupom čvorova $V = \{v_1, v_2, \dots, v_n\}$, težinom w_j čvora $v_j \in V$ i dužinom l_{ij} grane $(i, j) \in E$ koja povezuje čvorove v_i i v_j. Cilj problema apsolutnog centra je da pronađe tačku x na nekom od čvorova ili grana tako da je vrednost $max_{j=1,\dots,n}\{w_j\, d(v_j, x)\}$ minimizovana, pri čemu je $d(v_j, x)$ dužina najkraćeg puta između čvora v_j i tačke x. Optimalna vrednost je nazvana apsolutni radijus grafa *G*. Ukoliko je *x* ograničeno na čvorove grafa, dobija se centar grafa *G* i optimalna vrednost je radijus grafa *G*. Centar grafa *G* nije nužno i apsolutni centar grafa *G*, tj. apsolutni radijus je često manji od radijusa grafa. Recimo, u slučaju jednostavnog grafa sa dva čvora težine 1 i grane koja ih povezuje dužine 1, apsolutni radijus je 0.5, a radijus 1.

Kao generalizaciju problema apsolutnog centra, Hakimi je 1965. predstavio i problem *p*-centra [14]. Neka je $X_p = \{x_1, x_2, \dots, x_p\}$ skup od p tačaka u grafu *G*, a rastojanje $d(X_p, v_j)$ između X_p i čvora v_j izračunato kao $min_{i=1,\dots,p}\, d(x_i, v_j)$. Problem *p*-centra je u tom slučaju identifikovati skup X_p koji čine p tačaka iz grafa *G* na taj način da je vrednost $max_{j=1,\dots,n}\{w_j\, d(v_j, Xp)\}$ minimizovana. Ovako definisan *p*-centar problem predstavlja tzv. "*network location*" problem. U literaturi se mogu naći još nekoliko varijanti problema:

- netežinski *p*-centar problem: težine svih čvorova/grana su jednake,
- diskretan *p*-centar problem: graf $G = (J \cup I, E)$ je bipartitivan i kompletan, pri čemu J predstavlja skup svih potencijalnih lokacija zahtevanih tačaka, a I skup svih korisnika,
- *p*-centar problem nad skupom čvorova grafa (*eng. vertex-restricted p-center problem*): X_p je ograničen da bude podskup skupa čvorova grafa *G*.

Disertacija se isključivo bavi *p*-problemima nad skupom čvorova grafa, tako da se u nastavku teksta u definicijama problema i opisu algoritama za rešavanje podrazumeva da je reč o ovom tipu problema. Sa druge strane, algoritmi koji se u disertaciji predlažu kao rešenja svih *p*-problema ne zahtevaju da važi pretpostavka o nejednakosti trougla. To u opštem slučaju, bez gubitaka u smislu generalizacije problema, omogućava da se problemi tretiraju kao netežinski u pogledu čvorova grafa, tj. $max_{j=1,\dots,n}\{w_j\, d(v_j, Xp)\}$ postaje ekvivalentno problemu $max_{j=1,\dots,n}\, d'(v_j, Xp)$, gde je $d'(v_j, Xp) = w_j\, d(v_j, Xp)$.

Pojava *p*-centar problema je motivisala definisanje i drugih *p*-problema kao što su: "*capacitated p-center problem*", "*conditional p-center problem*", "*continuous p-center problem*", "*p-center problem with uncertain parameters*", "*p-median problem*", "*p-next center problem*" itd.

Problem *p*-sledećeg centra (*eng. p-next center problem*) daje akcenat na praktičnoj primeni rešenja problema. Posmatran kao problem rasporeda bolnica koje opslužuju obližnja naselja, rešenje *p*-sledeći centar problema pokušava da odgovori i na problem otkaza dodeljenog centra, tj. zatvaranja dodeljene bolnice. Upravo je ovaj problem, odnosno generalno problem nemogućnosti dodeljenog centra da opsluži svoje korisnike, motivisao izradu ove disertacije. U radu je definisano nekoliko problema i predloženo više algoritama koji pokušavaju da obezbede zamenski centar kao rešenje problema potencijalnog ispada jednog ili više dodeljenih centara.

Disertacija je organizovana u šest poglavlja.

Poglavlje 1 je uvodnog karaktera i sadrži definicije *p*-centar i *p*-medijan problema nad skupom čvorova grafa kao polaznih problema za definisanje svih problema i algoritama kojima se bavi disertacija. Ukratko je opisana i motivacija za proširenje ovih problema, kao i osnovni okvir za izgradnju svih u disertaciji predstavljenih algoritama za rešavanje *p*-problema. Algoritmi su zasnovani na metodu promenljivih okolina (*eng. variable neighborhood search method*, VNS), koji je poznat kao generički okvir za izgradnju heurističkih algoritama pretrage.

U Poglavlju 2 je razmatran problem *p*-sledećeg centra. Definisan je problem kao *p*-sledeći centar problem nad skupom čvorova grafa i predstavljen heuristički algoritam zasnovan na metodu promenljivih okolina koji

efikasno rešava ovaj problem. Predložena je i modifikacija algoritma koja u izvesnom kontekstu može prepoznati i globalno optimalna rešenja *p*-sledeći centar problema. Prezentovani su i rezultati testiranja algoritma nad u literaturi poznatim test skupom podataka, kao i rezultati poređenja sa postojećim algoritmima za rešavanje *p*-sledeći centar problema. Dati su i rezultati testova nad novo-generisanim test skupovima sa namerom da se oceni primenljivost algoritma i na velike instance problema, kao i na instance različitih gustina grana.

Poglavlje 3 se bavi problemom *p*-drugog centra. Definisan je *p*-drugi centar problem u kontekstu *p*-problema nad čvorovima grafa. Predstavljen je efikasan heuristički algoritam promenljivih okolina za rešavanje problema, kao i modifikacija algoritma sposobna da u nekim slučajevima prepozna i globalno optimalna rešenja problema. Predstavljeni su rezultati testiranja algoritma nad u literaturi poznatim *OR-Library* test skupom podataka. Takođe, algoritam je testiran i nad novim test instancama koje su znatno veće u pogledu broja čvorova i grana. Prezentovani su i rezultati testova u slučaju grafova različitih gustina.

U Poglavlju 4 se razmatraju generalizacije problema iz prethodna dva poglavlja. Definisani su problemi *p*-*α*-sledećih i *p*-*α*-najbližih centara kao proširenja *p*-sledeći i *p*-drugi centar problema većim brojem zamenskih centara. Rešenja ovih problema svakom korisniku dodeljuju α centara koji ih opslužuju, tj. jedan primaran centar i $\alpha - 1$ zamenskih u slučaju ispada primarnog centra. Prezentovani su rezultati izvršenja algoritama nad *OR-Library* test skupom podataka, kao i poređenja sa rezultatima algoritama iz prethodnih poglavlja.

U Poglavlju 5 je definisan novi *p*-sledeći medijan problem, kao i njegova generalizacija *p*-*α*-sledeći medijan problem. Problem *p*-sledećeg medijana, obezbeđuje jedan zamenski centar u slučaju otkaza primarnog centra u kontekstu *p*-medijan problema, dok rešenje *p*-*α*-sledeći medijan problema nudi $\alpha - 1$ zamenskih centara. Predloženi i opisani su heuristički algoritmi zasnovani na metodu promenljivih okolina za rešavanje ovih problema nad skupom čvorova grafa. Na kraju, predstavljeni su i rezultati testiranja ovih algoritama nad *OR-Library* test podacima. Problem *p*-sledećeg medijana je ekvivalentan *p*-2-sledeći medijan problemu, tako da su prezentovani i rezultati međusobnog poređenje ovih algoritama.

Poglavlje 6 je završnog karaktera i sadrži kratak osvrt na novo-definisane probleme i predložene heurističke algoritme za rešavanje posmatranih problema, kao i na rezultate testiranja, efikasnost i primenljivost ovih algoritama.

U pomenutim poglavljima su u cilju jednostavnosti i preglednosti dati sumarni rezultati testiranja predloženih algoritama za rešavanje *p*-problema nad skupom čvorova grafa. U Dodatku A, Dodatku B.1, Dodatku C i Dodatku D su tabelarno predstavljeni detaljni rezultati za svaku od test instanci nad kojom su respektivno testirani algoritmi za probleme *p*-sledećeg centra, *p*-drugog centra, *p*-*α*-sledećih centara, *p*-*α*-najbližih centara i *p*-sledećeg i *p*-*α*-sledećeg medijana. U Dodatku B.2 su upoređeni problemi *p*-centra, *p*-sledećeg centra i *p*-medijana sa problemom *p*-drugog centra i tabelarno predstavljeni rezultati poređenja algoritama za rešavanje ovih problema.

Originalni doprinos autora predstavljaju sledeći rezultati:

- Poglavlje 2: Svojstva 2.1, 2.2, 2.3 i 2.4 koja pokazuju da su teorijske karakteristike koje važe za *p*-centar problem [28] primenljive i na problem *p*-sledećeg centra; Svojstvo 2.5 za identifikaciju lokalnih optimuma, kao i Svojstvo 2.6 koje daje mogućnost egzaktnog rešavanja pojedinih instanci *p*-sledeći centar problema; nova implementacija heurističkog VNS algoritma za rešavanje problema *p*-sledećeg centra koja uključuje i adaptaciju *Whitaker* strukture podataka unutar *1-Interchange* lokalne procedure pretraživanja; modifikacija algoritma koja potencijalno prepoznaje egzaktna rešenja *p*-sledeći centar problema; nova najbolja poznata rešenja *OR-Library* instanci problema *p*-sledećeg centra; egzaktna rešenja za 22 od ukupno 40 instanci *p*-sledeći centar problema iz *OR-Library* test skupa podataka.
- Poglavlje 3: Svojstvo 3.1 koje pokazuje da su teorijske karakteristike koje važe za *p*-centar problem [28] primenljive i na problem *p*-drugog centra; Svojstvo 3.2 koje omogućava egzaktno rešavanje pojedinih instanci *p*-drugi centar problema; u literaturi premijerni VNS algoritam za rešavanje *p*-drugi centar problema koji uključuje efikasnu, teorijski zasnovanu, proceduru filtriranja potencijalnih rešenja; modifikacija algoritma koja može prepoznati egzaktna rešenja pojedinih instanci *p*-drugi centar problema; u literaturi prva rešenja *OR-Library* instanci problema *p*-drugog centra; egzaktna rešenja za 12 od ukupno 40 instanci *p*-drugi centar problema iz *OR-Library* test skupa podataka.

- Poglavlje 4: Svojstvo 4.1 koje generalizuje Svojstva 2.5 i 3.1 na opštije probleme *p*-*α*-sledećih i *p*-*α*-najbližih centara; u literaturi prvi VNS algoritmi za rešavanje problema *p*-*α*-sledećih i *p*-*α*-najbližih centara koji uključuju generičku proceduru filtriranja potencijalnih rešenja; u literaturi premijerna rešenja instanci problema *p*-*α*-sledećih i *p*-*α*-najbližih centara predstavljenih *OR-Library* test skupom podataka.
- Poglavlje 5: Definicije *p*-sledeći medijan i *p*-*α*-sledeći medijan problema; Svojstva 5.1, 5.2, 5.3, 5.4 i 5.5 koja daju teorijsku osnovu za efikasno rešavanje *p*-sledeći medijan i *p*-*α*-sledeći medijan problema; implementacija heurističkog VNS algoritma za rešavanje problema *p*-sledećeg medijana koja uključuje i adaptaciju *Whitaker* strukture podataka unutar *1-Interchange* lokalne procedure pretraživanja; modifikacija VNS algoritma iz Poglavlja 4 u cilju rešavanja *p*-*α*-sledeći medijan problema; prva rešenja instanci *p*-sledeći medijan i *p*-*α*-sledeći medijan problema predstavljenih *OR-Library* test skupom podataka.

U disertaciji definisani problemi i predloženi algoritmi za rešavanje *p*-problema su rezultat saradnje na izradi više radova sa profesorima dr Draganom Uroševićem, dr Nenadom Mladenovićem i dr Racom Todosijevićem.

Autor se ovom prilikom zahvaljuje dr Draganu Uroševiću na preuzimanju mentorstva, kao i ostalim članovima komisija dr Filipu Mariću, dr Milanki Gardašević Filipović i dr Ireni Jovanović na ukazanom poverenju i pomoći tokom odbrane kvalifikacionog ispita i disertacije, dr Nenadu Mladenoviću i dr Raci Todosijeviću na svesrdnoj pomoći pruženoj tokom izrade disertacije, i naposletku upravi Računarskog fakulteta, sekretaru Dragani Petrović i dekanima dr Stevanu Milinkoviću i dr Bojani Dimić Surli na ukazanom poverenju, podršci i razumevanju.

Sadržaj

1. Uvod

Problem *p*-centra, predstavljen davne 1965. godine, inspirisao je pojavu mnogih drugih problema koji uključuju optimalan raspored i dodelu centara. Problemi kao što su *p*-centar, "*capacitated*" *p*-centar, "*conditional*" *p*-centar, *p*-medijan i dr. su već dugo poznati i proučavani. I pored toga, problem eventualnog otkaza dodeljenog centra dugo nije privlačio pažnju istraživača.

Albareda-Sambola i sar. [1] su tek 2015. godine predstavili problem i inicijalne matematičke modele rešenja problema otkaza dodeljenog centra u okviru *p*-centar problema. Novi problem je nazvan *p*-sledeći centar i uključivao je dodelu ne samo primarnog već i zamenskog centra. Rad Albarede-Sambole i sar. [1] je motivisao istraživače da obrate pažnju i na ovaj tip problema, tako da je u poslednje vreme, a naročito nakon pojave masovnih migracija i COVID epidemije, predstavljeno više radova i algoritama koji rešavaju i ovaj tip problema, tj. mogućnost otkaza korisniku dodeljenog centra.

Poznato je da su *p*-centar, a i ostali iz njega izvedeni problemi, NP-teški problemi čije egzaktno rešenje zahteva velike resurse. Sa porastom veličine ovih problema (veliki broj centara i korisnika), egzaktni matematički modeli i algoritmi postaju neprimenljivi zbog ograničenosti memorijskih i procesorskih resursa. U tim slučajevima pribegava se heurističkim rešenjima koja vraćaju optimalna ili približno optimalna rešenja. Ova disertacija posvećuje pažnju upravo takvim rešenjima zasnovanim na poznatom metodu promenljivih okolina kao generičkom okviru za izgradnju heurističkih algoritama pretrage. Algoritmi predstavljeni u disertaciji pokušavaju da reše nekoliko različitih *p*-problema nad skupom čvorova grafa koji uključuju i rešenje eventualnog otkaza centara. Predstavljeni problemi i algoritmi za njihovo rešavanje su izvedeni iz *p*-centar i *p*-medijan problema i njihovih rešenja. Stoga, uvodno poglavlje se sastoji iz dva odeljka. U Odeljku 1 su definisani *p*-centar i *p*-medijan problemi i pomenuti postojeći algoritmi promenljivih okolina za njihovo rešavanje. Odeljak 2 sadrži opis generičkog algoritma promenljivih okolina.

1.1. Problemi *p*-centra i *p*-medijana nad skupom čvorova grafa

Problemi izbora p centara, tzv. *p*-problemi, definišu strukturalni odnos svojih gradivnih elemenata, nazvanih korisnici i centri. Centri su entiteti koji pružaju neku funkcionalnu uslugu korisnicima. U osnovi diskretnih *p*-problema je locirati p od mogućih n centara tako da se optimalno zadovolji opšti kriterijum definisan na osnovu neke relacije između korisnika i jednog ili više centara.

U principu, problemi kojima se bavi ova disertacija su najčešće izvedeni iz *p*-centar problema. Problem *p*-centra (pCP) je predstavljen 1965. godine [14] kao diskretan optimizacioni problem identifikacije lokacija p centara na takav način da se minimizuje najveće rastojanje između korisnika i njemu dodeljenog, tj. najbližeg centra. Primera radi, ovim problemom se može modelovati određivanje lokacija p ambulanti (centara) koje će opsluživati n naselja (korisnika) sa ciljem da rastojanje najudaljenijeg naselja od njemu dodeljene ambulante bude minimalno. Problem *p*-centra modeluje i mnoge druge probleme u realnom svetu, kao što su: određivanja lokacija za izgradnju bolnica i prihvatnih centara, postavljanje benzinskih pumpi, trafo stanica, stanica javnog prevoza itd.

Problem *p*-centra se formalno definiše nad neusmerenim težinskim grafom $G = (V, E)$, gde je V skup svih čvorova, tj. lokacija centara i korisnika, a E skup svih grana grafa koje povezuju te lokacije. Težine grana odgovaraju rastojanju između svojih krajeva. Neka je $d(i, j)$ najkraće rastojanje između čvorova i i j u grafu G. Rešenje problema *p*-centra je skup čvorova $P \subset V$, kardinalnosti p, koji sadrži centre, tako da je maksimalno rastojanje korisnika do dodeljenog centra minimizovano:

$$pCP(V) = \min_{\substack{P \subset V \\ |P|=p}} \max_{i \in V} \left\{ \min_{j \in P} d(i,j) \right\}. \tag{1.1}$$

Problem *p*-centra je NP-težak problem [19], ali je već dugo poznat i proučavan tako da postoji mnogo članaka i algoritama koji se bave ovim problemom. U literaturi postoji dosta, kako egzaktnih matematičkih

modela tako i heurističkih algoritama koji uspešno pronalaze rešenja problema *p*-centra. Egzaktni metodi, kao što su [4, 5, 7, 9, 18], se bave manjim problemima, dok se rešenja većih instanci traže heurističkim algoritmima. Najpoznatiji heuristički algoritmi su predstavljeni u radovima [8, 16, 28, 30].

Još jedan problem koji je motivisao izradu ovog rada je *p*-medijan problem. Problem *p*-medijana (pMP) je diskretni optimizacioni problem identifikacije p od potencijalnih n lokacija za postavljanje centara na takav način da se minimizuje suma rastojanja između svih korisnika i njima dodeljenih, tj. najbližih centara. Posmatran iz ugla logistike, *p*-medijan problem predstavlja određivanje p od mogućih n lokacija uslužnih centara sa ciljem minimizacije ukupne cene transporta za zadovoljenje potreba svih n korisnika, svakog opsluženog iz njemu najbližeg centra. Ovako definisan problem, ističe značaj problema u odnosu na *p*-centar problem. Za razliku od problema *p*-centra, rešenje *p*-medijan problema postavlja akcenat na snabdevanju svih korisnika, npr. minimizuje ukupnu cenu transporta toplotne energije od toplana do svih korisnika.

Nad istim grafom G kao i u slučaju problema *p*-centra, *p*-medijan problem se formalno definiše kao:

$$pMP(V) = \min_{\substack{P \subset V \\ |P|=p}} \sum_{i \in V} \left\{ \min_{j \in P} d(i,j) \right\}. \tag{1.2}$$

Efikasan heuristički algoritam za rešavanje *p*-medijan problema, kao implementacija generičkog okvira promenljivih okolina za izgradnju algoritama pretrage, predstavljen je u radu [15].

Može se reći da su *p*-centar i *p*-medijan problemi uspešno rešeni, ali postavlja se pitanje šta raditi u slučaju ispada dodeljenog centra. Delimično i kao odgovor na ovo pitanje, definisani su drugi problemi, kao što su "*capacitated p-center*", gde centri imaju ograničen kapacitet, ili "*conditional p-center*", "*fault tolerant*"... "*Conditional p-center*" problem podrazumeva da već postoji q centara i da se skup centara po potrebi može proširiti sa još dodatnih p centara tako da je minimizovano maksimalno rastojanje među korisnicima i dodeljenim centrima uzevši u obzir sve $q + p$ centre. "*Fault tolerant*" je generalizacija *p*-centar problema pri čemu je svakom korisniku dodeljeno više centara.

Motiv za izradu ovog rada upravo predstavlja mogućnost otkaza dodeljenog centra usled raznih humanitarnih i prirodnih katastrofa kao što su zemljotresi, požari, poplave ili pak masovne migracije ljudi usled ratnih sukoba i sl. Dodeljeni centar može postati preopterećen ili jednostavno havarisan i usled toga onemogućen da dalje pruža usluge svojim korisnicima. Problemi *p*-centra i *p*-medijana u tom slučaju ne nude rešenje. Potrebno je obratiti se zamenskom, tj. sledećem najbližem centru. Postoje dve klase ovakvih problema: prva kada se korisnik odmah obraća zamenskom centru i druga kada najpre poseti primarni pa tek onda zamenski centar. U zavisnosti od samog problema primenljivija je jedna ili druga klasa. Za očekivati je da se unapred zna za preopterećenje bolnice pa se pacijenti odmah obraćaju drugoj najbližoj ambulanti za pomoć. Međutim, u slučaju infrastrukturnih problema kao što su transport toplotne ili električne energije, najpre se "posećuje" primarna pa tek onda zamenska toplana ili trafo stanica. Za očekivati je da su električni vodovi instalirani tako da povezuju korisnika sa najbližom trafo stanicom pa tek onda trafo stanicu sa sledećom trafo stanicom.

Trebalo bi još ponoviti da su svi *p*-problemi, definisani i obrađeni u disertaciji, posmatrani kao problemi ograničeni na izbor centara iz unapred definisanog skupa čvorova (*eng. vertex-restricted*). Takođe, u predloženim algoritmima ne mora nužno važiti pretpostavka o nejednakosti trougla. Rastojanje između čvorova grafa, odnosno korisnika i centara, u opštem slučaju se predstavlja proizvoljnom merom težina grana, a težina svih čvorova je podrazumevano 1.

1.2. Metod promenljivih okolina

Metod promenljivih okolina (*eng. Variable Neighborhood Search*, VNS) je predstavljen 1997. godine u radu [27] kao generički okvir za izgradnju heurističkih algoritama zasnovanih na sistematičnoj promeni struktura susedstva rešenja tokom pretrage za optimalnim rešenjem problema. Od tada, mnogi algoritmi koji su izgrađeni na ovom okviru su uspešno primenjeni kao rešenja mnogih problema u oblastima operacionih istraživanja i kombinatornoj optimizaciji. VNS heuristike su implementirane kao rešenja *p*-centar problema

(*eng. p-center problem*) [28], problema verovatnog *p*-centra (*eng. probabilistic p-center problem*) [22], problema *p*-sledećeg centra (*eng. p-next center problem*) [21, 31]...

Bazični metod promenljivih okolina uključuje dve glavne faze: lokalnu pretragu tekućeg rešenja i fazu promene tekućeg rešenja (*eng. shaking phase*), gde se novo rešenje za pretragu metodom slučajnog uzorka bira u *k*-toj okolini tekućeg rešenja. Tokom lokalne pretrage, tekuće rešenje se pretražuje sa ciljem pronalaska lokalnog optimuma, dok u fazi promene tekućeg rešenja dolazi do "skoka" u *k*-to susedstvo rešenja da bi se izbegla potencijalna zamka dubokih lokalnih optimuma pretrage. Drugim rečima, bazična VNS implementacija tokom faze lokalne pretrage insistira na intenzifikaciji posmatranog rešenja, a skokom u udaljena rešenja postiže diversifikaciju procesa pretrage. Svrha intenzifikacije je intenzivno pretraživanje tekućeg rešenja sa ciljem maksimalnog poboljšanja rešenja, tj. identifikacije optimalnog rešenja u bliskim okolinama tekućeg rešenja. Nedostatak lokalne pretrage je taj što se većina vremena utroši na pretragu ograničenog skupa rešenja, što može dovesti do toga da najzanimljiviji delovi pretraživačkog prostora ostanu neistraženi, a samim tim i dobijena rešenja ne budu dovoljno kvalitetna. Diversifikacija pretrage rešava ovaj problem tako što proces pretrage navodi u regione koji prethodno nisu bili istraženi, tj. daje mogućnost pretrage udaljenih rešenja i na taj način proširuje skup potencijalnih rešenja novim rešenjima koja mogu biti značajno različita od tekućeg rešenja.

U nastavku teksta, predstavljeni algoritmi za rešavanje *p*-problema zasnovani su na implementaciji metoda promene susedstva. Stoga, najpre je data jednostavna implementacija VNS heuristike. U okviru bazične VNS implementacije, rešenje je prezentovano kao skup $P = \{p_1, \dots, p_n\}, |P| = p, P \subset V$, gde p_i predstavlja *i*-ti element tekućeg rešenja *P*, a *V* predstavlja skup svih potencijalnih elemenata rešenja. Shodno tome, *k*-to susedstvo rešenja *P* se definiše kao:

$$N_k(P) = \{P' |\ P' \subset V, |P'| = p, |P' \cap P| = p - k\}, k = 1, 2, \dots, p, \tag{1.3}$$

gde je P' rešenje iz skupa rešenja $N_k(P)$. Drugačije rečeno, $N_k(P)$ predstavlja skup svih rešenja dobijenih zamenom tačno *k* elemenata rešenja *P* novim elementima koji nisu inicijalno uključeni u *P*.

Algoritam 1.1: Bazični VNS algoritam

```
VNS (k_max)
P ← Initial_Solution()
k ← 1
    While k ≤ k_max
        P' ← Shake(P, N_k)
        P'' ← LocalSearch(P')
        If P'' is better than (or equal) P
            P ← P''
            k ← 1
        Else
            k ← k + 1
        End If
    End While
Return P
```

Na osnovu prethodnih definicija, Algoritam 1.1 prezentuje pseudokod bazične VNS metaheuristike za rešavanje *p*-problema. Algoritam počinje generisanjem slučajnog inicijalnog rešenja koje postaje tekuće optimalno rešenje *P*. Nakon toga se startuje glavna VNS petlja. Ona se sastoji od naizmenične procedure promene susedstva i izvršenja procedure lokalne pretrage u cilju unapređenja tekućeg optimalnog rešenja. Faza promene susedstava teži diversifikaciji procesa pretrage generisanjem slučajnog rešenja iz $N_k(P)$ susedstva. Nivo diversifikacije je kontrolisan parametrom k_{max} koji definiše najveću moguću *k* vrednost. Inicijalno, *k* je postavljeno na 1, što znači da se rešenje iz $N_1(P)$ koristi da diversifikuje rešenje *P*. U nastavku, svaki put kada dođe do promene tekućeg optimalnog rešenja, *k* se resetuje na 1. U suprotnom, vrednost *k* se uvećava za 1, što znači da će veći skok biti napravljen "*shaking*" procedurom *Shake*(P, N_k), a sa ciljem da se izbegne potencijalna zamka dubokog lokalnog optimuma. Do promene tekućeg optimalnog rešenje dolazi kada se procedurom lokalne pretrage pronađe bolje (ili jednako) rešenje. Tada, novo rešenje postaje tekuće optimalno rešenje. U

suprotnom se novo rešenje odbacuje. U fazi lokalne pretrage, bazični VNS implementira pretragu u okviru N_1 susedstva tekućeg rešenja.

VNS je u osnovu metaheuristika koja predstavlja samo nacrt za izgradnju algoritama, tako da se procedura lokalne pretrage može implementirati na različite načine. Npr., u radu [21] koji prezentuje VNS rešenje problema *p*-sledećeg centra, procedura lokalne pretrage je zasnovana na "*first improvement search*" strategiji, tj. čim se u N_1 okolini tekućeg rešenja pronađe bolje rešenje, to rešenje postaje tekuće i pretraga se nastavlja u N_1 okolini novog rešenja. Sa druge strane, u ovoj disertaciji implementirani su VNS algoritmi koji koriste "*best improvement search*" strategiju, tako što se bira najbolje moguće rešenje iz N_1 okoline tekućeg rešenja. Ukoliko je takvo rešenje bolje od tekućeg, postavlja se za novo tekuće i pretraga se nastavlja dalje. Algoritam se izvršava sve dok se ne dostigne maksimalna dozvoljena k vrednost, tj. k_{max}. Kao rezultat, vraća se tekuće optimalno rešenje, tj. najbolje rešenje pronađeno tokom procesa pretrage.

Procedura lokalne pretrage, u N_1 okolini tekućeg rešenja, ispituje $p * (n - p)$ mogućih rešenja u svakoj iteraciji. Izvesno je da mnoga od njih ne unapređuju tekuće rešenje. Stoga, efektivno bi bilo eliminisati ta rešenja iz procesa pretrage. To bi omogućilo efikasan skok u bolje rešenje i na taj način značajno ubrzalo proces lokalne pretrage. U kontekstu *p*-centar problema, takvo ubrzanje je implementirano u radu [28] i nazvano "*1-Interchange* ili *Vertex Substitution (VS) heuristic*". U ovoj disertaciji, oslanjajući se na prethodnu ideju, implementiran je efikasan metod zamene čvorova (VS) za rešavanje skupa srodnih *p*-problema koji tretiraju slučajeve kada dođe do ispada primarnih centara. Cilj je pokazati da se ideje primenjene u implementaciji VNS algoritma za rešavanje problema *p*-centra mogu primeniti i na složenije, tj. *p*-probleme izvedene iz *p*-centar problema. U nastavku, predstavljena je proširena VNS implementacija koja se izvršava sve dok ne istekne predefinisano vremensko ograničenje t_{max} i uključuje sofisticiranu "*shaking*" i proceduru lokalne pretrage koje filtriraju potencijalna rešenja pre procesa pretrage u kontekstu pomenutih *p*-problema.

2. Problem *p*-sledećeg centra

Problem *p*-sledećeg centra predstavlja identifikaciju lokacija *p* centara sa ciljem minimizacije maksimalnog rastojanja među *n* korisnika ($p \leq n$) do najbližeg centra plus rastojanja između tog centra i njemu najbližeg centra. U ovom poglavlju, predložen je novi algoritam zasnovan na metodu promenljivih okolina za rešavanje *p*-sledeći centar problema. Algoritam, pored procedure skoka u proizvoljno rešenje iz odgovarajuće okoline tekućeg rešenja, koristi i sofisticiranu proceduru lokalne pretrage oslonjenu na specifične strukture podataka za predstavljanje i konstruisanje rešenja. Implementacija uključuje filtriranje centara koji mogu ući u rešenje tako da se u razmatranje uzimaju samo oni koji potencijalno smanjuju vrednost funkcije cilja. Isti pristup je primenjen i na klasičan *p*-centar problem u rešenju [28]. Ovde je pokazano da se već poznata svojstva i efikasna implementacija VNS heuristike mogu primeniti i na rešavanje novog problema *p*-sledećeg centra. Efikasnost predložene implementacije je procenjena testiranjem nad u literaturi poznatim *OR-Library* skupom podataka kao i nad novo-kreiranim većim instancama problema sa 1000, 1500, 2000 i 2500 čvorova i instancama do 1000 čvorova sa različitim gustinama. Dobijeni rezultati su potvrdili efektivnost i efikasnost predloženog algoritma, tako da se može reći da su teorijska razmatranja i implementacija korišćena u rešavanju *p*-centar problema primenljiva i kao rešenje problema *p*-sledećeg centra.

2.1. Uvod

Problem *p*-sledećeg centra (*eng. p-next center problem*, pNCP) je relativno nov problem u oblasti operacionih istraživanja. Definiše se nad neusmerenim težinskim grafom $G = (V, E)$, gde je $V = \{v_1, v_2, \dots, v_n\}, |V| = n$ skup čvorova grafa koji predstavljaju lokacije potencijalnih centara kao i korisnika, a $E = \{(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ skup svih grana grafa. Težina svake grane (v_i, v_j) odgovara rastojanju između njenih čvorova, a $d(v_i, v_j)$ predstavlja najkraće rastojanje između čvorova v_i i v_j u grafu G. Rešenje pNCP je identifikovati *p* centara iz skupa V, tako da je maksimalna udaljenost korisnika do dodeljenog zamenskog centra minimizovana. Udaljenost korisnika do dodeljenog zamenskog centra se računa kao suma rastojanja između korisnika i najbližeg centra i rastojanja između tog centra i (njemu najbližeg) zamenskog centra. Formalno, vrednost funkcije cilja za dati skup $P \subset V$ koji čine *p* centara se računa kao:

$$f(P,V) = \max_{v_i \in V} \left\{ \min_{p_j \in P} d(v_i, p_j) + \min_{\substack{p_k \in P \\ p_k \neq p_j' \in \left\{ \arg\min_{p_j \in P} d(v_i, p_j) \right\}}} d(p_j', p_k) \right\}, \tag{2.1}$$

dok rešenje problema *p*-sledećeg centra predstavlja skup centara $P \subset V$, kardinalnosti *p*, takav da je vrednost funkcije cilja minimizovana:

$$pNCP(V) = \min_{\substack{P \subset V \\ |P| = p}} f(P,V). \tag{2.2}$$

Problem *p*-sledećeg centra je predstavljen od strane Albarede-Sambole i sar. u radu [1] iz 2015. godine kao proširenje *p*-centar problema sa ciljem rukovanja ispadom primarnog centra usled humanitarnih i drugih katastrofa. Prema tome, rešenje *p*-sledeći centar problema daje odgovor na neočekivane incidente koji mogu prouzrokovati kvar centra. Na primer, zemljotresi mogu onesposobiti najbližu ambulantu, ili recimo tokom masovnih migracija, često dolazi do preopterećenja prihvatnih centara. Tokom pandemija kao što je COVID-19, privremene bolnice i karantini se veoma brzo pune tako da ubrzo ostaju bez kapaciteta da prihvataju nove pacijente. Stoga, važno je imati rezervnu strategiju, tj. zamenski ili sledeći-najbliži centar spreman da prihvati korisnike u slučaju otkaza primarnog centra. Albareda-Sambola i sar. su u radu [1] primarni centar nazvali *reference*, a zamenski *backup* centar.

Iako je literatura bogata različitim pristupima za rešavanje problema *p*-centra [8, 28], pNCP kao novi problem, tek je u skorije vreme počeo da privlači pažnju naučnika. Zajedno sa definicijom problema, Albareda-Sambola i sar. [1] su predložili nekoliko egzaktnih matematičkih (IP) modela za rešavanje pNCP: dva i tri-indeksirane formule koje koriste promenljive putanje i formulaciju zasnovanu na pokrivajućim promenljivama. Autori su pokazali da mogu rešiti primere problema veličine do 50 čvorova u razumnom vremenskom intervalu. Takođe, autori u istom radu daju formalni dokaz da je problem *p*-sledećeg centra NP-težak problem.

Prvi heuristički algoritmi za pNCP su predstavljeni od strane Lopez-Sancheza i sar. 2019. godine [21]. Autori su predložili tri heuristike: GRASP (*Greedy Randomized Adaptive Search Procedure*), VNS i hibridnu verziju koja kombinuje GRASP i VNS. GRASP je zasnovan na metodologiji inicijalno predloženoj od strane Fea i Resenda 1989. godine [10]. Predstavljen je kao multi-start okvir koji se sastoji od faze konstrukcije i faze lokalne pretrage rešenja. U fazi konstrukcije se postepeno generiše rešenje na taj način što se u svakoj iteraciji tekućem rešenju priključuje novi element. Kombinacija "*greediness*" i "*randomness*" pristupa se postiže tako što se element koji će se priključiti rešenju nasumično bira iz prethodno definisane liste kandidata, a lista kandidata se generiše pomoću funkcije koja selektuje kandidate na osnovu hiper-parametra α koji kontroliše stepen "*greediness*" i "*randomness*" kombinacije. Ukoliko je $\alpha = 1$, kandidati se biraju potpuno nasumično, a $\alpha = 0$ determiniše pohlepnu ("*greedy*") funkciju koja kandidate bira na osnovu "*the most promising*" kriterijuma. Druga faza GRASP algoritma je lokalna pretraga koja pokušava da identifikuje lokalni optimum. Implementira jednostavan algoritam pretrage koji posećuje susedna (N_1) rešenja u nasumičnom redosledu i ukoliko naiđe na rešenje koje unapređuje tekuće, prihvata novo kao tekuće rešenje. Lokalna pretraga se ponavlja dokle god je moguće pronaći rešenje bolje od tekućeg. Kompletan algoritam se izvršava dokle god se ne generiše unapred određen broj rešenja. VNS implementira bazičnu VNS metaheuristiku koja je detaljno objašnjena u uvodnom poglavlju. Sastoji se od *shaking* i faze lokalne pretrage koja je implementirana na isti način kao i lokalna pretraga GRASP algoritma. Hibridna verzija algoritma kombinuje GRASP i VNS tako što je faza lokalne pretrage GRASP algoritma zamenjena kompletnim VNS algoritmom. Svi algoritmi iz rada Lopez-Sancheza i sar. [21] su testirani nad istim test setom kao i rešenja iz rada Albarede-Sambole i sar. [1] i ispostavilo se da hibridna verzija vraća dosta bolja rešenja od ostalih, ali da takođe i zahteva mnogo više procesorskog vremena. GRASP i VNS algoritmi su dali rezultate približno istog kvaliteta.

U ovom radu, predlaže se nova VNS heuristika za problem *p*-sledećeg centra. Ona se od prethodne VNS implementacije razlikuje po tome što koristi sofisticiraniju proceduru lokalne pretrage kao i *shaking* proceduru. Sofisticirana implementacija je inspirisana radom Mladenovića i sar. [28] na rešenju *p*-centar problema. Sastoji se od ranog prepoznavanja centara koji neće unaprediti tekuće rešenje. Dakle, pristupa se najpre filtriranju centara tako da se razmatraju samo oni koji potencijalno smanjuju vrednost funkcije cilja. Ideja filtriranja rešenja je već predstavljena u radu Ristića i sar. [31] iz 2021. godine, gde su autori načinili prvi korak u cilju unapređenja bazične VNS implementacije prezentovane u radu [21]. U ovoj disertaciji, načinjen je naredni korak i dodatno unapređena implementacija iz rada [31] uključivanjem novih struktura podataka u cilju ubrzanja procesa lokalne pretrage, kao i prezentovanjem sofisticiranije *shaking* procedure. Može se primetiti da VNS heuristika iz rada Ristića i sar. [31] koristi jednostavnu *shaking* proceduru kao i bazična VNS implementacija iz rada [21], dok nova implementacija uključuje mehanizam filtriranja i u *shaking* proceduru. Štaviše, u disertaciji se pokazuje da se tzv. *Whitaker* struktura podataka, implementirana u okviru (*vertex substitution*) faze lokalne pretrage, prethodno predložena za rešavanje *p*-medijan [15, 26, 32] i *p*-centar [28] problema, takođe može efikasno adaptirati za rešavanje problema *p*-sledećeg centra. *Whitaker* struktura podataka, prvobitno predložena u [32], uključuje pomoćne nizove koji za svaki čvor (tj. svakog korisnika) sadrže najbliži i drugi-najbliži centar. Ideja je pratiti tezu "Manje je više", u literaturi već prisutnu kao filozofski stav koji se prožima kroz radove [3, 6, 12, 24, 29]. Dizajniran je VNS algoritam koji uključuje minimalan broj komponenti u proces pretrage, ali u najefikasnijem maniru. Prateći ovakav filozofski pravac, u ovom radu je pokazano da se strukture iskorišćene za efikasnu VNS implementaciju rešenja *p*-centar problema [28] mogu koristiti i za rešavanje problema *p*-sledećeg centra. Novinu predstavlja teorijska i praktična adaptacija poznatih osobina *p*-centar problema novom problemu *p*-sledećeg centra. Drugim rečima, više teorije vodi ka što jednostavnijem algoritmu. Kao rezultat, algoritam pronalazi rešenja koja značajno prevazilaze *state-of-the-art* rezultate poznate u literaturi. Dodatno, da bi se proverile performanse predloženog pristupa u rešavanju šire klase problema, generisane su veće test instance sa 1000, 1500, 2000 i 2500 čvorova i instance definisane nad grafovima do 1000 čvorova sa različitim gustinama.

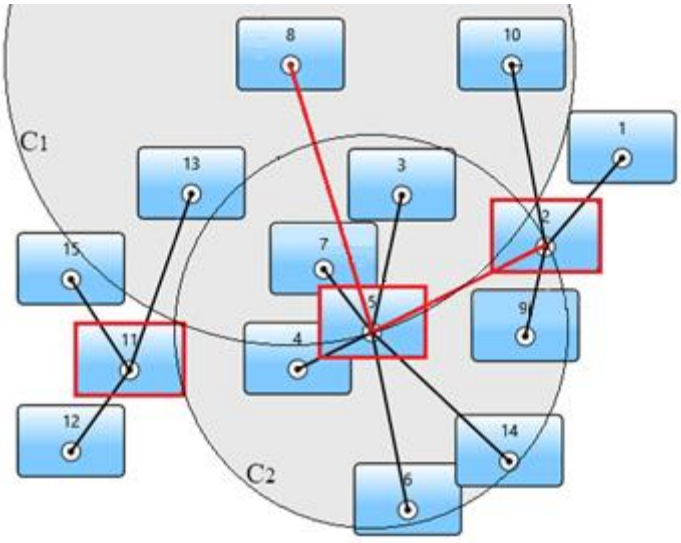
Dobijeni rezultati su nedvosmisleno potvrdili efektivnost i efikasnost predloženog algoritma. Ukratko, doprinos ovog poglavlja disertacije može se sažeti na sledeći način:

1) Predložena je nova VNS heuristika za problem *p*-sledećeg centra.
2) Poznata svojstva koja važe za *p*-centar problem su teorijski i praktično proširena na problem *p*-sledećeg centra. To uključuje i adaptaciju *Whitaker* strukture podataka unutar *1-Interchange* lokalne procedure pretraživanja.
3) Novo-predložena heuristika unapređuje *state-of-the-art* rezultate prezentujući nova najbolja poznata rešenja instanci problema *p*-sledećeg centra za kraće vreme. Pored toga, najlošije i prosečne vrednosti rešenja u 20 izvršenja algoritma su često bile bolje od do sada najboljih u literaturi poznatih rešenja.
4) Predložen je novi referentni test skup većih instanci. Rezultati testiranja su pokazali da se čak i sa tako velikim instancama *p*-sledeći centar problema algoritam može nositi na efektivan i efikasan način.

2.2. Efikasna heuristika zamene čvorova za problem *p*-sledećeg centra

Slično kao i u slučaju *p*-centar problema [28], svako rešenje problema *p*-sledećeg centra, predstavljeno skupom centara $P = \{p_1, p_2, \dots, p_p\}$, particioniše korisnike u p disjunktnih podskupova: $S_1, S_2, \dots, S_p$. Svaki od skupova S_i sadrži korisnike alocirane istom, tj. *i*-tom, *reference* centru. Rastojanje između *i*-tog *reference* centra i njegovog najudaljenijeg korisnika plus udaljenost tog *reference* centra od njemu najbližeg centra, tj. *backup* centra, predstavlja radijus skupa S_i. Rastojanje je nazvano radijus po analogiji sa terminologijom korišćenom za *p*-centar problem [28]. Neka je radijus skupa S_i označen kao $r(S_i)$, tada je vrednost funkcije cilja $f(P)$ u rešenju P određena radijusom "kritičnog" skupa, tj. skupa korisnika sa najvećim radijusom $f(P) = max_{i=1,\dots,p}\{r(S_i)\}$. Centar i korisnik koji odgovaraju vrednosti funkcije cilja su takođe nazvani kritičnim centrom p_c i kritičnim korisnikom u_c.

Na osnovu prethodnog, da bi se unapredilo rešenje, najveći radijus mora biti smanjen. To se može postići otvaranjem novog centra koji smanjuje rastojanje od kritičnog korisnika do *reference* centra i/ili smanjuje rastojanje između kritičnog centra i njegovog *backup* centra. U oba slučaja, trebalo bi otvoriti novi centar, a jedan od postojećih centara zatvoriti. Slučajevi su ilustrovani na Slici 2.1. Tekuće rešenje problema *p*-sledećeg centra sa $n = 15$ korisnika i $p = 3$ centra grupiše centre u tri disjunktna podskupa S_2, S_5 i S_{11}, reprezentovana *reference* centrima 2, 5 i 11. Kritični korisnik je čvor 8, dok je kritični centar čvor 5. Neka $d(u, v)$ predstavlja najkraće rastojanje između čvorova u i v. Da bi se unapredilo tekuće rešenje, radijus $d(8, 5) + d(5, 2)$ mora biti smanjen. Stoga, novi centri koji potencijalno redukuju vrednost funkcije cilja se nalaze u krugu C_1 čiji centar je čvor 8, a poluprečnik $d(8, 5)$, ili u krugu C_2 poluprečnika $d(5, 2)$ sa centrom u čvoru 5. Krug C_1 sadrži lokacije potencijalnih centara koji mogu postati novi *reference* centar, a krug C_2 centara koji mogu unaprediti poziciju *backup* centra kritičnog korisnika.



Sl. 2.1. Primer problema *p*-sledećeg centra za $n = 15$ korisnika i $p = 3$ centra; tekuće rešenje je $P = \{2, 5, 11\}$, a kritični korisnik $u_c = 8$

Prethodna diskusija sugeriše da postoje centri čijim otvaranjem tekuće rešenje izvesno neće biti unapređeno, tako da nema potrebe za evaluacijom svih mogućih rešenja iz $N_1(P)$ okoline u fazi zamene čvorova. Stoga, analogno kao u [28], u disertaciji se predlaže efikasan način da se neobećavajuća rešenja odbace i da se stavi fokus samo na ona koja potencijalno mogu unaprediti tekuće rešenje. U tu svrhu, *p*-sledeći centar problemu je adaptirano rešenje predstavljeno na način koji sledi.

Sem tekućeg rešenja $P = \{x_1, x_2, \dots, x_p\}$ koje čini skup prvih p centara niza x_{cur}, iskorišćeni su nizovi $c1_P$ i $c2_P$ da bi za svakog od korisnika čuvali najbliži i drugi-najbliži centar iz skupa P. Po konvenciji $c1_P(v_i) = argmin\{d(v_i, x_j) \mid x_j \in P\}$ za $v_i \in V \backslash P$, a $c1_P(x_i) = argmin\{d(x_i, x_j) \mid x_j \in P, x_j \neq x_i\}$ za $x_i \in P$. Slično, $c2_P(v_i) = argmin\{d(v_i, x_j) \mid x_j \in P, x_j \neq c1_P(v_i)\}$ za $v_i \in V \backslash P$, a $c2_P(x_i) = argmin\{d(x_i, x_j) \mid x_j \in P, x_j \neq x_i, x_j \neq c1_P(x_i)\}$ za $x_i \in P$.

Koristeći predstavljenu notaciju (tj. pojednostavljeno $c1 = c1_P$, $c2 = c2_P$), Algoritam 2.1 opisuje proceduru koja verifikuje da li je centar c_{in} obećavajući ili ne. Algoritam eksploatiše činjenicu da je centar obećavajući ukoliko se njegovim otvaranjem radijus kritičnog centra smanjuje. Postoje tri različita scenarija.

- Ukoliko su centri $c1(u_c)$ i $c2(u_c)$ ekvidistantni u odnosu na kritičnog korisnika u_c, novi centar c_{in} može učiniti centar $c2(u_c)$ boljim izborom za *reference* centar od $c1(u_c)$. To se dešava ukoliko $c2(u_c)$ sada ima bliži *backup* centar c_{in}, odnosno ako je $min\{d(c1(u_c), c1(c1(u_c))), d(c1(u_c), c_{in})\} > d(c2(u_c), c_{in})$.
- Ukoliko centar c_{in} nije udaljeniji od korisnika u_c u odnosu na $c1(u_c)$, tada radijus potencijalno može biti smanjen odabirom c_{in} za novi *reference* centar.
- Ukoliko je c_{in} bliži kritičnom centru u odnosu na *backup* centar, tada radijus može biti smanjen izborom c_{in} za novi *backup* centar.

Ovi scenariji su pokriveni pseudokodom procedure *ExistsRelaxedDistance* (Algoritam 2.1), koji pokazuje da se verifikacija da li je novi centar c_{in} obećavajući ili ne može postići uz konstantnu vremensku složenost koristeći nizove *c1* i *c2*.

Algoritam 2.1: Provera da li je moguće pronaći potencijalno bolje rešenje nakon otvaranja novog centra c_{in}

ExistsRelaxedDistance(c1, c2, u_c, c_{in}, f_{cur})
$p_c = c1(u_c)$
If $d(u_c, c1(u_c)) = d(u_c, c2(u_c))$ *and*
$min(d(c1(u_c), c1(c1(u_c))), d(c1(u_c), c_{in})) > min(d(c2(u_c), c1(c2(u_c))), d(c2(u_c), c_{in}))$
$p_c = c2(u_c)$
End If

*****the new center c_{in} is not farther from the user u_c than the reference center*****
If $d(u_c, c_{in}) \leq d(u_c, p_c)$ *and* $d(u_c, c_{in}) + d(c_{in}, c1(c_{in})) < f_{cur}$
Return *True*
End If

*****new center c_{in} is closer to the reference center $c1(u_c)$*****
If $d(u_c, c_{in}) \geq d(u_c, p_c)$ *and* $d(u_c, p_c) + min(d(p_c, c1(p_c)), d(p_c, c_{in})) < f_{cur}$
Return *True*
End If

Return *False*

Algoritam 2.2: Nacrt procedure zamene čvorova tokom faze lokalne pretrage za *p*-sledeći centar problem

LocalSearchVertexSubstitution(x_{cur}, c1, c2, u_c, f_{cur})
Main loop:
While *True*
f' ← ∞
c'$_{in}$ ← null; c'$_{out}$ ← null
P←{$x_{cur}(1)$, $x_{cur}(2)$, …, $x_{cur}(p)$}
For Each $c_{in} \in V \setminus P$
If ExistsRelaxedDistance(c1, c2, u_c, c_{in}, f_{cur})
*****Calculate z and r values*****
*z ← Calulate_z (**x_{cur}, c1, c2, u_c, c_{in}**)*
*r ← Calulate_r (**x_{cur}, c1, c2, u_c, c_{in}**)*
*****determine the best center to close and resulting objective function value*****

$$c_{out} = \underset{p_i \in P}{arg\,min} \left\{ max \left\{ z(p_i), \underset{\substack{p_j \in P \cup \{c_{in}\} \\ p_j \neq p_i \\ c1(p_j) \neq p_i}}{max} \{ r(p_j, c1(p_j)) \} \right\} \right\}$$

$$f = \underset{p_i \in P}{min} \left\{ max \left\{ z(p_i), \underset{\substack{p_j \in P \cup \{c_{in}\} \\ p_j \neq p_i \\ c1(p_j) \neq p_i}}{max} \{ r(p_j, c1(p_j)) \} \right\} \right\}$$

If *f < f'*
f' ← f
c'$_{in}$ ← c_{in}
c'$_{out}$ ← c_{out}
End If
End If
End For Each
If *$f_{cur} \leq f'$*
There is no improvement in the neighborhood:
Break Main loop
End If
Update(x_{cur}, c1, c2, u_c, f_{cur},** f', c'$_{in}$, c'$_{out}$**)
End While
Return x_{cur}, c1, c2, u_c, f_{cur}

Nakon što je potencijalno obećavajući centar c_{in} otvoren, potrebno je optimalno identifikovati centar c_{out} koji bi trebalo zatvoriti. Slično kao u [28], razvijena je nova procedura sa ciljem da nakon zatvaranja centra vrednost funkcije cilja $f(P \cup \{c_{in}\} \setminus \{c_{out}\})$ bude što manja. Za tu namenu, uvedene su dve dodatne strukture.

- $r(p_i, p_j)$ – predstavlja najveću vrednost funkcije cilja svih korisnika kojima je par centara (p_i, p_j) dodeljen kao *reference* i *backup* centar u rešenju $P' = P \cup \{c_{in}\}$, pri čemu $p_i, p_j \in P'$. Bitno je primetiti da se vremenska složenost izračunavanja ovih vrednosti redukuje na taj način što se ne računaju vrednosti za sve parove centara (p_i, p_j), već samo za $(p_i, c1_{P'}(p_i))$, gde $c1_{P'}(p_i)$ odgovara najbližem centru centra p_i u rešenju P'. Dakle, u okviru r strukture evidentiraju se radijusi centara iz rešenja P', tako da struktura sadrži vrednosti funkcije cilja u rešenju $P \cup \{c_{in}\}$.
- $z(p_i)$ – predstavlja najveću vrednost funkcije cilja u rešenju $P_i = P \cup \{c_{in}\} \setminus \{p_i\}$, svih korisnika kojima je centar $p_i \in P$ dodeljen kao *reference* ili *backup* centar u rešenju $P' = P \cup \{c_{in}\}$. Naime, ako se centar $p_i \in P$ zatvori, korisnicima koji ostaju bez centra u rešenju P_i mora se dodeliti novi par centara. Pretpostavka je da novi centar c_{in} neće biti zatvoren.

Na osnovu vrednosti iz r i z struktura, vrednost funkcije cilja u rešenju $P_i = P \cup \{c_{in}\} \setminus \{p_i\}$ se određuje kao:

$$f(P \cup \{c_{in}\} \setminus \{p_i\}) = max \left\{ \begin{array}{c} z(p_i), \underset{\substack{p_j \in P \cup \{c_{in}\} \\ p_j \neq p_i \\ c1_{P'}(p_j) \neq p_i}}{max} \{r(p_j, c1_{P'}(p_j))\} \end{array} \right\}. \quad (2.3)$$

Na osnovu prethodnog, optimalan centar za zatvaranje odgovara centru $p_i \in P$ koji zadovoljava:

$$c_{out} = \underset{p_i \in P}{arg\ min}\{f(P \cup \{c_{in}\} \setminus \{p_i\})\} = \underset{p_i \in P}{arg\ min} \left\{ max \left\{ \begin{array}{c} z(p_i), \underset{\substack{p_j \in P \cup \{c_{in}\} \\ p_j \neq p_i \\ c1_{P'}(p_j) \neq p_i}}{max} \{r(p_j, c1_{P'}(p_j))\} \end{array} \right\} \right\}. \quad (2.4)$$

Nacrt algoritma na najvišem nivou apstrakcije koji ispituje samo obećavajuća rešenja u okolini $N_1(P)$ i koristi r i z strukture da identifikuje najbolji centar za brisanje je predstavljen Algoritmom 2.2. Algoritam je nazvan *LocalSearchVertexSubstitution*. Na ulazu, Algoritam 2.2 zahteva tekuće rešenje x_{cur}, kritičnog korisnika u_c kao i nizove centara *c1* i *c2*. Vrednost funkcije cilja ulaznog rešenja je prosleđena preko parametra f_{cur}. Svaki put kada se pronađe bolje rešenje vrednosti parametara se ažuriraju, te stoga na izlazu procedura vraća najbolje rešenje pronađeno tokom pretrage.

U nastavku se razmatra ispravnost i kompleksnost algoritma. Najpre, biće objašnjeno kako strukture r i z mogu biti efikasno generisane na osnovu nizova $c1_P$ i $c2_P$.

Svojstvo 2.1. Koristeći nizove $c1_P$ i $c2_P$, r vrednosti u Algoritmu 2.2 mogu biti izračunate za $O(n)$ vremensku kompleksnost.

Dokaz. Razlikuju se tri slučaja:

1) Otvaranje novog centra c_{in} dovodi do toga da nekim korisnicima iz skupa V, centar c_{in} bude dodeljen kao novi *reference* centar na jedan od sledećih načina:

- novi centar c_{in} je bliži korisniku $v_i \in V$ od prethodnog *reference* centra $c1_P(v_i)$,
- centri c_{in} i $c1_P(v_i)$ su na jednakoj udaljenosti od $v_i \in V$, ali je rastojanje između novog centra c_{in} i njegovog *backup* centra $c1_P(c_{in})$ manje nego što je rastojanje između prethodnih *reference* i *backup* centara.

Shodno tome, ukoliko $d(u_i, u_j)$ predstavlja najkraće rastojanje između čvorova u_i i u_j, skup ovakvih korisnika je definisan kao:

$V' = \{v_i \in V | cndV_1'(v_i) \vee (cndV_2'(v_i) \wedge cndV_3'(v_i))\}$,
gde je:

$cndV_1'(v_i) = d(v_i, c_{in}) < d(v_i, c1_P(v_i))$,
$cndV_2'(v_i) = [d(v_i, c_{in}) = d(v_i, c1_P(v_i))]$,
$cndV_3'(v_i) = d(c_{in}, c1_P(c_{in})) < d\left(c1_P(v_i), c1_P(c1_P(v_i))\right)$.

U ovom slučaju, odgovarajuće r vrednosti se izračunavaju kao:

$$r(c_{in}, c1_P(c_{in})) = max_{v_i \in V'}\{d(v_i, c_{in}) + d(c_{in}, c1_P(c_{in}))\}. \quad (2.5)$$

2) Korisnici zadržavaju svoj *reference* centar i dobijaju novi *backup* centar c_{in}, ili $c2_P(v_i)$ postaje novi *reference* centar zajedno sa novim centrom c_{in} kao *backup* centrom korisnika $v_i \in V$. Kasnija situacija se dešava kada su centri $c1_P(v_i)$ i $c2_P(v_i)$ na jednakoj udaljenosti od v_i ali otvaranje centra c_{in} čini centar $c2_P(v_i)$ boljim izborom za *reference* centar usled smanjenog rastojanja do novog *backup* centra c_{in}, tj. važi da je

$d(c2_P(v_i), c_{in}) < min\left\{d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right), d(c1_P(v_i), c_{in})\right\}$. Na osnovu prethodnog, razlikuju se dva skupa korisnika:

$V'' = \{v_i | cndV_1''(v_i) \wedge (cndV_2''(v_i) \vee cndV_3''(v_i))\}$,
gde je:

$cndV_1''(v_i) = d(c1_P(v_i), c_{in}) < d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right)$,
$cndV_2''(v_i) = d\left(v_i, c1_P(v_i)\right) < d(v_i, c_{in})$,
$cndV_3''(v_i) = \left[d\left(v_i, c1_P(v_i)\right) = d(v_i, c_{in})\right] \wedge d(c1_P(v_i), c_{in}) < d\left(c_{in}, c1_P(c_{in})\right)$ i

$V''' = \{v_i \in V | cndV_1'''(v_i) \wedge cndV_2'''(v_i)\}$,
gde je:

$cndV_1'''(v_i) = \left[d\left(v_i, c1_P(v_i)\right) = d\left(v_i, c2_P(v_i)\right)\right]$,
$cndV_2'''(v_i) = d(c2_P(v_i), c_{in}) < min\{d(c1_P(v_i), c1_P(c1_P(v_i))), d(c1_P(v_i), c_{in})\}$.

Na osnovu ova dva skupa korisnika, odgovarajuće *r* vrednosti se izračunaju kao:

$$r\left(p_j, c_{in}\right) = max\left\{maxV''\left(p_j, c_{in}\right), maxV'''\left(p_j, c_{in}\right)\right\}, \quad (2.6)$$

gde je:

$maxV''\left(p_j, c_{in}\right) = max_{v_i \in V''}\left\{d\left(v_i, p_j\right) + d\left(p_j, c_{in}\right) \middle| p_j = c1_P(v_i)\right\}$,
$maxV'''\left(p_j, c_{in}\right) = max_{v_i \in V'''}\left\{d\left(v_i, p_j\right) + d\left(p_j, c_{in}\right) \middle| p_j = c2_P(v_i)\right\}$.

3) Preostali skup uključuje korisnike koji zadržavaju svoj par centara iz rešenja *P*:

$V'''' = V \setminus (V' \cup V'' \cup V''')$.

U ovom slučaju, *r* vrednosti su date kao:

$$r\left(p_j, c1_P\left(p_j\right)\right) = max_{v_i \in V''''}\left\{d\left(v_i, p_j\right) + d\left(p_j, c1_P\left(p_j\right)\right) \middle| p_j = c1_P(v_i)\right\}. \quad (2.7)$$

U svim prethodnim izrazima (2.5) – (2.7) vrednosti u zagradama mogu biti izračunate za konstantno vreme i stoga je vremenska kompleksnost izračunavanja svih *r* vrednosti:

$O(|V'| + |V''| + |V'''| + |V''''|) = O(|V|) = O(n)$. ■

Svojstvo 2.2. Pomoću nizova $c1_P$ i $c2_P$, *z* vrednosti u Algoritmu 2.2 mogu biti izračunate za vremensku kompleksnost $O(n)$.

Dokaz. Najpre, primetimo da za bilo koji čvor $v \in V$ najbliži centar iz skupa $P_i = P \cup \{c_{in}\} \backslash \{p_i\}$, označen kao $c1_{P_i}(v)$, može biti određen u konstantnoj vremenskoj složenosti. Naime, korisnik nakon zatvaranja centra p_i, ili zadržava svoj najbliži centar $c1_{P'}(v)$ iz rešenja $P' = P_i \cup \{p_i\} = P \cup \{c_{in}\}$, ili mu je kao novi *reference* centar dodeljen drugi-najbliži centar $c2_{P'}(v)$ iz rešenja P'. Slično, $c1_{P'}(v)$ je isti centar kao i $c1_P(v)$, tj. najbliži centar u skupu *P*, ili je izabran kao korisniku v bliži od centara $c2_P(v)$ i c_{in}, gde je c_{in} novo-otvoreni centar. Centar $c2_{P'}(v)$ se određuje kao najbliži iz skupa centara $\{c1_P(v), c2_P(v), c_{in}\} \backslash \{c1_{P'}(v)\}$. Na osnovu ovih zapažanja, definišu se sledeća dva slučaja:

1) Zatvaranje centra $p_i \in P$ dovodi do toga da korisnik $v_j \in V$ izgubi svoj *reference* centar, tj. $c1_{P'}(v) = p_i$. Skup ovakvih korisnika je definisan kao:

$\overline{V_i'} = \{v_j \in V | c1_{P'}(v_j) = p_i\}$.

U ovom slučaju, korisniku $v_j \in V$ se kao novi *reference* centar dodeljuje drugi-najbliži centar iz rešenja P', tj. najbliži centar iz skupa centara $\{c1_P(v), c2_P(v), c_{in}\} \backslash \{p_i\}$. Na osnovu toga, ukoliko $d(u_i, u_j)$ predstavlja najkraće rastojanje između čvorova u_i i u_j, maksimalna vrednost *p*-sledeći centar funkcije među svim korisnicima koji su izgubili svoj *reference* centar se određuje kao:

$$z'(p_i) = \max_{v_j \in \overline{V}_i} \left\{ \min_{p \in \{c1_P(v_j),\ c2_P(v_j),\ c_{in}\} \setminus \{p_i\}} \left\{ d(v_j, p) + d\left(p, c1_{P_i}(p)\right) \right\} \right\}. \quad (2.8)$$

U izrazu (2.8), $c1_{P_i}(p)$ je najbliži centar korisniku p u skupu centara $P_i = P \cup \{c_{in}\} \setminus \{p_i\}$. Određuje se u konstantnoj vremenskoj složenosti pretraživanjem skupa centara $\{c1_P(p), c2_P(p), c_{in}\} \setminus \{p, p_i\}$, što dalje implicira da izračunavanje $z'(p_i)$ vrednosti zahteva $O\left(\left|\overline{V}_i'\right|\right)$ vremensku kompleksnost.

2) Korisnik $v_j \in V$ zadržava svoj *reference* centar iz rešenja P', ali zatvaranjem centra $p_i \in P$ gubi *backup* centar $c1_{P'}\left(c1_{P'}(v_j)\right)$. Skup ovakvih korisnika je definisan kao:

$$\overline{V}_i'' = \left\{ v_j \in V \middle| c1_{P'}\left(c1_{P'}(v_j)\right) = p_i \wedge c1_{P'}(v_j) \neq p_i \right\}.$$

U ovom slučaju, kao novi *backup* centar dodeljuje se drugi-najbliži centar $c2_{P'}\left(c1_{P'}(v_j)\right)$ centru $c1_{P'}(v_j)$ iz rešenja P', tj. najbliži centar iz skupa centara $\left\{c1_P\left(c1_{P'}(v_j)\right), c2_P\left(c1_{P'}(v_j)\right), c_{in}\right\} \setminus \{p_i, c1_{P'}(v_j)\}$. Prema tome, maksimalna vrednost *p*-sledeći centar funkcije među svim korisnicima koji su izgubili svoj *backup* centar se određuje kao:

$$z''(p_i) = \max_{v_j \in \overline{V}_i''} \left\{ \min_{p \in \left\{c1_P\left(c1_{P'}(v_j)\right),\ c2_P\left(c1_{P'}(v_j)\right),\ c_{in}\right\} \setminus \{p_i,\ c1_{P'}(v_j)\}} \left\{ d\left(v_j, c1_{P'}(v_j)\right) + d\left(c1_{P'}(v_j), p\right) \right\} \right\}. \quad (2.9)$$

U izrazu (2.9), centri $c1_{P'}(v_j)$, $c1_P\left(c1_{P'}(v_j)\right)$ i $c2_P\left(c1_{P'}(v_j)\right)$ se mogu pronaći u konstantnoj vremenskoj složenosti, pa prema tome izračunavanje $z''(p_i)$ vrednosti zahteva $O\left(\left|\overline{V}_i''\right|\right)$ vremensku kompleksnost.

Na osnovu (2.8) i (2.9) rezultujuća vrednost funkcije cilja svih korisnika koji su izgubili svoj *reference* ili *backup* centar izračunava se kao:

$$z(p_i) = max\{z'(p_i), z''(p_i)\}. \quad (2.10)$$

Dakle, na osnovu 1) i 2) vremenska kompleksnost izračunavanja z vrednosti u odnosu na broj korisnika je:

$O\left(\sum_{i=1,\dots,p}\left(\left|\overline{V}_i'\right| + \left|\overline{V}_i''\right|\right)\right) \approx O(|V|) = O(n)$. ■

Imajući poznate *r* i *z* vrednosti, sledeće svojstvo pokazuje kako je moguće pronaći optimalan centar za zatvaranje na efikasan način.

Svojstvo 2.3. Najbolji izbor centra koji bi trebalo zatvoriti je predstavljen kao:

$$c_{out} = \arg\min_{p_i \in P} \left\{ max \left\{ z(p_i), \max_{\substack{p_j \in P' \\ p_j \neq p_i \\ c1_{P'}(p_j) \neq p_i}} \left\{ r(p_j, c1_{P'}(p_j)) \right\} \right\} \right\}$$

i stoga, može biti određen za vremensku složenost od $O(p^2)$. U izrazu je $P' = P \cup \{c_{in}\}$, a $c1_{P'}(p_j)$ predstavlja najbliži centar centru p_j u rešenju P'.

Dokaz. Neka je opet skup centara $P_i = P \setminus \{p_i\} \cup \{c_{in}\}$ rešenje dobijeno zatvaranjem centra p_i i otvaranjem centra c_{in}. Dodatno, definisan je $\overline{V}_i''' = V \setminus \left\{\overline{V}_i' \cup \overline{V}_i''\right\}$ skup korisnika koji zadržavaju svoje *reference* i *backup* centre i nakon zatvaranja $p_i \in P$ centra. Skupovi $\overline{V}_i'$ i $\overline{V}_i''$ su definisani na isti način kao u prethodnom Svojstvu 2.2. Koristeći ovu notaciju, vrednost funkcije cilja najboljeg rešenja P_i^* je određena na sledeći način:

$$
\begin{aligned}
f(P_i^*) &= \min_{i=1,\dots,p} f(P_i) \\
&= \min_{i=1,\dots,p} \left\{ \max_{v_j \in V} \left\{ d\left(v_j, c1_{P_i}(v_j)\right) + d\left(c1_{P_i}(v_j), c1_{P_i}\left(c1_{P_i}(v_j)\right)\right)\right\}\right\} \\
&= \min_{i=1,\dots,p} \left\{ \max_{v_j \in \left(\overline{V}_i' \cup \overline{V}_i'' \cup \overline{V}_i'''\right)} \left\{ d\left(v_j, c1_{P_i}(v_j)\right) + d\left(c1_{P_i}(v_j), c1_{P_i}\left(c1_{P_i}(v_j)\right)\right)\right\}\right\} \\
&= \min_{i=1,\dots,p} \left\{ \max \left\{ \begin{array}{l} \max\limits_{v_j \in \left(\overline{V}_i' \cup \overline{V}_i''\right)} \left\{ d\left(v_j, c1_{P_i}(v_j)\right) + d\left(c1_{P_i}(v_j), c1_{P_i}\left(c1_{P_i}(v_j)\right)\right)\right\}, \\ \max\limits_{v_j \in \overline{V}_i'''} \left\{ d\left(v_j, c1_{P_j}(v_j)\right) + d\left(c1_{P_i}(v_j), c1_{P_i}\left(c1_{P_j}(v_j)\right)\right)\right\} \end{array} \right\}\right\} \\
&= \min_{i=1,\dots,p} \left\{ \max \left\{ z(p_j), \max_{\substack{p_j \in P' \\ p_j \neq p_i \\ c1_{P'}(p_j) \neq p_i}} \left\{ r(p_j, c1_{P'}(p_j)) \right\} \right\}\right\} \\
&= \min_{p_i \in P} \left\{ \max \left\{ z(p_i), \max_{\substack{p_j \in P' \\ p_j \neq p_i \\ c1_{P'}(p_j) \neq p_i}} \left\{ r(p_j, c1_{P'}(p_j)) \right\} \right\}\right\}.
\end{aligned}
\tag{2.11}
$$

Primetimo da je vrednost $max_{v_j \in \left(\overline{V}_i' \cup \overline{V}_i''\right)} \left\{ d\left(v_j, c1_{P_i}(v_j)\right) + d\left(c1_{P_i}(v_j), c1_{P_i}\left(c1_{P_i}(v_j)\right)\right)\right\}$, u izrazu (2.11), jednaka $z(p_j)$ vrednosti. Sa druge strane, $max_{v_j \in \overline{V}_i'''} \left\{ d\left(v_j, c1_{P_j}(v_j)\right) + d\left(c1_{P_i}(v_j), c1_{P_i}\left(c1_{P_j}(v_j)\right)\right)\right\}$ je zapravo najveći "radijus" centara koji su zajedno sa svojim *backup* centrima ostali u rešenju nakon zatvaranja centra p_i i stoga se može izračunati kao $max_{\substack{p_j \in P' \\ p_j \neq p_i, c1_{P'}(p_j) \neq p_i}} \{r(p_j, c1_{P'}(p_j))\}$.

Dakle, na osnovu (2.11) sledi da se korišćenjem pomoćnih struktura podataka, algoritam za određivanje najboljeg centra za zatvaranje može implementirati sa vremenskom kompleksnošću od $O(p^2)$. ■

Svojstvo 2.3 potvrđuje da pohlepna procedura predstavljena Algoritmom 2.2 optimalno identifikuje centar c_{out} za zatvaranje. Neposredna posledica izraza (2.11), opisana u nastavku, daje broj proverenih rešenja primenom pomenutog pohlepnog pristupa.

Posledica. Korišćenjem pohlepnog pristupa za određivanje najboljeg centra za zatvaranje, p različitih rešenja iz okoline $N_1(P)$ je posećeno, gde je P tekuće rešenje problema p-sledećeg centra.

Svojstvo 2.4. Vremenska kompleksnost jedne iteracije *LocalSearchVertexSubstitution* procedure (Algoritam 2.2), uz korišćenje pomoćnih struktura podataka, u najgorem slučaju je $O(n^2 + p^2 n)$.

Dokaz. Da bi se odredila vremenska složenost jedne iteracije *LocalSearchVertexSubstitution* algoritma potrebno je najpre odrediti kompleksnost svih procedura koje on koristi. Na osnovu Svojstava 2.1 i 2.2, vremenska složenost izračunavanja r i z vrednosti je $O(n)$. Najbolji centar za zatvaranje je određen za $O(p^2)$ vremensku kompleksnost. *Update* procedura ažurira nizove *c1* i *c2* dužine p za svakog od n korisnika, pa je prema tome vremenska složenost $O(pn)$. Vremenska složenost *ExistsRelaxedDistance* procedure je konstantna. *LocalSearchVertexSubstitution* algoritam poziva *ExistsRelaxedDistance*, kao i proceduru zamene centara najviše $n - p$ puta. *Update* procedura se poziva samo jedanput. Na osnovu toga, kompleksnost *LocalSearchVertexSubstitution* procedure u najgorem slučaju je $O(n^2 + p^2 n) + O(pn) \approx O(n^2 + p^2 n)$. ■

Posledica. Ukoliko je p značajno manje od n ($p << n$), vremenska kompleksnost jedne iteracije *LocalSearchVertexSubstitution* algoritma (Algoritam 2.2) je $O(n^2)$.

Svakako, *LocalSearchVertexSubstitution* predstavlja heuristički pristup implementaciji *1-Interchange* algoritma, tj. proceduri zamene čvorova u N_1 okolini tekućeg rešenja. Zapravo, heuristički pristup je primenjen u svakoj od sledećih funkcija: *ExistsRelaxedDistance*, *Calulate_z* i *Calulate_r*. Pristup proizilazi iz činjenice da se ne vodi evidencija o svim centrima koji su ekvidistantni (najbliži) korisniku, već se prate samo dva centra. Preliminarni eksperimenti u kojima su proveravani svi ekvidistantni centri su rezultirali negativnim uticajem na ukupne performanse pristupa: proces pretrage je usporen i stoga su često dobijana lošija rešenja za isto vreme pretrage. Sa druge strane, kao što je pomenuto u uvodu, cilj je predložiti rešenje problema *p*-sledećeg centra u duhu "manje je više" (*eng.* "*less is more*") pristupa koji potencira što manje algoritamskih komponenti sa najvišim mogućim performansama. U tom cilju, evidentiraju se najviše dva ekvidistantna centra i kao što je pokazano u narednim poglavljima, čak i sa ovakvom redukcijom rešenje se pokazalo superiornim u odnosu na već postojeće algoritme za pNCP.

Algoritam 2.3: Zamena čvorova u N_1 okolini (*eng. 1-interchange move*) u kontekstu *p*-sledeći centar problema

Move(x_{cur}, c_{in}, c1, c2)
Initialization:
Set $z(x_{cur}(i)) \leftarrow 0$ for all $i = 1, ..., p$
Set $r(x_{cur}(i), x_{cur}(j)) \leftarrow 0$ for all $i = 1, ..., p$ and $j = i + 1, ..., p, c_{in}$

Add center:
$in \leftarrow x_{cur}(c_{in})$
***For Each** user = $x_{cur}(1), \ldots, x_{cur}(n)$*
*****calculate function value if center in is a new reference or backup center*****
f_{in}, center ← GetFunctionValueForNewCentarIn(user, in, c1, c2) [Algorithm 2.3.1]
*****calculate function value if center in is a new reference or backup center and "center" is deleted*****
f_{out} ← GetBackupFunctionValueForNewCentarIn(user, in, center, c1, c2) [Algorithm 2.3.2]
*****calculate function value if center in is not a new reference center*****
f_{cur} ← GetFunctionValue(user, in, c1, c2) [Algorithm 2.3.3]

Update r and z values:
***If** $f_{in} \leq f_{cur}$*
******center in is either a new reference or backup center****
r(in, center) ← r(center, in) ← max(f_{in}, r(in, center))
*****calculate function value if center in is not a new reference center and "center" is deleted*****
f'_{out} ← GetBackupFunctionValue(user, in, center, c1, c2) [Algorithm 2.3.4]
*****in case that "center" is deleted*****
z(center) ← max(min(f_{out},f'_{out}), z(center))
Else
*****nc1(user, in) is the reference center; c1(nc1(user, in)) is the backup center*****
r(nc1(user, in), c1(nc1(user, in))) ← max(f_{cur}, r(nc1(user, in), c1(nc1(user, in))))
*****in case that the reference center is deleted*****
f'_{out} ← GetBackupFunctionValue(user, in, nc1(user, in), c1, c2) [Algorithm 2.3.4]
*$f \leftarrow f_{in}$ **if** nc1(user, in) ≠ center **else** f_{out}*
z(nc1(user, in)) ← max(min(f,f'_{out}), z(nc1(user, in)))
*****in case that the backup center is deleted*****
f''_{out} ← GetBackupFunctionValue(user, in, c1(nc1(user, in)), c1, c2) [Algorithm 2.3.4]
*$f \leftarrow f_{in}$ **if** c1(nc1(user, in)) ≠ center **else** f_{out}*
z(c1(nc1(user, in))) ← max(min(f,f''_{out}), z(c1(nc1(user, in))))
End If
End For Each

Best deletion:
c_{out}, f_{cur} ← FindBestDeletion(x_{cur}, r, z, c_{in}) [Algorithm 2.3.5]

***Return** f_{cur}, c_{out}*

Algoritam 2.3.1: Izračunavanje vrednosti pNCP funkcije korisnika *u* sa novim *reference* ili *backup* centrom *in* ("*counterpart center*" je novi *backup* ukoliko je *in* novi *reference* centar i obrnuto)

GetFunctionValueForNewCentarIn(u, in, c1, c2)
in as the new reference center*
$f \leftarrow nc1Dist(u, in)$ **if** $dist(u, in) \leq dist(u, nc1(u, in))$ **else** ∞
$counterpart\_center \leftarrow c1(in)$

in as the new backup center*
$f' \leftarrow dist(u, nc1(u, in)) + dist(nc1(u, in), in)$ **if** $dist(u, nc1(u, in)) \leq dist(u, in)$ **else** ∞
If $f > f'$
    $f \leftarrow f'$
    $counterpart\_center \leftarrow nc1(u, in)$
End If

Return *f, counterpart_center*

Algoritam 2.3.2: Izračunavanje vrednosti pNCP funkcije korisnika *u* nakon zatvaranja centra *out* i dodele novog *reference* ili *backup* centra *in*

GetBackupFunctionValueForNewCentarIn(u, in, out, c1, c2)
****"center" is the closest center (not including in), after center out has been deleted****
$center \leftarrow nc1(u, in)$ **if** $nc1(u, in) \neq out$ **else** $nc2(u, in)$

in as the new reference center*
If $c1(in) \neq out$
    $f' \leftarrow nc1Dist(u, in)$ **if** $dist(u, in) \leq dist(u, center)$ **else** ∞
Else
    $f' \leftarrow dist(u, in) + dist(in, c2(in))$ **if** $dist(u, in) \leq dist(u, center)$ **else** ∞
End If

in as the new backup center*
$f'' \leftarrow dist(u, center) + dist(center, in)$ **if** $dist(u, center) \leq dist(u, in)$ **else** ∞

Return $min(f', f'')$

Algoritam 2.3.3: Izračunavanje vrednosti pNCP funkcije korisnika *u* sa potencijalno novim *backup* centrom *in* (ukoliko je *in* novi *reference* centar, kao rezultat se vraća ∞)

GetFunctionValue(u, in, c1, c2)
$f \leftarrow nc1Dist(u, nc1(u, in))$ **if** $dist(u, nc1(u, in)) \leq dist(u, in)$ **else** ∞
Return *f*

Algoritam 2.3.4: Izračunavanje vrednosti pNCP funkcije korisnika *u* nakon zatvaranja centra *out* (novi centar *in* može ostati nedodeljen ili postati *backup* centar, a ako je *in* ipak novi *reference* centar vrednost funkcije je ∞)

```
GetBackupFunctionValue(u, in, out, c1, c2)
If out ≠ nc1(u, in) and out ≠ c1(nc1(u, in))
        **neither the reference nor the backup center is deleted**
        f ← GetFunctionValue(u, in, c1, c2) [Algorithm 2.3.3]
Else
        If out = nc1(u, in)
                **the reference center is deleted**
                f ← nc2_1Dist(u, in) if dist(u, nc2(u, in)) ≤ dist(u, in) else ∞
        Else
                **the backup center is deleted**
                f ←nc2_2Dist(u, in) if dist(u, nc1(u, in)) ≤ dist(u, in) else ∞
        End If
End If
Return f
```

Algoritam 2.3.5: Optimalan odabir centra za zatvaranje u tekućem rešenju i izračunavanje vrednosti funkcije cilja novog rešenja

```
FindBestDeletion(x_cur, r, z, c_in)
f ← ∞
For Each i = {1, ..., p}
        cur ← z(x_cur(i))
        For Each j = {1, ..., p} ∪{c_in} \ {i}
                If x_cur(i) ≠ c1(x_cur(j))
                        cur ← max(r(x_cur(j), c1(x_cur(j))), cur)
                End If
        End For Each
        If f > cur
                f ← cur
                c_out ← i
        End If
End For Each
Return c_out, f
```

Algoritam 2.2 daje samo nacrt *LocalSearchVertexSubstitution* implementacije na visokom nivou apstrakcije. Detaljan opis je predstavljen kroz pseudokod Algoritama 2.3 – 2.5. Rešenje problema predstavlja niz x_{cur}, na taj način da prvih p elemenata niza čine tekuće rešenje. Preostalih n - p elemenata sa kraja niza, gde je n broj korisnika, predstavlja samo korisnike u tekućem rešenju. Algoritam 2.3 opisuje *Move* proceduru, koja na osnovu ulaznih parametara x_{cur}, c_{in}, *c1* i *c2*, kreira *r* i *z* strukture i vraća novu vrednost funkcije cilja f_{cur}, kao i centar c_{out} koji bi trebalo zameniti centrom c_{in}. Pri tome, c_{in} i c_{out} predstavljaju indekse centara u nizu x_{cur}. *Update* procedura (Algoritam 2.4) ažurira nizove centara *c1* i *c2* na osnovu novog tekućeg rešenja x_{cur}. Dodatno, predstavljeni algoritmi koriste sledeće funkcije:

- *nc1(v, in, c1, c2)* – vraća *reference* (najbliži) centar korisnika *v*. Ukoliko su centri *c1(v)* i *c2(v)* na jednakoj udaljenosti od korisnika *v*, prednost ima centar sa bližim *backup* centrom. Novi centar *in* takođe može biti *backup* centar,
- *nc2(v, in, c1, c2)* – vraća *backup* centar korisnika *v*,
- *nc1Dist(v, p, c1)* - vraća vrednost funkcije cilja za korisnika *v* i *reference* centar *p*,
- *nc2_1Dist(v, in, c1, c2)* – vraća novu vrednost funkcije cilja korisnika *v* nakon što se u tekućem rešenju njegov *reference* centar zameni novim centrom *in*,
- *nc2_2Dist(v, in, c1, c2)* - vraća novu vrednost funkcije cilja korisnika *v* nakon što se u tekućem rešenju njegov *backup* centar zameni novim centrom *in*,

- *dist(v, u)* – vraća dužinu najkraćeg puta između čvorova *v* i *u*.

Zbog preglednosti, u algoritmima su izostavljeni *c1* i *c2* argumenati prilikom poziva ovih funkcija.

Kako koraci Algoritma 2.3 nisu tako očigledni, sledi kratko objašnjenje. *Move* procedura na osnovu izraza (2.3) i (2.4) izračunava *r* i *z* vrednosti i određuje optimalan centar za zatvaranje. Vrednost funkcije cilja tekućeg rešenja proširenog centrom $in = x_{cur}(c_{in})$ odgovara maksimalnom elementu *r* strukture. Struktura se ažurira obilaskom svakog od korisnika, tj. najpre se proverava da li nakon otvaranja centra *in*, *in* postaje jedan od centara posećenog korisnika (označenog sa *user*). U tom slučaju, izračunava se f_{in} vrednost funkcije koja odgovara *user*-u kada je *in* njegov novi *reference* ili *backup* centar. U suprotnom, kada *in* nije dodeljen *user*-u, f_{in} dobija vrednost ∞. Takođe, u slučaju da *user* zadrži svoj *reference* centar i u novom rešenju koje uključuje centar *in*, f_{cur} predstavlja vrednost funkcije koja odgovara *user*-u u tom rešenju. U suprotnom, ukoliko je *in* novi *reference* centar, f_{cur} vrednost je ∞.

Upoređivanjem f_{in} i f_{cur} vrednosti, zaključuje se da li je centar *in* dodeljen korisniku *user* u novom rešenju. Ukoliko važi da je $f_{in} \leq f_{cur}$, *in* je novi *reference* ili *backup* centar korisnika *user*. U tom slučaju, određuje se drugi centar, tj. *reference* centar korisnika *user* ukoliko je *in* novi *backup* centar i obrnuto. Dodeljeni centar koji je različit od *in* se označava sa *center*. Ažurira se *r* vrednost koja odgovara dodeljenom paru centara (*in*, *center*) ukoliko je trenutna vrednost manja od f_{in}. Svakako, novi centar *in* neće biti odmah zatvoren, tako da je potrebno odrediti još samo vrednost funkcije koja odgovara korisniku *user* u slučaju zatvaranja dodeljenog centra *center*. Nakon zatvaranja ovog centra, *in* ponovo može biti dodeljen *user*-u kao *reference* ili *backup* centar, i u tom slučaju nova vrednost funkcije cilja koja odgovara *user*-u je f_{out}. Takođe, može se desiti da korisnik *user* za svoj *reference* centar povrati jedan od prethodno dodeljenih centara iz tekućeg rešenja, što odgovara vrednosti f'_{out}. Svakako, teži se minimalnoj vrednosti, tako da će nova vrednost funkcije biti $min(f_{out}, f'_{out})$. Na osnovu ove vrednosti ažurira se *z* element koji odgovara centru *center*, tj. njegovom potencijalnom zatvaranju.

Ukoliko *user* ipak zadrži svoje prethodno dodeljene centre ($f_{cur} < f_{in}$), *r* vrednost koja odgovara ovom paru centara se ažurira na osnovu vrednosti f_{cur}. Uz to, potrebno je još ažurirati i *z* vrednosti koje odgovaraju potencijalnom zatvaranju i *reference* i *backup* centra.

Konačno, na osnovu ažuriranih *r* i *z* vrednosti, centar koji bi trebalo zatvoriti se određuje tako da nova vrednost funkcije cilja bude minimizovana na osnovu izraza (2.4).

Algoritam 2.4: Ažuriranje nizova najbližih i drugih-najbližih centara svih korisnika

```
Update(x_cur, c1, c2)
For each user, properly update the closest and the second-closest center:
For Each user = x_cur(1), …, x_cur(n)
        c1_dist ← ∞; c1_center ← null
        c2_dist ← ∞; c2_center ← null
        For Each center = x_cur(1), ..., x_cur(p)
                d ← dist(user, center)
                If d < c1_dist or d = c1_dist and dist(center, c1(center)) < dist(c1_center, c1(c1_center))
                        c2_center ← c1_center
                        c2_dist ← c1_dist
                        c1_center ← center
                        c1_dist ← d
                Else If d < c2_dist or d = c2_dist and
                        dist(center, c1(center)) < dist(c2_center, c1(c2_center))
                        c2_center ← center
                        c2_dist ← d
                End If
        End For Each
        c1(user) ← c1_center
        c2(user) ← c2_center
End For Each
Return c1, c2
```

Na kraju, Algoritam 2.5 sadrži *LocalSearchVertexSubstitution* pseudokod. Zajedno sa *c1* i *c2* nizovima, algoritam prihvata tekuće rešenje x_{cur}, kritičnog korisnika u_c i trenutnu vrednost funkcije cilja f_{cur} kao ulazne argumente i ažurira ih ukoliko naiđe na bolje rešenje.

Algoritam 2.5: Procedura zamene čvorova tokom faze lokalne pretrage za *p*-sledeći centar problem

LocalSearchVertexSubstitution(x_{cur}, c1, c2, u_c, f_{cur})
Main loop:
While *True*
    $f' \leftarrow \infty$
    $c_{in} \leftarrow null$; $c_{out} \leftarrow null$
    For Each *in = p +1, ..., n*
        Investigate $N_1(x_{cur})$ neighborhood with center x_{cur}(in) as a new center and find the best deletion:
        If *ExistsRelaxedDistance(c1, c2, u_c, x_{cur}(in), f_{cur})*
            f, out ← *Move(x_{cur}, in, c1, c2)*
            If $f < f'$
                $f' \leftarrow f$
                $c_{in} \leftarrow in$
                $c_{out} \leftarrow out$
            End If
        End If
    End For Each
    If $f_{cur} \leq f'$
        There is not found improvement in the neighborhood:
        Break Main loop
    End If
    $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
    Update(x_{cur}, c1, c2)
    $f_{cur} \leftarrow f'$
    $u_c \leftarrow$ ***select*** *user* ***from*** $\{x_{cur}(1), \ldots, x_{cur}(n)\}$ ***where*** *nc1Dist(user, nc1(user, null))* $= f_{cur}$
End While
Return x_{cur}, u_c, f_{cur}

2.3. *Shaking* procedura za problem *p*-sledećeg centra

Uz sofisticiranu lokalnu pretragu, implementirana je takođe i sofisticirana *shaking* procedura. *Shaking* procedura je i dalje zasnovana na generisanju slučajnog rešenja iz okoline $N_k(P)$, ali umesto potpuno slučajnog, u svakoj iteraciji se generiše delimično pohlepno rešenje. Naime, da bi se generisalo rešenje iz $N_k(P)$ okoline, procedura uključuje *k* iteracija pri čemu se u svakoj iteraciji vrši razmena po jednog para centara. Novi centar koji se otvara se bira tako da se vrednost funkcije kritičnog korisnika smanjuje, dok se zatvaranje centra vrši potpuno slučajnim izborom. Nacrt *shaking* procedure je dat u Algoritmu 2.6.

Sofisticiranija verzija VNS Algoritma 1.1, koja je implementirana u disertaciji je predstavljena Algoritmom 2.7. Algoritam prati iste principe kao Algoritam 1.1**,** ali uz naprednije načine intenzifikacije (vidi Algoritam 2.2) i diversifikacije procesa pretrage (vidi Algoritam 2.6).

S obzirom da *c1*, *c2*, a i druge vrednosti (rastojanja do ostalih centara) mogu biti jednake, i kako predloženi algoritam uzima u obzir samo par najbližih centara, moguće je "potceniti" pronađeno rešenje. Svakako, algoritam efikasno procenjuje gornju granicu vrednosti funkcije cilja, i stoga se na kraju vrednost pronađenog rešenja ponovo izračunava.

Algoritam 2.6: *Shaking* procedura u kontekstu *p*-sledeći centar problema

***Shaking*(x_{cur}, c1, c2, u_c, f_{cur}, k)**
For $j = 1, ..., k$
    Choose center to be opened c_{in} at random
    If ExistsRelaxedDistance(c1, c2, u_c, c_{in}, f_{cur})
        Choose center to be closed c_{out} at random
        Update(x_{cur}, c1, c2, u_c, f_{cur}, c_{in}, c_{out})
    End If
End For

Algoritam 2.7: Sofisticirani VNS algoritam za problem *p*-sledećeg centra

VariableNeighborhoodSearch(k_{max}, t_{max})
Initialization:
Randomly initialize x_{opt}; according to x_{opt} initialize arrays c1 and c2, f_{opt}, u_c; copy initial solution into the current one, i.e., copy f_{opt}, x_{opt}, c1, c2 and u_c into f_{cur}, x_{cur}, c1', c2' and u_c', respectively.
Repeat the Main step until the stopping condition is met (e.g., time ≤ t_{max})
Main step:
    $k \leftarrow 1$
    While $k \leq k_{max}$
        Shaking operator:
        *****generate a solution at random from kth neighborhood*****
        For Each $j = 1, ..., k$
        - *Take center to be inserted (c_{in}) at random if ExistsRelaxedDistance(c1', c2', u_c', $x_{cur}(c_{in})$, f_{cur})*
        - *Find center to be deleted (c_{out}) at random*
        - *Update x_{cur}, c1' and c2', i.e., execute:*
          $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
          Update(x_{cur}, c1', c2')
        - *Update f_{cur} and u_c' according to x_{cur}, c1' and c2'*

        End For Each
        Local search:
        If *any potentially better solution found*
            x_{cur}, c1', c2', u_c', $f_{cur} \leftarrow$ *LocalSearchVertexSubstitution(x_{cur}, c1', c2', u_c', f_{cur})*
            Move or not:
            If $f_{cur} \leq f_{opt}$
                *****Save current solution as the optimal; return to N_1*****
                $x_{opt} \leftarrow x_{cur}$; $f_{opt} \leftarrow f_{cur}$; $u_c \leftarrow u_c$'; c1 ← c1'; c2 ← c2'
                $k \leftarrow 1$
            Else
                *****Current solution is the optimal; change the neighborhood*****
                $x_{cur} \leftarrow x_{opt}$; $f_{cur} \leftarrow f_{opt}$; u_c' ← u_c; c1' ← c1; c2' ← c2
                $k \leftarrow k + 1$
            End If
        End If
    End While
Return $\{x_{opt}(1), ..., x_{opt}(p)\}$, f_{opt}

Izvorni kod kompletne implementacije algoritma se može naći na sledećoj adresi:
https://1drv.ms/u/s!AnaX64fUELcfgoF74G7lMuEksN5SGg?e=BFFH8q.

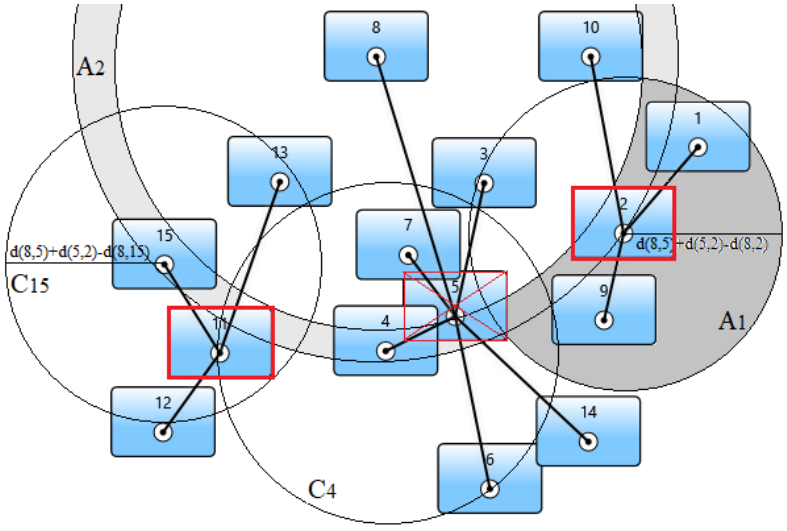
2.4. Modifikacija algoritma za prepoznavanje egzaktnih rešenja problema *p*-sledećeg centra

U Poglavljima 2.2 i 2.3 opisana je efikasna implementacija VNS metoda za rešavanje problema *p*-sledećeg centra. Algoritam, primenjen nad *OR-Library* test setom (inicijalno namenjenim testiranju rešenja *p*-medijan problema), pronašao je do sada najbolja poznata rešenja *p*-sledeći centar problema. Međutim, u pitanju je klasičan heuristički algoritam koji ne može garantovati globalnu optimalnost svojih rešenja. U ovom poglavlju je predstavljena modifikacija algoritma sposobna da u nekim slučajevima prepozna da li je pronađeno rešenje problema globalno optimalno.

Rečeno je da predloženi algoritam u svakoj iteraciji lokalne pretrage pokušava da unapredi tekuće rešenje. Primećeno je da vrednost funkcije cilja u slučaju problema *p*-sledećeg centra odgovara maksimalnoj sumi udaljenosti korisnika do najbližeg centra i rastojanja između tog centra i njemu najbližeg centra. Korisnik koji odgovara maksimalnoj udaljenosti je nazvan kritičnim korisnikom. Najbliži centar nekom korisniku je *reference* centar, a njemu najbliži *backup* centar. U osnovi faze lokalne pretrage algoritma je ideja o zameni jednog od centara tekućeg rešenja novim centrom tako da vrednost funkcije cilja koja odgovara kritičnom korisniku bude smanjena. Potrebno je kritičnom korisniku dodeliti novi *reference* centar koji je bliži od prethodnog ili novi *backup* centar koji bi bio bliži *reference* centru. Na ovaj način, vrednost funkcije kritičnog korisnika se može smanjiti, ali se ne garantuje unapređenje tekućeg rešenja kao ni optimalnost rešenja. Modifikovani algoritam proširuje rešenje iz prethodnog poglavlja i garantuje optimalnost novog rešenja u N_1 okolini tekućeg rešenja, ali nauštrb vremenske kompleksnosti algoritma, a samim tim i vremena potrebnog da se pronađe rešenje. U cilju objašnjenja algoritma za identifikaciju lokalno optimalnih rešenja, primer iz Poglavlja 2.2 (Slika 2.1) je proširen pojašnjenjem koje sledi i Slikom 2.2.

Najpre sledi kratka rekapitulacija primera ilustrovanog Slikom 2.1. Predstavljen je primer problema *p*-sledećeg centra sa $n = 15$ korisnika i $p = 3$ centra. Tekuće rešenje čine tri *reference* centra 2, 5 i 11. Kritični korisnik je čvor $u_c = 8$. Tokom iteracije lokalne pretrage, a u cilju unapređenja tekućeg rešenja, moguće je potražiti novi *reference* centar bliži kritičnom korisniku 8 (unutar kruga C_1) ili novi *backup* centar bliži *reference* centru 5 (unutar kruga C_2). Sa druge strane, ukoliko postoji bar još jedan centar na istoj udaljenosti kao i *reference* centar 5 (na kružnici C_1), kao novi *backup* može se otvoriti bilo koji centar, bliži tom ekvidistantnom centru nego što je prethodnom *reference* centru njegov *backup* centar.

Ukoliko u novom rešenju *reference* centar 5 kritičnog korisnika 8 bude zatvoren, novi *reference* centar se, sem u krugu C_1, može potražiti još i u oblasti A_2, određenoj rastojanjima krititičnog korisnika do najbližeg (centar 5) i drugog-najbližeg (centar 2) centra u tekućem rešenju (Slika 2.2). Svaki od centara iz ove oblasti je potencijalno novi *reference* centar i da bi svojim otvaranjem eventualno unapredio vrednost funkcije cilja, potrebno je pronaći *backup* centar na udaljenosti takvoj da je vrednost funkcije kritičnog korisnika 8 smanjena, tj. manja od $d(8,5) + d(5,2)$. Na primer, kao novi *reference* centar može se otvoriti centar 15, a odgovarajući *backup* centar se potražiti na rastojanju manjem od $d(8,5) + d(5,2) - d(8,15)$, tj. unutar kruga C_{15} (kao što je to u slučaju centra 11). Dakle, može se otvoriti novi, ali novi *reference* centar takođe može postati i postojeći centar ukoliko se nalazi na spoljašnjoj kružnici oblasti A_2. U tom slučaju, potrebno je pronaći postojeći ili pak otvoriti novi *backup* centar na odgovarajućoj udaljenosti od novog *reference* centara. Recimo, novi *reference* centar kritičnog korisnika 8 može postati centar 2, a kao novi *backup* centar otvoriti se jedan od centara koji se nalaze unutar oblasti A_1. Poluprečnik oblasti A_1 je $d(8,5) + d(5,2) - d(8,2)$. Nova vrednost funkcije cilja korisnika 8 nakon otvaranja centra u oblasti A_1, npr. centra 1, biće svakako manja od prethodne vrednosti $d(8,5) + d(5,2)$.



Sl. 2.2. Primer problema *p*-sledećeg centra za $n = 15$ korisnika i $p = 3$ centra; tekuće rešenje je $P = \{2, 5, 11\}$; kritični korisnik je $u_c = 8$; kritični centar $p_c = 5$ će biti zatvoren

Na osnovu prethodne analize, pronalazi se centar koji je potrebno uključiti u tekuće rešenje da bi se vrednost funkcije koja odgovara kritičnom korisniku smanjila. Drugim rečima, vrši se filtriranje potencijalno novih centara tako da u obzir dolaze samo oni koji smanjuju vrednost funkcije koja odgovara kritičnom korisniku. Iako se u opštem slučaju ne smanjuje složenost algoritma, filtriranjem potencijalnih rešenja sužava se opseg pretrage i ubrzava konvergencija ka lokalnom optimumu. Ipak, otvaranje novog centra nije dovoljno da bi novo rešenje bilo garantovano bolje od tekućeg. Da li će biti, zavisi od drugih korisnika koji u novom rešenju budu izgubili neke od svojih centara. Može se desiti da nekom od korisnika, koji u prethodnom rešenju nije kritični korisnik, budu dodeljeni novi centri tako da vrednost funkcije cilja bude i veća od prethodne vrednosti koja odgovara kritičnom korisniku. Prema tome, potrebno je optimalno izabrati i centar koji se zatvara tako da nova vrednost funkcije cilja bude što manja. Implementacija procedure koja optimalno identifikuje centar koji bi trebalo zatvoriti je prethodno objašnjena u Poglavlju 2.2. Sada je fokus na proširenju predloženog algoritma koje omogućava prepoznavanje optimalnih rešenja. U tom cilju, najpre se predstavlja svojstvo na osnovu kojeg je dalje implementirana pomenuta modifikacija.

Svojstvo 2.5. Neka je u_c kritični korisnik, p_c njegov *reference* centar, p'_c drugi-najbliži centar, a $c1(v)$ najbliži centar korisnika/centra v ($v \in V$) u tekućem rešenju P, gde je V skup svih korisnika. Tada, ukoliko $d(u, v)$ predstavlja najkraće rastojanje između čvorova u i v i postoji bolje rešenje u okolini $N_1(P)$ od tekućeg rešenja P, mora važiti bar jedno od sledećih tvrđenja:

(i) novi centar c_{in} nije udaljeniji od kritičnog korisnika u odnosu na centar p_c ($d(c_{in}, u_c) \leq d(p_c, u_c)$) i važi da je $d(u_c, c_{in}) + d\big(c_{in}, c1(c_{in})\big) < d(u_c, p_c) + d\big(p_c, c1(p_c)\big)$ (krug C_1 na Slici 2.1),

(ii) novi centar c_{in} nije bliži kritičnom korisniku u odnosu na centar p_c ($d(c_{in}, u_c) \geq d(p_c, u_c)$), postoji centar p ($p \in P$, uzimajući u obzir i p_c) takav da je na jednakoj udaljenosti od kritičnog korisnika kao i centar p_c ($d(p, u_c) = d(p_c, u_c)$) i važi $d(p, c_{in}) < d\big(p_c, c1(p_c)\big)$,

(iii) centar p_c je zatvoren, novi centar c_{in} nije udaljeniji od kritičnog korisnika u odnosu na centar p'_c (oblast A_2 na Slici 2.2: $d(c_{in}, u_c) \leq d(p'_c, u_c)$) i postoji centar p ($p \in P$ i $p \neq p_c$) takav da važi $d(u_c, c_{in}) + d(c_{in}, p) < d(u_c, p_c) + d\big(p_c, c1(p_c)\big)$,

(iv) centar p_c je zatvoren, novi centar c_{in} nije bliži kritičnom korisniku u odnosu na centar p'_c ($d(c_{in}, u_c) \geq d(p'_c, u_c)$) i postoje centri p_r i p_b ($p_r \in P$, $p_b \in P \cup \{c_{in}\}$ i $p_r \neq p_b \neq p_c$) takvi da važi $d(p_r, u_c) = d(p'_c, u_c)$ i $d(u_c, p_r) + d(p_r, p_b) < d(u_c, p_c) + d\big(p_c, c1(p_c)\big)$ (npr. oblast A_1 na Slici 2.2).

Dokaz. Vrednost funkcije cilja u rešenju P je određena sumom odgovarajućih rastojanja do centara koji su dodeljeni kritičnom korisniku: $f(P) = d(u_c, p_c) + d\big(p_c, c1(p_c)\big)$. Da bi novo rešenje u okolini $N_1(P)$ bilo bolje od tekućeg rešenja, kritičnom korisniku u_c mora biti dodeljen novi *reference* i/ili *backup* centar. Ukoliko

najbliži centar p_c ostane otvoren u novom rešenju, jedan od tih centara svakako mora biti novi centar c_{in}. Posmatrajmo posebno slučajeve kada centar p_c jeste i kada nije zatvoren:

1) *Reference* centar p_c nije zatvoren. Pretpostavimo da ne važe tvrđenja (i) i (ii). Posmatrajmo Sliku 2.1. Novi centar c_{in} se može naći sa unutrašnje strane kružnice C_1, na kružnici C_1 i sa spoljašnje strane kružnice C_1.

- Ukoliko se novi centar nalazi sa unutrašnje strane kružnice, biće najbliži kritičnom korisniku u_c, tj. novi *reference* centar korisnika u_c. Kako ne važi (i), to je: $d(u_c, c_{in}) + d\big(c_{in}, c1(c_{in})\big) \geq f(P)$, što je u kontradikciji sa činjenicom da je novo rešenje bolje od tekućeg rešenja P.

- U slučaju da je c_{in} na kružnici, korisnik u_c može zadržati svoj *reference* centar p_c ili dobiti novi, bilo c_{in} ili neki od centara $p \in P$ koji su na jednakoj udaljenosti od u_c kao i centar p_c (nalaze se na kružnici C_1). Kako ne važe (i) i (ii), to je: $d(u_c, c_{in}) + d\big(c_{in}, c1(c_{in})\big) \geq f(P)$, kao što je i za svaki centar p koji se nalazi na kružnici C_1 ($d(u_c, p) = d(u_c, p_c)$), uključujući i centar p_c: $d(u_c, p) + d(p, c_{in}) \geq f(P)$. Prema tome, dodelom centra c_{in}, bilo kao novog *reference* ili *backup* centra, kritičnom korisniku u_c, vrednost funkcije cilja nije smanjena.

- Kada je novi centar c_{in} van kruga C_1, *reference* centar kritičnog korisnika može ostati p_c ili postati neki od centara $p \in P$ koji su na jednakoj udaljenosti od u_c kao i centar p_c (nalaze se na kružnici C_1). Kako ne važi (ii), to je za svaki centar p koji se nalazi na kružnici C_1 (uključujući i p_c): $d(u_c, p) + d(p, c_{in}) \geq f(P)$. Prema tome, dodelom novog *backup* centra c_{in} kritičnom korisniku u_c, vrednost funkcije cilja nije smanjena.

2) *Reference* centar p_c je zatvoren. Pretpostavimo da ne važe tvrđenja (iii) i (iv). Posmatrajmo Sliku 2.2. Novi centar c_{in} se može naći sa unutrašnje strane spoljašnje kružnice oblasti A_2, na kružnici i u oblasti sa spoljašnje strane ove kružnice.

- Ukoliko se novi centar c_{in} nalazi sa unutrašnje strane unutrašnje kružnice oblasti A_2, ispravnost svojstva se može pokazati slično kao i u prvom slučaju pod 1).

- Svakako, kada je novi centar c_{in} sa unutrašnje strane spoljašnje kružnice oblasti A_2, biće najbliži kritičnom korisniku u_c, tj. novi *reference* centar korisnika u_c. Neka je u novom rešenju centar p *backup* centra c_{in}, tj. $p = c1(c_{in})$ ukoliko je $c1(c_{in}) \neq p_c$, a u suprotnom drugi-najbliži centar korisnika c_{in} u rešenju P. Kako ne važi (iii), to je: $d(u_c, c_{in}) + d(c_{in}, p) \geq f(P)$, što je u kontradikciji sa tim da je novo rešenje bolje od P.

- U slučaju da je c_{in} na spoljašnjoj kružnici oblasti A_2, korisnik u_c za novi *reference* centar dobija jedan od centara $p \in P \cup \{c_{in}\}$ koji su na jednakoj udaljenosti od u_c kao i centar p'_c, uključujući i p'_c i c_{in}. Neka je centar p' *backup* centra p u rešenju $P \cup \{c_{in}\} \backslash \{p_c\}$. Kako ne važe (iii) i (iv), to je za svaki centar p (uključujući i c_{in}): $d(u_c, p) + d(p, p') \geq f(P)$, što je u kontradikciji sa činjenicom da je novo rešenje bolje od tekućeg rešenja P.

- Ukoliko je novi centar c_{in} u oblasti sa spoljašnje strane spoljašnje kružnice oblasti A_2, korisnik u_c za novi *reference* centar dobija jedan od centara $p \in P$ koji su na jednakoj udaljenosti od u_c kao i centar p'_c, uključujući i p'_c. Neka je centar p' *backup* centra p u rešenju $P \cup \{c_{in}\} \backslash \{p_c\}$. Kako ne važi (iv), to je za svaki centar p: $d(u_c, p) + d(p, p') \geq f(P)$. Prema tome, vrednost funkcije cilja nije smanjena, što je u kontradikciji sa pretpostavkom da je novo rešenje bolje od tekućeg.

Dakle, *reference* centar p_c može ostati otvoren ili biti zatvoren u novom rešenju iz $N_1(P)$ okoline. Novi centar tog rešenja se može naći sa unutrašnje strane kružnice C_1, na kružnici ili sa spoljašnje strane te kružnice, odnosno na spoljašnjoj kružnici oblasti A_2, unutar ili van te oblasti, pa prema tome, na osnovu 1) i 2) sledi ispravnost svojstva. ■

Sledeće svojstvo daje mogućnost da pojedina rešenja *p*-sledeći centar problema budu identifikovana kao globalno optimalana rešenja.

Svojstvo 2.6. Neka je kritični korisnik u_c istovremeno i jedan od centara tekućeg rešenja P, a $c1(u_c)$ njegov najbliži centar u rešenju P, ($u_c \in P$, $c1(u_c) \in P$ i $c1(u_c) \neq u_c$). Tada, ukoliko u skupu potencijalno novih centara ne postoji centar c_{in} ($c_{in} \notin P$) koji je bliži kritičnom korisniku u_c od centra $c1(u_c)$, tekuće rešenje P je globalno optimalno rešenje problema *p*-sledećeg centra.

Dokaz. Neka je $f(P)$ vrednost funkcije cilja tekućeg rešenja P, a $f(u_i)$ vrednost funkcije korisnika u_i. Uz to, $d\big(u_i, u_j\big)$ predstavlja najkraće rastojanje između čvorova u_i i u_j. Tada važi:

$$f(P) = max_{i=1,\dots,n} f(u_i) = f(u_c) = d(u_c, u_c) + d\big(u_c, c1(u_c)\big) = d\big(u_c, c1(u_c)\big). \quad (2.12)$$

Vrednost funkcije cilja je određena vrednošću funkcije kritičnog korisnika. Prema tome, da bi se smanjila vrednost funkcije cilja neophodno je u tekuće rešenje uključiti barem jedan novi centar c_{in} tako da je $d(u_c, c_{in}) < d\big(u_c, c1(u_c)\big)$. Ukoliko nije moguće pronaći takav centar, to znači da nije moguće unaprediti tekuće rešenje, tj. tekuće rešenje P je globalno optimalno rešenje problema p-sledećeg centra. ■

Algoritam 2.8: Provera da li je moguće pronaći potencijalno bolje rešenje nakon otvaranja novog centra c_{in}

ExistsRelaxedDistance(x_{cur}, c1, c2, u_c, c_{in})
$p_c = c1(u_c)$
$f = dist(u_c, p_c) + dist(p_c, c1(p_c))$

Property 2.5(i):
If $dist(c_{in}, u_c) \leq dist(p_c, u_c)$ *and* $dist(u_c, c_{in}) + dist(c_{in}, c1(c_{in})) < f$
    Return *True*
End If

Property 2.5(ii):
If $dist(p_c, c_{in}) < dist(p_c, c1(p_c))$
    Return *True*
End If

Property 2.5(ii):
If $dist(c2(u_c), u_c) = dist(p_c, u_c)$
    For Each $p_r \in \{x_{cur}(1), \dots, x_{cur}(p)\} \setminus \{p_c\}$
        If $dist(u_c, p_c) = dist(u_c, p_r)$ *and* $dist(p_r, c_{in}) < dist(p_c, c1(p_c))$
            Return *True*
        End If
    End For Each
End If

Property 2.5(iii) – p_c is deleted:
If $dist(c_{in}, u_c) \leq dist(c2(u_c), u_c)$
    $p = c1(c_{in})$ ***if*** $c1(c_{in}) \neq p_c$ ***else*** $c2(c_{in})$
    If $dist(u_c, c_{in}) + dist(c_{in}, p) < f$
        Return *True*
    End If
End If

Property 2.5(iv) – p_c is deleted:
If $dist(c2(u_c), u_c) \leq dist(c_{in}, u_c)$
    For Each $p_r \in \{x_{cur}(1), \dots, x_{cur}(p)\} \setminus \{p_c\}$
        If $dist(u_c, p_r) = dist(u_c, c2(u_c))$
            If $dist(u_c, p_r) + dist(p_r, c_{in}) < f$
                Return *True*
            End If
            $p_b = c1(p_r)$ ***if*** $c1(p_r) \neq p_c$ ***else*** $c2(p_r)$
            If $dist(u_c, p_r) + dist(p_r, p_b) < f$
                Return *True*
            End If
        End If
    End For Each
End If

Return *False*

U fazi lokalne pretrage, u svakoj iteraciji pokušava se smanjiti vrednost funkcije cilja tako što se jedan od centara tekućeg rešenja P zamenjuje novim centrom c_{in} ($c_{in} \notin P$). Opseg pretrage novih centara sužava se samo na centre koji potencijalno unapređuju tekuće rešenje, tj. smanjuju vrednost funkcije cilja kritičnog korisnika u_c. Na osnovu Svojstva 2.5, implementirana je procedura (Algoritam 2.8) koja proverava da li je otvaranjem novog centra moguće unaprediti tekuće rešenje. Algoritam se zasniva na istim pretpostavkama, pomoćnim strukturama i pozivu funkcija kao i algoritmi u prethodnim poglavljima.

Na osnovu Svojstva 2.6, dizajnirana je modifikacija sofisticiranog VNS algoritma iz Poglavlja 2.2, sposobna da u nekim slučajevima prepozna egzaktna rešenja problema *p*-sledećeg centra. Ideja je, pre *shaking* faze, proveriti da li je kritični korisnik uključen u tekuće rešenje i da li u tom slučaju postoji barem jedan novi centar koji potencijalno unapređuje tekuće rešenje. Ukoliko ne postoji, izvršenje se zaustavlja i algoritam vraća tekuće rešenje kao globalno optimalno (Algoritam 2.9). Izvršenje algoritma je takođe vremenski ograničeno vrednošću parametra t_{max}.

Algoritam 2.9: Sofisticirani VNS algoritam za prepoznavanje egzaktnih rešenja problema *p*-sledećeg centra

VariableNeighborhoodSearch(k_{max}, t_{max})
Initialization:
…
Repeat the Main step until the stopping condition is met (e.g., time ≤ t_{max})
Main step:
$k \leftarrow 1$
While $k \leq k_{max}$

If u_c*' is center*
$c_{in} \leftarrow$ ***find*** *center* ***from*** $\{x_{cur}(p+1), \ldots, x_{cur}(n)\}$
where $d(u_c', center) < d(u_c', c1'(u_c'))$
If c_{in} *is not found*
*****the optimal solution has been found*****
Return $\{x_{opt}(1), \ldots, x_{opt}(p)\}, f_{opt}$
End If
End If

Shaking operator:
*****generate a solution at random from kth neighborhood*****
For Each $j = 1, \ldots, k$
- *Take center to be inserted (c_{in}) if ExistsRelaxedDistance(x_{cur}, c1', c2', u_c', $x_{cur}(c_{in})$)*
- *Find center to be deleted (c_{out}) at random*
- *Update x_{cur}, c1' and c2', i.e., execute:*
  $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
  Update(x_{cur}, c1', c2')
- *Update f_{cur} and u_c' according to x_{cur}, c1' and c2'*

End For Each

Local search:
…

End While
Return $\{x_{opt}(1), \ldots, x_{opt}(p)\}, f_{opt}$

Da bi se ubrzala konvergencija ka lokalnom optimumu, u fazi lokalne pretrage vrši se filtriranje potencijalnih centara na osnovu poziva procedure *ExistsRelaxedDistance*. Očigledno je da promena složenosti *ExistsRelaxedDistance* procedure utiče na vreme izvršenja algoritma lokalne pretrage, kao što je i predstavljeno u svojstvima koja slede.

Svojstvo 2.7. Vremenska složenost *Move* algoritma (Algoritam 2.3) je $O(n + p^2)$.

Dokaz. Neka je:

- $P = \{p_1, p_2, \dots, p_p\}$ – tekuće rešenje p-sledeći centar problema,
- $V = \{v_1, v_2, \dots, v_n\}$ – skup svih korisnika,
- $P' = P \cup \{c_{in}\}$, $c_{in} \in V \backslash P$ – rešenje koje se dobija dodavanjem centra c_{in} tekućem rešenju,
- $P_i = P' \backslash \{p_i\} = P \cup \{c_{in}\} \backslash \{p_i\}$, $p_i \in P$ – rešenje koje se dobija brisanjem centra p_i iz rešenja P',
- $r(p_i, p_j)$, $p_i \in P'$, $p_j \in P'$ i $p_i \neq p_j$ – maksimum vrednosti funkcije cilja među svim korisnicima kojima su (p_i, p_j) dodeljeni kao *reference* i *backup* centri u rešenju P',
- $z(p_i)$, $p_i \in P$ – maksimum vrednosti funkcije cilja, nakon isključenja centra p_i, svih korisnika kojima je centar p_i dodeljen kao *reference* ili *backup* centar u rešenju P',
- $rc_X(v_j)$, $j = 1, \dots, n$ – *reference* centar korisnika v_j u rešenju $X \in \{P', P_i\}$,
- $c1_X(v_j)$, $j = 1, \dots, n$ – najbliži centar korisniku/centru v_j u rešenju $X \in \{P, P', P_i\}$,
- $c2_X(v_j)$, $j = 1, \dots, n$ – drugi-najbliži centar korisniku/centru v_j u rešenju $X \in \{P, P'\}$,
- $d(u, v)$ – najkraće rastojanje između čvorova u i v.

Posmatrajmo najpre skup centara P', tj. slučaj kada je novi centar c_{in} uključen u tekuće rešenje P, a da pri tom nije zatvoren niti jedan centar. Za svako $v_i \in V$, u rešenju P' važi:

$$c1_{P'}(v_i) = \begin{cases} c_{in}, \ if\ cndCP_1(v_i) \vee \left(cndCP_2(v_i) \wedge cndCP_3(v_i)\right) \\ c1_P(v_i), \ if\ cndCP_4(v_i) \vee \left(cndCP_5(v_i) \wedge cndCP_6(v_i)\right), \\ c2_P(v_i), \ otherwise \end{cases} \tag{2.13}$$

gde je:

$$cndCP_1(v_i) = d(v_i, c_{in}) < d\left(v_i, c1_P(v_i)\right),$$

$$cndCP_2(v_i) = \left[d(v_i, c_{in}) = d\left(v_i, c1_P(v_i)\right)\right] \wedge d\left(c_{in}, c1_P(c_{in})\right) < min\left\{d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right), d(c1_P(v_i), c_{in})\right\},$$

$$cndCP_3(v_i) = d(v_i, c_{in}) < d\left(v_i, c2_P(v_i)\right) \vee d\left(c_{in}, c1_P(c_{in})\right) < min\left\{d\left(c2_P(v_i), c1_P\left(c2_P(v_i)\right)\right), (c2_P(v_i), c_{in})\right\},$$

$$cndCP_4(v_i) = d\left(v_i, c1_P(v_i)\right) < d\left(v_i, c2_P(v_i)\right),$$

$$cndCP_5(v_i) = \left[d\left(v_i, c1_P(v_i)\right) = d\left(v_i, c2_P(v_i)\right)\right],$$

$$cndCP_6(v_i) = min\left\{d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right), d(c1_P(v_i), c_{in})\right\} \leq min\left\{d\left(c2_P(v_i), c1_P\left(c2_P(v_i)\right)\right), (c2_P(v_i), c_{in})\right\}.$$

Tada je:

$$rc_{P'}(v_i) = \begin{cases} c1_{P'}(v_i), v_i \notin P' \\ v_i, v_i \in P' \end{cases}. \tag{2.14}$$

Pretpostavimo da je:

1) $c1_{P'}(v_i) = c_{in}$, tada je:

$$c2'_{P'}(v_i) = \begin{cases} c1_P(v_i), \ cndCP'_1(v_i) \vee \left(cndCP'_2(v_i) \wedge cndCP'_3(v_i)\right) \\ c2_P(v_i), \ otherwise \end{cases}, \tag{2.15}$$

gde je:

$$cndCP'_1(v_i) = d\left(v_i, c1_P(v_i)\right) < d\left(v_i, c2_P(v_i)\right),$$

$$cndCP'_2(v_i) = \left[d\left(v_i, c1_P(v_i)\right) = d\left(v_i, c2_P(v_i)\right)\right],$$

$$cndCP'_3(v_i) = min\left\{d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right), d(c1_P(v_i), c_{in})\right\} \leq min\left\{d\left(c2_P(v_i), c1_P\left(c2_P(v_i)\right)\right), (c2_P(v_i), c_{in})\right\}.$$

2) $c1_{P'}(v_i) = c1_P(v_i)$:

$$c2''_{P'}(v_i) = \begin{cases} c_{in}, \ cndCP''_1(v_i) \vee cndCP''_2(v_i) \\ c2_P(v_i), \ otherwise \end{cases}, \tag{2.16}$$

gde je:

$$cndCP''_1(v_i) = d(v_i, c_{in}) < d\left(v_i, c2_P(v_i)\right),$$

$$cndCP''_2(v_i) = \left[d(v_i, c_{in}) = d\left(v_i, c2_P(v_i)\right)\right] \wedge d\left(c_{in}, c1_P(c_{in})\right) < min\left\{d\left(c2_P(v_i), c1_P\left(c2_P(v_i)\right)\right), (c2_P(v_i), c_{in})\right\}.$$

3) $c1_{P'}(v_i) = c2_P(v_i)$:

$$c2'''_{P'}(v_i) = \begin{cases} c_{in}, \ cndCP'''_1(v_i) \wedge cndCP'''_2(v_i) \\ c1_P(v_i), \ otherwise \end{cases}, \tag{2.17}$$

gde je:

$$cndCP'''_1(v_i) = \left[d(v_i, c_{in}) = d\left(v_i, c1_P(v_i)\right)\right],$$

$$cndCP'''_2(v_i) = d\left(c_{in}, c1_P(c_{in})\right) < min\left\{d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right), (c1_P(v_i), c_{in})\right\}.$$

Iz (2.15), (2.16) i (2.17) sledi:

$$c2_{P'}(v_i) = \begin{cases} c2'_{P'}(v_i),\ c1_{P'}(v_i) = c_{in} \\ c2''_{P'}(v_i),\ c1_{P'}(v_i) = c1_P(v_i). \\ c2'''_{P'}(v_i),\ c1_{P'}(v_i) = c2_P(v_i) \end{cases} \tag{2.18}$$

Na osnovu prethodnog sledi:

$$r\left(p_j, c1_{P'}(p_j)\right) = max_{v_i \in V}\left\{d\left(v_i, p_j\right) \mid p_j = rc_{P'}(v_i)\right\} + d\left(p_j, c1_{P'}(p_j)\right). \tag{2.19}$$

Posmatrajmo sada skup P_i, tj. rešenje koje se dobija kada se iz skupa P' isključi centar p_i. Upoređujući rešenja P' i P_i, skup korisnika V se deli na dva disjunktna podskupa:

- V_i', korisnici koji u rešenju P_i zadržavaju svoj par centara iz rešenja P':

$z'(p_i) = 0,$ (2.20)

- V_i'', korisnici koji nakon zatvaranja centra p_i gube svoj *reference* i/ili *backup* centar iz rešenja P':

$$z''(p_i) = \max_{v_j \in V_i''}\left\{d\left(v_j, rc_{P_i}(v_i)\right) + d\left(rc_{P_i}(v_i), c1_{P_i}\left(rc_{P_i}(v_i)\right)\right)\right\}. \tag{2.21}$$

Iz (2.20) i (2.21) sledi:

$$z(p_i) = max\{z'(p_i), z''(p_i)\} = z''(p_i) = \max_{v_j \in V_i''}\left\{d\left(v_j, rc_{P_i}(v_i)\right) + d\left(rc_{P_i}(v_i), c1_{P_i}\left(rc_{P_i}(v_i)\right)\right)\right\}. \tag{2.22}$$

Takođe, za svako $v_i \in V$, u rešenju P_i važi:

$$c1_{P_i}(v_i) = \begin{cases} c1_{P'}(v_i),\ cndCP_1^{iv}(v_i) \vee \left(cndCP_2^{iv}(v_i) \wedge \left(cndCP_3^{iv}(v_i) \vee cndCP_4^{iv}(v_i)\right)\right), \\ c2_{P'}(v_i),\ otherwise \end{cases} \tag{2.23}$$

gde je:

$cndCP_1^{iv}(v_i) = [c2_{P'}(v_i) = p_i],$

$cndCP_2^{iv}(v_i) = c1_{P'}(v_i) \neq p_i,$

$cndCP_3^{iv}(v_i) = d\left(v_i, c1_{P'}(v_i)\right) < d\left(v_i, c2_{P'}(v_i)\right),$

$cndCP_4^{iv}(v_i) = d\left(c1_{P'}(v_i), bc_{P_i}\left(c1_{P'}(v_i)\right)\right) < d\left(c2_{P'}(v_i), bc_{P_i}\left(c2_{P'}(v_i)\right)\right),\ bc_{P_i}(c) = \begin{cases} c1_{P'}(c), c1_{P'}(c) \neq p_i \\ c2_{P'}(c), otherwise \end{cases},\ c \in \{c1_{P'}(v_i), c2_{P'}(v_i)\}.$

Tada je:

$$rc_{P_i}(v_i) = \begin{cases} c1_{P_i}(v_i), v_i \notin P_i \\ v_i, v_i \in P_i \end{cases}. \tag{2.24}$$

Konačno, na osnovu prethodnih izraza, *Best deletion* se predstavlja kao:

$$\begin{aligned} f\left(P^{(best)}\right) &= \min_{i=1,\dots,p} f(P_i) \\ &= \min_{i=1,\dots,p}\left\{\max_{v_j \in V}\left\{d\left(v_j, rc_{P_i}(v_j)\right) + d\left(rc_{P_i}(v_j), c1_{P_i}\left(rc_{P_i}(v_j)\right)\right)\right\}\right\} \\ &= \min_{i=1,\dots,p}\left\{\max_{v_j \in (V_i' \cup V_i'')}\left\{d\left(v_j, rc_{P_i}(v_j)\right) + d\left(rc_{P_i}(v_j), c1_{P_i}\left(rc_{P_i}(v_j)\right)\right)\right\}\right\} \\ &= \min_{i=1,\dots,p}\left\{max\left\{\begin{matrix}\max_{v_j \in V_i'}\left\{d\left(v_j, rc_{P_i}(v_i)\right) + d\left(rc_{P_i}(v_i), c1_{P_i}\left(rc_{P_i}(v_i)\right)\right)\right\}, \\ \max_{v_j \in V_i''}\left\{d\left(v_j, rc_{P_i}(v_i)\right) + d\left(rc_{P_i}(v_i), c1_{P_i}\left(rc_{P_i}(v_i)\right)\right)\right\}\end{matrix}\right\}\right\} \\ &= \min_{i=1,\dots,p}\left\{max\left\{\begin{matrix}\max_{p_j \in P \cup \{c_{in}\} \setminus \{p_i\}, c1_{P'}(p_j) \neq p_i}\left\{r\left(p_j, c1_{P'}(p_j)\right)\right\}, \\ z(p_i)\end{matrix}\right\}\right\} \\ &= \min_{p_i \in P}\left\{max\left\{z(p_i), \max_{p_j \in P \cup \{c_{in}\} \setminus \{p_i\}, c1_{P'}(p_j) \neq p_i}\left\{r\left(p_j, c1_{P'}(p_j)\right)\right\}\right\}\right\}. \end{aligned} \tag{2.25}$$

Poslednji izraz opisuje proceduru predstavljenu *Best deletion* korakom u Algoritmu 2.3.5.

Iz (2.13), (2.14), (2.23) i (2.24) sledi da se $c1_X(v_i)$ i $rc_X(v_i)$ vrednosti, gde $X \in \{P', P_i\}$, pronalaze u konstantnoj vremenskoj složenosti. Na osnovu toga, jednostavno je primetiti da izračunavanja r i z vrednosti u (2.19) i (2.22) imaju linearnu vremensku kompleksnost u odnosu na broj korisnika, $O(n)$. *Best deletion* korak se izvršava za $O(p^2)$, pa prema tome ukupna vremenska složenost *Move* procedure je $O(n + p^2)$. ■

Svojstvo 2.8. Vremenska složenost jedne iteracije *LocalSearchVertexSubstitution* algoritma (Algoritam 2.5) u najgorem slučaju je $O(n^2 + p^2 n)$.

Dokaz. Da bi se odredila vremenska složenost iteracije *LocalSearchVertexSubstitution* algoritma, najpre je potrebno utvrditi složenost ostalih procedura. Na osnovu Svojstva 2.7, vremenska složenost *Move* procedure je $O(n + p^2)$. *Update* procedura (Algoritam 2.4) ažurira nizove *c1* i *c2* dužine *p* za svakog od *n* korisnika tako da vremenska složenost iznosi $O(pn)$. Vremenska složenost *ExistsRelaxedDistance* procedure je $O(p)$. *LocalSearchVertexSubstitution* algoritam u najgorem slučaju poziva *ExistsRelaxedDistance* i *Move* procedure $n - p$ puta i jednom *Update* proceduru, pa je prema tome vremenska složenost u najgorem slučaju $O(n^2 + p^2 n) + O(pn) + O(n) \approx O(n^2 + p^2 n)$. ■

Posledica. Ukoliko je $p << n$, vremenska složenost jedne iteracije *LocalSearchVertexSubstitution* algoritma u najgorem slučaju je $O(n^2)$.

Na osnovu Svojstava 2.4 i 2.8 sledi da modifikovani algoritam povećava samo konstantan faktor vremenske složenosti. Asimptotska složenost jedne iteracije *LocalSearchVertexSubstitution* procedure ostaje $O(n^2 + p^2 n)$ u najgorem slučaju.

2.5. Rezultati testiranja algoritma za problem *p*-sledećeg centra

U ovom poglavlju su predstavljeni rezultati testiranja predloženog VNS algoritma za problem *p*-sledećeg centra. Algoritam je implementiran u C++ programskom jeziku, a sva testiranja izvedena na *Intel Core i7-8700K (3.7 GHz) CPU* sa *32GB RAM*-a konfiguraciji. Proces testiranja je sproveden u tri eksperimentalne faze.

- Prvi deo eksperimenta je imao za cilj da proceni performanse predloženog sofisticiranog algoritma u poređenju sa jednostavnom VNS implementacijom iz rada [31]. Za tu svrhu, iskorišćene su referentne test instance za *p*-sledeći centar problem iz poznate *OR-Library* [2] test platforme. Test set sadrži 40 instanci sa *n* (broj korisnika) koje varira od 100 do 900 i *p* (broj izabranih centara) između 5 i 200.

- U drugom delu eksperimenta upoređuje se predloženi sofisticirani VNS sa *state-of-the-art* heuristikama iz literature, tj. hibridnom GRASP-VNS verzijom iz [21]. U ovom slučaju, testiranja su izvršena nad prilagođenim *OR-Library* test setom koji je iskorišćen i u [21]. Ovaj skup podataka sadrži test instance izvedene iz *OR-Library* primera, pmed1-pmed4 i pmed6-pmed8, tako što je uzeto u obzir samo prvih *n* čvorova iz pomenutih primera. Najveće instance sadrže 200 čvorova.

- Poslednji deo eksperimenta je sproveden sa ciljem da se ocene performanse predloženog algoritma primenjenog nad većim instancama *p*-sledeći centar problema i grafovima različitih gustina. Generisana su dva nova skupa test primera sa većim brojem čvorova *n* i većim gustinama od prethodno korišćenih test instanci. Prvi skup sadrži 44 instance sa *n* koje varira od 1000 do 2500 i vrednosti *p* između 5 i 200. Drugi set podataka sadrži 48 test instanci (500 - 1000 čvorova) definisanih nad grafovima gustine od 50% do 80% i vrednosti *p* između 5 i 200. Ovo su test setovi koji su po prvi put predstavljeni u literaturi, a samim tim i za testiranje problema *p*-sledećeg centra.

Algoritam je nad svakom test instancom izvršen po 20 puta, svaki put počev od različitog inicijalnog rešenja. U pogledu podešavanja parametara, predložena VNS heuristika ima dva formalna parametra: k_{max} – maksimalan *shaking* nivo i t_{max} – maksimalno dozvoljeno *CPU* vreme izvršenja. Isprobane su različite kombinacije parametara $k_{max} \in \{p/4, p/2, p\}$ i $t_{max} \in \{n, 2n\}$. Ispostavilo se da su rezultati neznatno bolji sa većim k_{max} vrednostima. Sa druge strane, algoritam je uglavnom pronalazio najbolje rešenje mnogo pre isteka *CPU* vremenskog ograničenja. Otuda, odlučeno je da se prezentuju rezultati samo za $k_{max} = p$ i $t_{max} = n$ sekundi.

2.5.1. Rezultati testiranja nad *OR-Library* instancama

U ovom poglavlju je predstavljeno poređenje sofisticirane VNS heuristike i jednostavnijih implementacija bazičnog metoda promenljivih okolina. Bazična VNS heuristika je predstavljena u Algoritmu 1.1: ne koristi pomoćne strukture niti sofisticiranu lokalnu pretragu i *shaking* proceduru. Drugim rečima, u koraku lokalne pretrage, VNS ispituje sva moguća rešenja u $N_1(P)$ okolini, dok u iteraciji *shaking* procedure generiše potpuno slučajno rešenje iz $N_k(P)$ okoline. Nasuprot tome, sofisticirana VNS implementacija, predstavljena u Algoritmu 2.7, se oslanja na pomoćne strukture podataka i sofisticirane algoritam lokalne pretrage i *shaking* proceduru. Parametri za testiranje svih novih VNS varijanti su postavljeni na iste vrednosti ($k_{max} = p$ i $t_{max} = n$ sekundi) u svim testovima.

Tabela 2.1 prikazuje sumarne rezultate dobijene testiranjem sofisticiranog VNS-a nad *OR-Library* test instancama, dok su detaljni rezultati dostupni u Dodatku A. U sumarnim rezultatima, instance sa istim brojem čvorova su posmatrane kao jedan test slučaj. Prva kolona u Tabeli 2.1 daje nazive test instanci, dok naredne dve kolone predstavljaju broj čvorova (korisnika) *n* i broj grana *m* u svakom od test primera. U kolonama "*Best Value*", "*AVG Value*" i "*Worst Value*" prikazuje se prosek najboljih, prosečnih i najgorih vrednosti dobijenih rešenja u 20 izvršenja predloženog algoritma. Kolona "*Time*" sadrži prosečno vreme pronalaska najboljeg rešenja (*eng. time-to-target*). *Time-to-target* se odnosi na *CPU* vreme u sekundama, potrebno VNS implementaciji da pronađe rešenje predstavljeno na izlazu. Kolona "*#Best*" sadrži broj koliko puta je algoritam pronašao najbolje poznato rešenje. Kako izvorne *OR-Library* instance u literaturi nisu široko prisutne u testiranju problema p-sledećeg centra, kao najbolje poznato rešenje uzeto je najbolje rešenje dobijeno predloženim VNS algoritmom. Poslednje dve kolone predstavljaju prosečno procentualno odstupanje (*eng. gap*) srednjih i najgorih vrednosti rešenja u odnosu na vrednosti najboljih rešenja datih u koloni "*Best Value*".

Tabela 2.1. Rezultati za sofisticirani VNS algoritam testiran nad *OR-Library* test instancama

	N	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1-pmed5	100	200	131.00	131.00	131.00	0.60	20.00	0.00	0.00
pmed6-pmed10	200	800	82.60	82.60	82.60	6.00	20.00	0.00	0.00
pmed11-pmed15	300	1800	58.60	58.60	58.60	4.39	20.00	0.00	0.00
pmed16-pmed20	400	3200	44.40	44.47	44.60	36.53	18.60	0.43	0.15
pmed21-pmed25	500	5000	41.20	41.21	41.40	28.54	19.80	0.42	0.02
pmed26-pmed30	600	7200	43.60	43.62	43.80	29.75	19.60	0.53	0.05
pmed31-pmed34	700	9800	42.50	42.69	42.75	55.14	16.25	1.14	0.85
pmed35-pmed37	800	1280	37.00	37.00	37.00	10.47	20.00	0.00	0.00
pmed38-pmed40	900	16200	45.67	45.67	45.67	17.87	20.00	0.00	0.00
AVG						**20.86s**	**19.38**	**0.29%**	**0.11%**

Na osnovu rezultata predstavljenih u Tabeli 2.1 i Tabeli A.1 iz Dodatka A, primećuje se da je većina manjih instanci (do $n = 300$ čvorova) veoma brzo rešena. Samo je za 3 od 15 instanci sa $n \leq 300$ prosečno vreme pronalaska najboljeg rešenja bilo veće od 2 sekunde. Kao što je i očekivano, sa porastom veličine instanci raste i prosečno vreme potrebno da se pronađe najbolje rešenje. Prosečno, *time-to-target* nad kompletnim test skupom je samo 20.86 sekundi. Sa druge strane, u pogledu kvaliteta rešenja, najbolja poznata rešenja nisu pronađena u svih 20 izvršenja samo za 4 od ukupno 40 instanci. Štaviše, samo za dve instance (pmed17 i pmed33), najbolje rešenje je pronađeno manje od osamnaest puta. Sveukupno, sofisticirani VNS je pronašao najbolja poznata rešenja u 19.38 od 20 slučajeva u proseku, ili u 96.90%. U pogledu prosečnog procentualnog odstupanja srednjih i najgorih vrednosti pronađenih rešenja, može se zaključiti da ne postoji značajan jaz između

najgorih/prosečnih i najboljih rešenja, samo 0.29% i 0.11% u proseku, respektivno. Odstupanja su nešto veća (4.55% i 3.41%) samo za pmed33, ali sa druge strane, odstupanja najgorih i prosečnih rešenja drugih instanci nisu veća od 2.63% i 0.76%, što je sasvim prihvatljivo.

Poređenje sa jednostavnom VNS implementacijom. Da bi se verifikovao kvalitet predloženog VNS algoritma, implementiran je jednostavan VNS, tj. faza lokalne pretrage predloženog algoritma je zamenjena jednostavnom procedurom koja sekvencijalno pretražuje sva moguća rešenja i pronalazi najbolje. Algoritam je takođe izvršen 20 puta sa istim hiper-parametrima nad istim test setom podataka, i rezultati su prezentovani u Tabeli 2.2. Za VNS koji koristi jednostavan algoritam lokalne pretrage, predstavljeni su srednja vrednost najboljih rešenja u 20 izvršenja (kolona "*Best Found Value*") i prosečno vreme do pronalaska najboljeg rešenja ("*Time*") za svaki od test primera. Kolona "*#Best Known*" sadrži ukupan broj izvršenja bazičnog VNS-a (sa jednostavnom lokalnom pretragom) koja su rezultirala pronalaskom najboljeg poznatog rešenja, identifikovanog VNS implementacijom sa sofisticiranom lokalnom pretragom. Konačno, poslednja kolona daje procentualno odstupanje vrednosti najboljeg rešenja pronađenog bazičnim VNS-om sa jednostavnom lokalnom pretragom od vrednosti najboljeg poznatog rešenja sofisticiranog VNS algoritma. Procentualno odstupanje je izračunato kao $\frac{Best\ found - Best\ known}{Best\ known} * 100$.

Tabela 2.2. Bazični VNS sa jednostavnom procedurom lokalne pretrage

	N	*Best Known Value*	*Best Found Value*	*Time*	*#Best Known*	*Gap (best found-best known) /best known *100*
pmed1-pmed5	100	131.00	131.00	6.16	20.00	0.00
pmed6-pmed10	200	82.60	82.60	44.86	17.40	0.00
pmed11-pmed15	300	58.60	58.80	117.64	15.00	0.43
pmed16-pmed20	400	44.40	46.20	236.24	7.80	5.43
pmed21-pmed25	500	41.20	42.80	329.46	8.60	4.96
pmed26-pmed30	600	43.60	43.60	343.43	12.40	0.00
pmed31-pmed34	700	42.50	45.25	309.36	15.00	12.50
pmed35-pmed37	800	37.00	37.00	283.79	14.00	0.00
pmed38-pmed40	900	45.67	47.67	350.77	13.33	8.70
AVG				**213.25s**	**13.70**	**3.26%**

Na osnovu prezentovanih rezultata sledi da bazični VNS sa jednostavnom procedurom lokalne pretrage ne proizvodi zadovoljavajuće rezultate. Algoritam nije pronašao najbolja poznata rešenja za 7 od 40 instanci, a u proseku najbolje poznato rešenje je pronađeno samo u 13.70 od 20 slučajeva. Ispostavilo se da su vrednosti najboljih rešenja pronađenih sofisticiranim algoritmom bolje za 3.26% u proseku. U četiri od ukupno 9 test slučajeva prosečna odstupanja su čak veća od 4%. Takođe, prosečno vreme pronalaska najboljeg rešenja je poraslo više od deset puta (213.25 sekundi u Tabeli 2.2 u odnosu na 20.86 sekundi iz Tabele 2.1). Sve ove činjenice ukazuju na benefite korišćenja sofisticirane umesto jednostavne VNS implementacije: znatno bolja rešenja za drastično kraće vreme.

Poređenje sa Filtriranim VNS metodom iz rada [31]. Dobijeni rezultati za *OR-Library* instance su takođe upoređeni sa dostupnim rezultatima algoritma iz rada [31]. Autori u radu [31] predlažu implementaciju istog VNS okvira sa jednostavnom procedurom lokalne pretrage koja filtrira rešenja na osnovu neposredno izračunate vrednosti funkcije cilja. Različito od procedure iz rada [31], novo-predložena procedura dinamički evaluira vrednost funkcije cilja, i na taj način unapređuje efikasnost algoritma lokalne pretrage. Kalkulacije

koriste pomoćne nizove i međurezultate koji su zasnovani na teorijskoj diskusiji iz rada [28], prilagođenoj *p*-sledeći centar problemu i prezentovanoj kroz Svojstva 2.1 – 2.4. Dodatno, novo-predloženi VNS algoritam koristi i naprednu *shaking* proceduru predstavljenu u Algoritmu 2.6.

Tabela 2.3. Poređenje sa Filtriranim VNS metodom iz rada Ristića i sar.

	N	*Best Known Value*	*Best Found Value*	*Time*	*#Best Known*	*Gap (best found-best known) /best known *100*
pmed1-pmed5	100	131.00	131.00	80.93	15.00	0.00
pmed6-pmed10	200	82.60	83.20	227.96	7.60	0.74
pmed11-pmed15	300	58.60	59.60	539.59	11.20	2.13
pmed16-pmed20	400	44.40	46.40	838.37	2.00	5.80
pmed21-pmed25	500	41.20	44.20	807.55	5.00	8.69
pmed26-pmed30	600	43.60	44.00	688.50	11.20	1.05
pmed31-pmed34	700	42.50	45.25	689.07	11.75	12.50
pmed35-pmed37	800	37.00	37.00	928.67	10.00	0.00
pmed38-pmed40	900	45.67	47.67	843.21	12.00	8.70
AVG				**599.66s**	**9.32**	**4.20%**

Kao i algoritam sa jednostavnom lokalnom pretragom, Filtrirani VNS metod iz rada [31] se nije pokazao uspešnim u poređenju sa novom VNS implementacijom. Ispostavilo se da je algoritam iz rada [31] dao čak i lošije rezultate u odnosu na jednostavan VNS. Nije uspeo da pronađe najbolja poznata rešenja za 11 od 40 instanci, i u proseku je identifikovao najbolje poznato rešenje samo u 9.32 od 20 slučajeva. Najbolja rešenja pronađena sofisticiranom VNS implementacijom su bolja za 4.20% u proseku. Takođe, prosečno vreme pronalaska najboljeg rešenja Filtriranog VNS metoda [31] je značajno veće, 599.66 sekundi u poređenju sa vremenom sofisticiranog VNS algoritma od 20.86 sekundi. Dakle, rezultati poređenja su ponovo potvrdili benefite korišćenja sofisticirane implementacije: mnogo bolja rešenja za znatno kraće vreme.

2.5.2. Rezultati poređenja sa hibridnom verzijom algoritma iz rada Lopez-Sancheza i sar.

U ovom odeljku se upoređuju rešenja pronađena predloženim algoritmom i algoritmom iz rada [21]. Lopez-Sanchez i sar. u radu [21] predlažu tri heuristička algoritma za problem *p*-sledećeg centra: GRASP, VNS i hibridni GRASP-VNS algoritam, koji kombinuje dva prethodno pomenuta. Iako su GRASP i VNS mnogo brži, nisu uporedivi sa hibridnom verzijom u pogledu kvaliteta rešenja. Stoga, u nastavku su prikazani samo rezultati poređenja u disertaciji predloženog algoritma sa hibridnom verzijom algoritma iz rada [21].

Poređenje rezultata algoritama je prezentovano u Tabelama 2.4 i 2.5 i Tabelama A.3 – A.6 u Dodatku A. Tabele 2.4 i 2.5 daju sumarne rezultate test instanci sa istim brojem čvorova. Kao i ranije, instance sa istim brojem korisnika su grupisane u jedan test slučaj. Dodatno u odnosu na prethodno objašnjene nazive kolona, nove tabele imaju i kolone koje sadrže procentualno odstupanje vrednosti najboljeg, srednjeg i najgoreg rešenja predložene VNS implementacije u odnosu na hibridni GRASP-VNS algoritam. Računaju se kao $\frac{value - GRASP\_VNS}{GRASP\_VNS} * 100$, gde je "*value*" vrednost najboljeg, srednjeg ili najgoreg rešenja predloženog VNS algoritma. Negativna vrednost odstupanja ukazuje na to da je sofisticirani algoritam pronašao bolje rešenje od hibridnog GRASP-VNS-a. Kolona "*GRASP-VNS Value*" sadrži vrednost rešenja pronađenog hibridnim

algoritmom iz rada [21]. Poslednja kolona "*#New Best Known*" predstavlja broj test instanci za koje je u disertaciji predloženi VNS algoritam uspeo da ustanovi novo najbolje poznato rešenje.

Tabela 2.4. Poređenje sa hibridnim algoritmom Lopez-Sancheza i sar. za manje instance problema

		Sophisticated VNS				
N	*GRASP-VNS Value*	*Time*	*Best VNS*	*#Best*	*Gap Best Vs GRASP-VNS*	*#New Best Known*
10	103.75	0.005	102.00	20	-1.37	1
20	118.38	0.010	118.38	20	0.00	0
30	134.25	0.080	134.25	20	0.00	0
40	119.75	0.130	119.75	20	0.00	0
50	108.50	0.210	108.38	20	-0.11	1
AVG		0.100s		20	-0.16%	***Total: 2***

Za manje instance problema ($n \leq 50$), algoritam je uspeo da pronađe najbolja poznata rešenja u svakom od 20 izvršenja za 0.10 sekundi u proseku. U poređenju sa hibridnim GRASP-VNS algoritmom, predloženi VNS je identifikovao ista rešenje sem za par instanci (pmed1_09 i pmed2_01), za koje je novi algoritam pronašao bolja rešenja. Ispostavilo se da su vrednosti rešenja pronađenih sofisticiranim VNS-om prosečno bolje za 0.16%.

Tabela 2.5. Poređenje sa hibridnim algoritmom Lopez-Sancheza i sar. za veće instance problema

		Sophisticated VNS								
N	*GRASP-VNS Value*	*Time*	*Worst Value*	*AVG Value*	*Best Value*	*#Best*	*Gap Worst Vs GRASP-VNS*	*Gap AVG Vs GRASP-VNS*	*Gap Best Vs GRASP-VNS*	*#New Best Known*
60	102.33	0.14	102.17	102.16	102.08	18.42	-0.18	-0.19	-0.25	2
70	105.00	0.54	105.00	105.00	105.00	20.00	0.00	0.00	0.00	0
80	110.33	3.31	109.00	109.00	109.00	20.00	-1.32	-1.32	-1.32	5
90	107.25	2.94	107.00	106.75	106.69	19.00	-0.26	-0.49	-0.54	5
100	108.88	3.20	108.38	108.03	107.94	19.38	-0.59	-0.90	-0.98	5
150	64.00	5.89	62.08	61.99	61.75	17.25	-2.89	-3.01	-3.35	7
200	68.83	19.71	62.58	62.57	62.50	18.67	-9.40	-9.42	-9.51	10
AVG		4.93s				18.98	-1.95%	-2.06%	-2.15%	***Total: 34***

Razmatrajući primere većih problema, primetno je da algoritam nije bio tako uspešan kao u slučaju originalnih *OR-Library* instanci u pogledu broja izvršenja koja su rezultovala pronalaskom najboljeg rešenja, ali je svakako potvrdio svoju stabilnost. Bilo je nekoliko instanci za koje je najbolje poznato rešenje pronađeno samo jednom, tri ili četiri puta u 20 izvršenja (pmed4_01, pmed6_02, pmed6_05 i pmed8_01). Sa druge strane, algoritam je bio uspešan u 94.89% slučajeva, tj. pronašao je najbolje poznato rešenje prosečno u 18.98 od 20 izvršenja. Nema značajnih razlika u vrednostima najgorih, srednjih i najboljih rešenja, što ukazuje na visoku stabilnost predloženog VNS algoritma. U poređenju sa hibridnim GRASP-VNS algoritmom, pronašao je isto ili bolje rešenje za svaku test instancu u proseku za 4.93 sekunde. Bilo je nekoliko test instanci koje su rezultovale pozitivnim odstupanjem najgoreg i srednjeg rešenja u odnosu na rešenje dobijeno GRASP-VNS algoritmom. Ipak, u proseku, čak i najgora rešenja su pouzdanija od rešenja GRASP-VNS algoritma. Svakako, najbolje rešenje predloženog algoritma je često značajno bolje od rešenja hibridnog GRASP-VNS algoritma. Sofisticirani VNS je ukupno uspostavio nova najbolja poznata rešenja za 34 od 92 test primera. Shodno tome, može se zaključiti da je predloženi algoritam veoma pouzdan i da proizvodi bolje rezultate od trenutno *state-of-the-art* GRASP-VNS algoritma. Ovo još jednom pokazuje da sofisticiranija implementacija algoritma može dati bolje rezultate od jednostavnog hibridnog pristupa.

Trebalo bi napomenuti da u radu [21] autori nisu prikazali vreme pronalaska najboljeg rešenja za svoje algoritme. Iz tog razloga, nije bilo moguće uporediti ovo vreme za sofisticirani VNS i hibridni GRASP-VNS algoritam. Sa druge strane, ukupno vreme izvršenja hibridnog algoritma nad instancama sa 200 čvorova je oko 1200 sekundi, 400 sekundi nad instancama sa 150 čvorova i 150 sekundi nad instancama sa 100 čvorova (vidi [21]). Kako je vremensko ograničenje izvršenja novo-predložene VNS implementacije postavljeno na *n* sekundi, to implicira da uz slično ili manje vremensko ograničenje sofisticirani VNS proizvodi bolje rezultate od hibridnog GRASP-VNS algoritma.

2.5.3. Rezultati testiranja nad većim instancama problema

Da bi se dalje procenile performanse sofisticiranog VNS algoritma, generisane su veće test instance sa 1000, 1500, 2000 i 2500 čvorova kao i instance definisane nad grafovima različitih gustina sa do 1000 čvorova. Novi test primeri su generisani kao *random k-regularni* grafovi sa predefinisanim brojem čvorova i grana. Algoritam je izvršen po 20 puta nad svakom test instancom i rezultati su prezentovani u tabelama koje slede. U dodatku prethodno objašnjenih kolona, Tabela 2.6 sadrži novu kolonu koja pokazuje gustinu grafa nad kojim je test instanca definisana.

Tabela 2.6. Rezultati za test instance definisane nad grafovima sa različitim gustinama

N	*Density*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (AVG-best) /best*100*
500	50.10	5.50	5.50	5.50	6.55	20.00	0.00	0.00
500	60.12	4.75	4.75	4.75	8.46	20.00	0.00	0.00
500	80.16	4.25	4.25	4.25	5.47	20.00	0.00	0.00
600	50.08	5.00	5.23	5.25	18.88	15.50	6.25	5.63
600	60.10	4.50	4.50	4.50	26.35	20.00	0.00	0.00
600	80.13	4.25	4.25	4.25	5.62	20.00	0.00	0.00
800	50.06	4.50	4.56	4.75	95.23	18.75	5.00	1.25
800	60.08	4.25	4.25	4.25	24.39	20.00	0.00	0.00
800	80.10	4.25	4.25	4.25	13.57	20.00	0.00	0.00
1000	50.05	4.25	4.25	4.25	69.45	20.00	0.00	0.00
1000	60.06	4.25	4.25	4.25	36.76	20.00	0.00	0.00
1000	80.08	4.00	4.00	4.00	40.45	20.00	0.00	0.00
AVG					**29.27s**	**19.52**	**0.94%**	**0.57%**

Sudeći po rezultatima iz Tabele 2.6, evidentno je da gustina grafa ne utiče na tačnost i efikasnost algoritma. Algoritam pronalazi najbolje rešenje u 19.52 od 20 izvršenja u proseku. Za samo dve instance (rnddnskreg15 i rnddnskreg26) algoritam nije pronašao najbolje rešenje u svakom od 20 izvršenja. Srednje vreme potrebno da se pronađe najbolje rešenje je bilo 29.27 sekundi.

Tabela 2.7. Rezultati za velike test instance

N	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (AVG-best) /best*100*
1000	50000	9.73	9.93	10.09	185.37	15.91	4.33	2.46
1500	112500	8.27	8.46	8.64	293.69	16.27	4.57	2.44
2000	200000	7.36	7.69	7.91	502.32	13.45	8.12	4.98
2500	312500	6.82	7.09	7.18	709.75	14.55	5.03	3.74
AVG					**422.78s**	**15.05**	**5.51%**	**3.41%**

Tabela 2.7 pokazuje da algoritam primenjen nad velikim instancama problema nije bio tako uspešan kao u drugim primerima. Algoritam pronalazi najbolje rešenje prosečno u 15.05 od 20 izvršenja. Bilo je nekoliko instanci za koje je algoritam pronašao najbolje rešenje samo jednom ili dva puta, ali apsolutno odstupanje najgorih od najboljih rešenja nije bilo veće od 1. Prosečno vreme pronalaska najboljeg rešenja je bilo 422.78 sekundi. Sve u svemu, da se zaključiti da je ponašanje novog sofisticiranog VNS algoritma u proseku veoma zadovoljavajuće.

Detaljni test rezultati se mogu naći na: *https://1drv.ms/b/s!AnaX64fUELcf6AgDcszqeNPfcbIX?e=q0n4hU* i *https://1drv.ms/b/s!AnaX64fUELcf6Alk6HM9Wc-8hv6v?e=enR3NS*.

2.6. Egzaktna rešenja *OR-Library* test instanci za problem *p*-sledećeg centra

Osamdesetih godina prošlog veka, jedan od problema u oblasti operacionih istraživanja je bio standardizacija skupa podataka za testiranje implementiranih algoritama. Da bi bilo moguće uporediti efikasnost različitih predloženih algoritama potrebno je testirati ih nad istim skupom test podataka. Sem standardizacije, problem je bio i dostupnost test setova. Kao rešenje ovih problema, 1990. godine je predstavljena *OR-Library* [2] platforma za distribuciju koja sadrži skupove podataka za testiranje rešenja različitih problema u oblasti operacionih istraživanja. Istraživači su mogli da zatraže i na svoj *email* nalog dobiju bilo koji test set iz biblioteke. Između ostalih, *OR-Library* sadrži setove podataka za testiranje rešenja *p*-medijan problema, "*warehouse location*" problema, problema trgovačkog putnika i mnoge druge. *Data* setovi su dostupni na *http://people.brunel.ac.uk/~mastjjb/jeb/info.html*. Vremenom su setovi podataka iz *OR-Library* platforme iskorišćeni za testiranje rešenja različitih problema kao što su: *p*-centar [28], *p*-medijan [15], *p*-sledeći centar [1, 21, 31] itd.

U disertaciji je predložen algoritam koji može prepoznati neka od egzaktnih rešenja *p*-sledeći centar problema i izvršen je po 20 puta nad svakom test instancom *OR-Library* test seta, inicijalno namenjenog testiranju rešenja *p*-medijan problema. Izvršen je na istoj hardverskoj konfiguraciji sa istim hiper-parametrima (k_{max} = p i t_{max} = n sekundi) kao i izvorni sofisticirani VNS. Dobijeni rezultati su predstavljeni u sledećim tabelama koje takođe sadrže iste kolone kao i tabele iz prethodnog poglavlja. Dodatno, poslednja kolona Tabela 2.8 i 2.9, označena kao "*Exact Value*", prikazuje da li je pronađeno tačno rešenje, odnosno daje vrednost tog egzaktnog rešenja.

Tabela 2.8. Rezultati za *OR-Library* test instance

	P	*N*	*Best Known Value*	*Best Found Value*	*Time*	*#Best Known*	*Gap (best found-best known) /best known *100*	*Exact Value*
pmed1	5	100	166	166	0.10	20	0.00	
pmed2	10	100	135	135	1.34	20	0.00	
pmed3	10	100	151	151	3.00	20	0.00	
pmed4	20	100	118	118	5.85	20	0.00	
pmed5	33	100	85	85	0.76	16	0.00	✓
pmed6	5	200	107	107	6.60	20	0.00	
pmed7	10	200	84	84	35.75	15	0.00	
pmed8	20	200	81	84	7.73	0	3.70	
pmed9	40	200	71	71	12.46	16	0.00	✓
pmed10	67	200	70	70	0.79	20	0.00	✓

pmed11	5	300	70	70	2.26	19	0.00	
pmed12	10	300	72	72	2.49	14	0.00	✓
pmed13	30	300	47	47	59.69	1	0.00	
pmed14	60	300	60	60	13.36	20	0.00	✓
pmed15	100	300	44	44	68.38	14	0.00	✓
pmed16	5	400	54	54	86.95	19	0.00	
pmed17	10	400	46	47	23.68	0	2.17	
pmed18	40	400	50	50	26.76	17	0.00	✓
pmed19	80	400	32	37	250.07	0	15.63	
pmed20	133	400	40	40	222.64	13	0.00	✓
pmed21	5	500	48	48	51.18	15	0.00	
pmed22	10	500	49	49	22.45	3	0.00	
pmed23	50	500	32	37	209.20	0	15.63	
pmed24	100	500	33	33	317.23	2	0.00	✓
pmed25	167	500	44	44	29.03	20	0.00	✓
pmed26	5	600	47	47	55.67	16	0.00	
pmed27	10	600	38	38	210.91	2	0.00	
pmed28	60	600	57	57	2.79	20	0.00	✓
pmed29	120	600	36	36	158.65	20	0.00	✓
pmed30	200	600	40	40	32.45	20	0.00	✓
pmed31	5	700	35	35	30.10	16	0.00	
pmed32	10	700	72	72	4.17	20	0.00	✓
pmed33	70	700	22	28	406.79	0	27.27	
pmed34	140	700	41	41	12.86	20	0.00	✓
pmed35	5	800	36	36	24.75	5	0.00	
pmed36	10	800	42	42	12.77	20	0.00	✓
pmed37	80	800	33	33	176.91	20	0.00	✓
pmed38	5	900	40	40	13.81	18	0.00	✓
pmed39	10	900	74	74	9.06	20	0.00	✓
pmed40	90	900	23	25	517.58	0	8.70	
AVG					**78.23s**	**13.53**	**1.83%**	***Total: 19***

Tabela 2.8 pokazuje da predloženi algoritam nije tako efikasan kao izvorni algoritam iz Poglavlja 2.2. Algoritam pronalazi najbolje poznato rešenje prosečno u 13.53 od ukupno 20 izvršenja. Izvorni algoritam je prosečno uspešniji za 1.83%. Sa druge strane, algoritam je uspeo da identifikuje optimalna rešenja za 19 od 40

instanci, što je indikovano u poslednjoj koloni tabele. U svakom slučaju, ostalo je 6 instanci za koje najbolje poznato rešenje nije pronađeno. Stoga je algoritam izvršen još jednom, ali ovoga puta sa najboljim poznatim rešenjima kao inicijalnim rešenjima. Dobijeni rezultati su prezentovani u sledećoj tabeli. Tabela 2.9 je proširena i kolonom koja sadrži broj grana (*m*) grafa test instance.

Tabela 2.9. Egzaktna rešenja *OR-Library* test instanci

	P	*N*	*M*	*Time*	*Exact Value*
pmed5	33	100	200	0.01	85
pmed9	40	200	800	0.04	71
pmed10	67	200	800	0.04	70
pmed12	10	300	1800	0.23	72
pmed14	60	300	1800	0.25	60
pmed15	100	300	1800	0.14	44
pmed18	40	400	3200	0.08	50
pmed19	80	400	3200	0.29	32
pmed20	133	400	3200	0.52	40
pmed24	100	500	5000	1.02	33
pmed25	167	500	5000	0.65	44
pmed28	60	600	7200	1.95	57
pmed29	120	600	7200	1.98	36
pmed30	200	600	7200	1.93	40
pmed32	10	700	9800	3.18	72
pmed33	70	700	9800	3.25	22
pmed34	140	700	9800	3.23	41
pmed36	10	800	12800	4.80	42
pmed37	80	800	12800	1.10	33
pmed38	5	900	16200	7.20	40
pmed39	10	900	16200	7.02	74
pmed40	90	900	16200	0.99	23
AVG	**73.86**	**536.36**	**6909.01**	**1.81s**	***Total: 22***

Tabela 2.9 sadrži rezultate izvršenja algoritma samo za test instance koje su tačno rešene. Algoritam je izvršen po jednom nad svakom test instancom *OR-Library* seta sa istim vrednostima parametara ($k_{max} = p$ i $t_{max} = n$ sekundi) i najboljim poznatim rešenjima kao inicijalnim. Tabela pokazuje da je algoritam uspeo da identifikuje globalno optimalna rešenja za 22 od ukupno 40 instanci. Važno je napomenuti da su uglavnom veće instance egzaktno rešene. Prosečne *p*, *n* i *m* vrednosti tačno rešenih instanci su 73.86, 536.36 i 6909.01, respektivno.

2.7. Zaključak

Problem *p*-sledećeg centra je predstavljen kao model rukovanja humanitarnom logistikom, a danas, tokom COVID-19 pandemije, interesovanje za ovakav problem je i značajno poraslo. Cilj je identifikovati *p* od potencijalnih *n* centara, recimo ambulanti, sposobnih da opslužuju sve korisnike, tj. pacijente, na takav način da je maksimalno rastojanje od lokacije korisnika do *backup* centra (centar dodeljen korisniku ukoliko je *reference* centar onesposobljen) putem koji prolazi kroz *reference* centar (korisniku najbliži centar) minimizovano među svim korisnicima.

Dizajn predloženog VNS algoritma za rešavanje problema *p*-sledećeg centra prati filozofski stav "Manje je više", koristeći što manji broj komponenti pretrage, ali u najefikasnijem maniru. Preciznije, u disertaciji je teorijski i empirijski pokazano da svojstva i strukture podataka ranije primenjene na *p*-centar problem mogu biti proširene i na ovaj novi problem. Drugim rečima, šira teorijska osnova omogućava dizajn maksimalno pojednostavljenog algoritma koji je sposoban da pronađe bolja rešenja od trenutno *state-of-the-art* algoritama. Shodno tome, pokazano je da se procedura promene susedstva može značajno unaprediti praćenjem sofisticiranih pravila filtriranja, a takođe i da tzv. *Whitaker* struktura podataka, razvijena za rešavanje *p*-medijana, funkcioniše i u slučaju problema *p*-sledećeg centra.

Dobijeni rezultati testiranja sugerišu da predloženi pristup rešavanju problema značajno prevazilazi prethodne *state-of-the-art* rezultate iz literature. U poređenju sa najboljim rešenjima dobijenim hibridnim GRASP-VNS algoritmom iz [21], pokazano je da algoritam reprodukuje ili unapređuje vrednosti svih najboljih poznatih rešenja. Sve manje test instance (do 50 čvorova) su rešene uz vreme potrebno da se po prvi put dođe do najboljeg rešenja od samo 0.10 sekundi u proseku. Štaviše, za manje test instance, najbolje rešenje je pronađeno u svakom od 20 izvršenja algoritma. Sa druge strane, za veće test instance, VNS je uspešno identifikovao najbolja poznata rešenja u 94.89% izvršenja za prosečno 4.93 sekunde do pronalaska najboljeg rešenja. Predloženi VNS je uspostavio 34 nova najbolja poznata rešenja od ukupno 92 test primera većih problema, unapređujući vrednosti *state-of-the-art* rešenja za 27%. U većini slučajeva, čak i vrednosti prosečnih i najgorih rešenja u disertaciji predloženog algoritma su bolje od u literaturi najboljih poznatih vrednosti. Da bi se još preciznije procenile performanse algoritma, generisane su i velike instance problema sa 1000, 1500, 2000 i 2500 čvorova i instance definisane nad grafovima do 1000 čvorova sa različitim gustinama grana (50% - 80%). Pokazano je da gustina grafa ne utiče na preciznost i efikasnost algoritma. Osim za dve instance problema, najbolje rešenje je pronađeno u svakom od 20 izvršenja algoritma, ili prosečno u 97.60% slučajeva. Srednje vreme pronalaska najboljeg rešenja je bilo 29.27 sekundi. Za velike instance (do $n = 2500$), algoritam nije bio tako uspešan, ali i dalje stabilan i visoko efikasan. Pronašao je najbolja rešenja u 75.23% izvršenja uz prosečno 422.78 sekundi do pronalaska najboljeg rešenja. Sa druge strane, apsolutno odstupanje vrednosti najgorih od najboljih rešenja nije bilo veće od 1.

U Poglavlju 2.4 je predstavljena i modifikacija prethodno objašnjenog VNS algoritma koja identifikuje neka od egzaktnih rešenja instanci problema *p*-sledećeg centra iz *OR-Library* [2] test skupa podataka inicijalno namenjenog testiranju rešenja *p*-medijan problema. *OR-Library* platforma za distribuciju test setova se pokazala kao izuzetno dobra i opšte prihvaćena među istraživačima za testiranje i upoređivanje efikasnosti algoritama u raznim oblastima operacionih istraživanja. Nakon što je *p*-sledeći centar predstavljen 2015. godine, kao novi NP-težak problem, svi u literaturi do sada predloženi metodi i algoritmi koji rešavaju ovaj problem su testirani nad varijantama "*p*-medijan" *OR-Library* test seta podataka. Stoga je izuzetno značajno ponuditi tačna rešenja primera iz ovog skupa podataka, a posebno većih instanci problema koje ne mogu biti rešene egzaktnim matematičkim modelima. U disertaciji je za rešavanje problema iskorišćen heuristički algoritam zasnovan na metodu promenljivih okolina, tj. modifikovan je sofisticirani VNS za rešavanje *p*-sledeći centar problema tako da bude sposoban da prepozna neka od globalno optimalnih rešenja. Izvršen nad *OR-Library* test setom, algoritam se pokazao manje efikasnim u odnosu na izvorni algoritam, ali je uspeo da identifikuje optimalna rešenja za 22 od 40 instanci. Dakle, pokazao je 55% uspešnosti u identifikovanju tačnih rešenja *OR-Library* test seta i to prvenstveno primera većih instanci problema.

3. Problem *p*-drugog centra

Problem *p*-centra je poznat i veoma proučavan problem koji podrazumeva identifikaciju *p* od potencijalnih *n* lokacija centara na takav način da se minimizuje najveće rastojanje između korisnika i najbližeg centra. Vremenom je kao potencijalni problem detektovan otkaz najbližeg centra i u tom slučaju je potrebno pronaći sledeći najbliži centar. Jedna od definicija ovog problema je *p*-drugi centar, koji predstavlja uopštenje problema *p*-centra tako što identifikuje *p* od potencijalnih *n* lokacija centara ($p \leq n$) u cilju minimizacije maksimalne sume rastojanja između korisnika i njemu najbližeg i drugog-najbližeg centra. U ovom poglavlju, kao rešenje problema *p*-drugog centra predložen je heuristički algoritam zasnovan na metodu promenljivih okolina u koji je uključen efikasan metod lokalne pretrage koji ubrzava konvergenciju ka lokalnom optimumu. Kao modifikacija, predložen je i algoritam sposoban da u pojedinim primerima problema utvrdi da li je pronađeno rešenje globalno optimalno. Algoritam je testiran nad u literaturi već poznatim skupom test instanci i rezultati pokazuju da u vremenski prihvatljivom okviru vraća optimalno ili približno optimalno rešenje problema. Štaviše, za dalju procenu performansi algoritma, generisane su test instance sa 1000, 1500, 2000 i 2500 čvorova i instance definisane nad grafovima do 1000 čvorova sa različitim vrednostima gustina. Dobijeni rezultati pokazuju visoku efikasnost i stabilnost testiranog algoritma.

3.1. Uvod

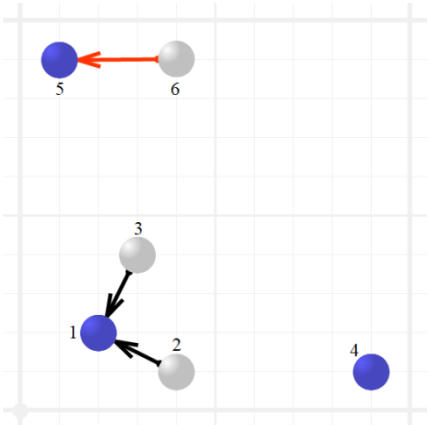
Problem *p*-centra (pCP) predstavlja identifikaciju lokacija *p* od potencijalnih *n* centara na takav način da se minimizuje najveće rastojanje između korisnika i njemu dodeljenog, tj. najbližeg centra. Problem *p*-centra je NP-težak [19], ali je već dugo poznat i proučavan tako da postoji mnogo članaka i algoritama koji uspešno rešavaju ovaj problem. Ipak, *p*-centar ne nudi rešenje problema u slučaju da dodeljeni centar nije u mogućnosti da opsluži korisnika. Problem nastaje kada u uslovima humanitarnih katastrofa centar postane preopterećen ili jednostavno dođe do kvara na centru. Intuitivno rešenje je svakom korisniku dodeliti rezervni centar u slučaju otkaza primarnog. Vođeni ovom idejom Albareda-Sambola i sar. [1] 2015. godine su, kao generalizaciju pCP-a, definisali *p*-sledeći centar problem (pNCP). Kao rešenje pNCP-a, potrebno je izabrati *p* od potencijalnih *n* centara na takav način da se minimizuje ne samo maksimalno rastojanje između korisnika i njemu najbližeg centra već i rastojanje između tog centra i nemu susednog centra.

Problem *p*-sledećeg centra podrazumeva obilazak oba, i primarnog i rezervnog centra. Očekivati je ipak da se unapred zna da li je potrebno obratiti se rezervnom centru i stoga je u literaturi kao odgovor na potencijalni otkaz centra definisan i problem *p*-drugog centra [21]. Problem *p*-drugog centra (*eng. p-second center problem*, pSCP) predstavlja generalizaciju *p*-centar problema, u smislu identifikacije lokacija *p* od mogućih *n* centara sa ciljem minimizacije maksimalne sume udaljenosti korisnika do njemu najbližeg (tzv. *reference*) i drugog-najbližeg (tzv. *backup*) centra. Formalno, neka je $G = (V, E)$ neusmeren težinski graf gde su težine grana određene rastojanjem između svojih krajeva, V skup svih čvorova, a E skup grana grafa. Centri, kao i ostali korisnici predstavljaju čvorove grafa, a $d(i,j)$ je najkraće rastojanje između čvorova i i j, izračunato kao rezultat algoritma određivanja najkraćih rastojanja u grafu G. Rešenje problema *p*-drugog centra je skup čvorova $P \subset V$, kardinalnosti p, koji sadrži centre tako da je, uzevši sve korisnike u obzir, maksimalna suma udaljenosti korisnika do najbližeg i drugog-najbližeg centra minimizovana:

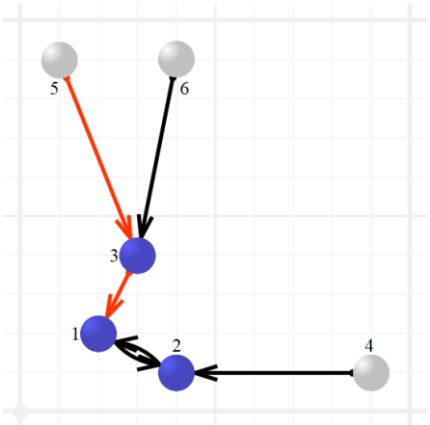
$$pSCP(V) = \min_{\substack{P \subset V \\ |P|=p}} \max_{i \in V} \left\{ \min_{j \in P} d(i,j) + \min_{k \in P \setminus \left\{ \arg\min_{j \in P} d(i,j) \right\}} d(i,k) \right\}. \tag{3.1}$$

Problemi *p*-centra, *p*-sledećeg centra i *p*-drugog centra su veoma slični. Da bi se istakla razlika između ovih problema, sledi ilustracija rešenja jednostavnog primera za $n = 6$ korisnika i $p = 3$ centra. Optimalno rešenje za pCP je 3 jedinice i centri su locirani na pozicijama 1, 4 i 5. Vrednost funkcije cilja, 3 jedinice, izračunata je kao maksimalno rastojanja svih korisnika do svojih *reference* centara, što je u ovom slučaju rastojanje između čvorova 6 i 5. Vrednost optimalnog rešenja za pNCP je 7.62 jedinice i centri su postavljeni na pozicijama 1, 2 i

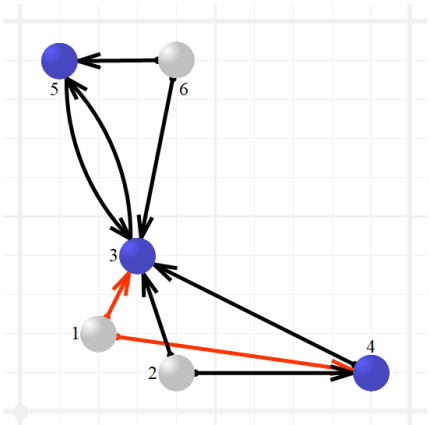
3. Vrednost funkcije od 7.62 jedinice je izračunata kao maksimalna suma rastojanja između korisnika i njegovog *reference* centra i rastojanja između *reference* i *backup* centara. U konkretnom slučaju, to je suma dužina od čvora 5 do čvora 3 plus dužina od čvora 3 do čvora 1. Sa druge strane, vrednost rešenja *{3, 4, 5}*, kao optimalnog rešenja za pSCP je 9.31 jedinica. Vrednost funkcije cilja je izračunata kao maksimalno rastojanje između korisnika i njegovog *reference* centra plus rastojanje između korisnika i *backup* centra, ili kao što je na Slici 3.3 označeno svetlijim strelicama, suma dužina između čvorova 1 i 3 i između 1 i 4.



Sl. 3.1. Optimalno rešenje za *p*-centar problem (*n = 6* i *p = 3*)



Sl. 3.2. Optimalno rešenje za *p*-sledeći centar problem (*n = 6* i *p = 3*)



Sl. 3.3. Optimalno rešenje za *p*-drugi centar problem (*n = 6* i *p = 3*)

Kada su svi centri funkcionalni, korisnici dolaze do najbližeg centra (pCP). Na Slici 3.1, korisnici iz centara 1, 4 i 5 ostaju u svojim centrima s obzirom da su to *reference* centri, a korisnici iz 2 i 3 odlaze u centar 1, odnosno korisnik 6 odlazi u centar 5 s obzirom da mu je to najbliži centar. Kako god, može se ispostaviti da kada korisnik stigne do centra da tada taj centar ne bude funkcionalan. U tom slučaju, korisnici dolaze do

najbližih *backup* centara (pNCP), kao što je predstavljeno na Slici 3.2. U primeru, korisnici iz centara 1, 2 i 3 ostaju u ovim centrima s obzirom da su to *reference* centri, ali u slučaju da neki od njih nisu više funkcionalni, korisnici nastavljaju do sledećih najbližih centara, tj. *backup* centara. *Backup* centar centara 2 i 3 je centar 1, a centra 1 *backup* je centar 2. Sa druge strane, za očekivati je da korisnici unapred saznaju za otkaz *reference* centra. Tada direktno odlaze do drugog-najbližeg centra (pSCP). Ipak, pSCP je definisan kao minimalna suma rastojanja do najbližih i drugih-najbližih centara, što u primeru na Slici 3.3 odgovara sumi rastojanja korisnika 1 do centara 3 i 4, korisnika 2 do centara 3 i 4, korisnika 3 do centara 3 i 5, korisnika 4 do centara 4 i 3, korisnika 5 do centara 5 i 3 i korisnika 6 do centara 5 i 3.

Opisani problemi su slični, ali očigledno je da se optimalna rešenja za pCP, pNCP i pSCP razlikuju. Kod pNCP dolazi do grupisanja svih centara bliže jedan drugome, dok je u slučaju pSCP primetno grupisanje parova centra oko grupa susednih korisnika.

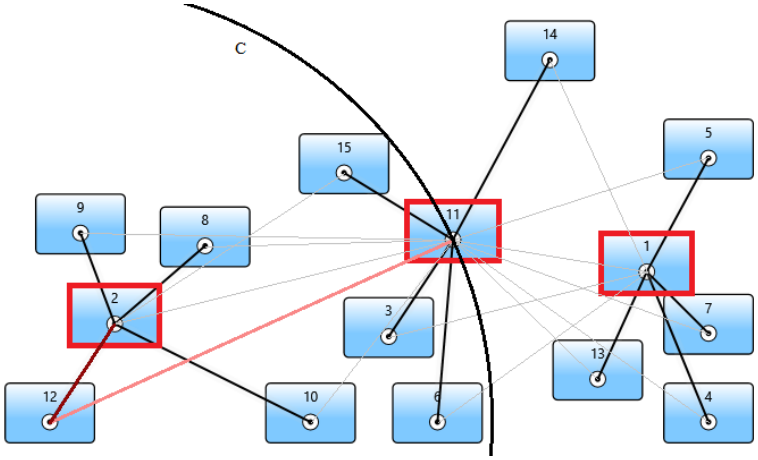
Problem *p*-drugog centra, kao proširenje pCP, predstavlja NP-težak problem. Kao rešenje problema, u disertaciji se predlaže heuristički algoritam zasnovan na metodu promenljivih okolina koji uključuje efikasan metod lokalne pretrage koji ubrzava konvergenciju ka lokalnom optimumu. U tom cilju, u narednim poglavljima sledi opis algoritma i rezultati testiranja kao i poređenja sa u literaturi već poznatim algoritmima za rešavanje sličnih problema.

3.2. Algoritam za rešavanje problema *p*-drugog centra

Predloženi algoritam za rešavanje problema *p*-drugog centra je izgrađen na metaheurističkom metodu promenljivih okolina (*eng. Variable Neighborhood Search*, VNS). VNS su predstavili Mladenović i Hansen u radu [27] iz 1997. godine kao generički metod za izgradnju algoritama pretrage.

Problem *p*-drugog centra je generalizacija problema *p*-centra. Mladenović i sar. [28] su 2004. godine prezentovali efikasan VNS algoritam za *p*-centar problem, tj. problem optimalne identifikacije lokacija *p* od *n* centara sa ciljem minimizacije maksimalne udaljenosti korisnika do najbližeg centra. Kao rešenje problema *p*-drugog centra, u disertaciji se preuzima originalni algoritam iz rada [28] uz jednostavnu modifikaciju, tako da se minimizuje ne maksimalno rastojanje od korisnika do najbližeg centra već suma rastojanja do najbližeg i drugog-najbližeg centra.

U osnovi, VNS algoritam u svakoj svojoj iteraciji pokušava da unapredi tekuće rešenje, označeno kao *P*. Sastoji se od naizmeničnih procedura slučajnog odabira rešenja iz $N_k(P)$ okoline i lokalne pretrage izabranog rešenja u cilju pronalaska lokalnog optimuma. Kao pojašnjenje predložene implementacije, sledi primer problema *p*-drugog centra sa $n = 15$ korisnika i $p = 3$ centra (Slika 3.4). Tekuće rešenje problema predstavlja skup centara $P = \{1, 2, 11\}$.



Sl. 3.4. Primer problema *p*-drugog centra sa $n = 15$ i $p = 3$; tekuće rešenja je $P = \{1, 2, 11\}$, a kritični korisnik $u_c = 12$

Na Slici 3.4, korisnici su tamnijim linijama (granama) povezani sa svojim najbližim a svetlijim sa drugim-najbližim centrima. Udaljenost do centara odgovara dužini grana. Korisnik sa najvećom sumom rastojanja do odgovarajućih centara je korisnik 12. Označen je kao kritični korisnik, u_c = 12. Upravo suma udaljenosti najbližeg i drugog-najbližeg centra (centara 2 i 11) od kritičnog korisnika određuje vrednost funkcije cilja. Da bi se dalje unapredila vrednost funkcije cilja, neophodno je smanjiti udaljenost kritičnog korisnika do najbližeg i/ili drugog-najbližeg centra. U tom cilju, potrebno je pronaći novi centar koji je bliži kritičnom korisniku od drugog-najbližeg centra.

Svojstvo 3.1. Neka je u_c kritični korisnik, $c1(u_c)$ njegov najbliži, a $c2(u_c)$ drugi-najbliži centar u tekućem rešenju P, ($c1(u_c) \in P$, $c2(u_c) \in P$). Tada, ukoliko postoji bolje rešenje u okolini $N_1(P)$ od tekućeg rešenja P, novi centar c_{in} $\left(c_{in} \neq c1(u_c)\right)$ mora biti bliži kritičnom korisniku u_c od njegovog prethodno drugog-najbližeg centra $c_2(u_c)$.

Dokaz. Neka je $f(P)$ vrednost funkcije cilja tekućeg rešenja P, a $f(u_i)$ vrednost funkcije korisnika u_i. Uz to, $d(u_i, u_j)$ predstavlja najkraće rastojanje između čvorova u_i i u_j. Tada važi:

$$f(P) = max_{i=1,\dots,n} f(u_i) = f(u_c) = d\big(u_c, c1(u_c)\big) + d\big(u_c, c2(u_c)\big). \tag{3.2}$$

Vrednost funkcije cilja je određena vrednošću funkcije kritičnog korisnika. Prema tome, da bi se smanjila vrednost funkcije cilja neophodno je u tekuće rešenje uključiti novi centar c_{in} tako da je:

$$d\big(u_c, c1(u_c)\big) + d(u_c, c_{in}) < d\big(u_c, c1(u_c)\big) + d\big(u_c, c2(u_c)\big), \tag{3.3}$$

tj.

$$d(u_c, c_{in}) < d\big(u_c, c2(u_c)\big). \blacksquare \tag{3.4}$$

Posledica. Ukoliko kritični korisnik nije jedinstven, Svojstvo 3.1 važi za svakog od kritičnih korisnika, tj. $(\forall u \in U_c)\left(d(u, c_{in}) < d\big(u, c2(u_c)\big)\right)$, gde je U_c skup svih kritičnih korisnika. U suprotnom, u okolini $N_1(P)$ ne postoji ni jedno rešenje bolje od tekućeg rešenja P.

Na osnovu prethodnog svojstva, može se videti kako se kardinalnost skupa potencijalno novih centara smanjuje i tako ubrzava konvergencija ka lokalnom minimumu. Filtriranje centara omogućava redukciju veličine kompletnog susedstva rešenja P sa $p * (n - p)$ na $p * |N(u_c)|$, gde je:

$$N(u_c) = \{u | d\big(u_c, c1(u_c)\big) + d(u_c, u) < f(P)\}. \tag{3.5}$$

Štaviše, kardinalnost se dalje smanjuje sa narednim iteracijama i na taj način u srednjem slučaju povećava brzina konvergencije ka lokalnom minimumu. Međutim, tako se smanjuje samo konstantan faktor u kompleksnosti heuristike, ali ne i njena vremenska složenost u najgorem slučaju.

Na osnovu Svojstva 3.1, pretraga se redukuje samo na čvorove u koji zadovoljavaju uslov $d(u_c, u) < d(u_c, c2(u_c))$. U konkretnom slučaju sa Slike 3.1, skup potencijalno novih centara čine čvorovi unutar kruga C (čvorovi koji su bliži kritičnom korisniku od drugog-najbližeg centra). Neka je kao novi centar c_{in} izabran čvor 3. Novi centar svakako smanjuje udaljenost kritičkog korisnika 12 od dodeljenih centara, ali da li će i vrednost funkcije cilja biti redukovana zavisi od toga da li će korisnici koji izgube jedan od svojih centara zadržati sumu udaljenosti od novo-dodeljenih centara na nivou manjem od prethodne vrednosti funkcije cilja. Upravo *Move* procedura (Algoritam 3.1) identifikuje centar tekuće pretraživanog rešenja koji bi trebalo zameniti novim centrom u cilju unapređenja vrednosti funkcije cilja, a ujedno i izračunava vrednost funkcije cilja novog rešenja. Zapravo, umesto provere svih p mogućih isključenja centara, tokom obilaska svakog korisnika samo se evidentira potencijalno nova vrednost funkcije cilja u slučaju da se jedan od dva dodeljena centra zatvori (*New center* korak). Vrednosti se čuvaju u odgovarajućim elementima z niza. Nakon toga, kao optimalan centar za brisanje se bira onaj koji odgovara minimalnoj z vrednosti (*Best deletion* korak). Na opisani način vremenska kompleksnost se redukuje sa $O(pn)$ na $O(n) + O(p) \approx O(n)$. Konačno, u *Function calculation* koraku izračunava se nova vrednost funkcije cilja. Vremenska složenost *Move* procedure je $O(n)$.

Algoritam podrazumeva da tekuće rešenje P predstavlja prvih p elemenata niza x_{cur}, a ostali korisnici su n - p elemenata sa kraja istog niza. Indeks novog centra je označen sa c_{in}. Dodatno, prezentovani algoritam koristi sledeće strukture:

- *dist(u, v)* – najkraće rastojanje između čvorova *u* i *v*,
- *c1(u)* – najbliži centar korisnika *u*,
- *c2(u)* – drugi-najbliži centar korisnika *u*,
- *c3(u)* – treći-najbliži centar korisnika *u*,
- *z(v)* – maksimalna vrednost funkcije cilja, nakon što je centar *v* obrisan, uzevši u obzir sve korisnike kojima je *v* bio najbliži ili drugi-najbliži centar.

Algoritam 3.1: Zamena čvorova u N_1 okolini (*eng. 1-interchange move*) u kontekstu *p*-drugi centar problema

Move(x_{cur}, c_{in}, c1, c2, c3)
Initialization:
Set $z(x_{cur}(i)) \leftarrow 0$ for all $i = 1, ..., p$

New center:
$in \leftarrow x_{cur}(c_{in})$
For Each *user = $x_{cur}(1), \ldots, x_{cur}(n)$*
    If *dist(user, in) < dist(user, c2(user))*
        *****in as a new closest or second-closest center*****
        z(c1(user)) ← max(dist(user, in) + dist(user, c2(user)), z(c1(user)))
    Else
        *****user keeps the same centers*****
        z(c1(user)) ← max(min(dist(user, in), dist(user, c3(user))) + dist(user, c2(user)), z(c1(user)))
        z(c2(user)) ← max(min(dist(user, in), dist(user, c3(user))) + dist(user, c1(user)), z(c2(user)))
    End If
End For Each

Best deletion:
min ← ∞
For Each *i = {1, ..., p}*
    If *min > $z(x_{cur}(i))$*
        min ← $z(x_{cur}(i))$
        $c_{out} \leftarrow i$
    End If
End For Each

Function calculation:
$f_{cur} \leftarrow 0$
out ← $x_{cur}(c_{out})$
For Each *user = $x_{cur}(1), \ldots, x_{cur}(n)$*
    If *c1(user) = out*
        c1 ← c2(user)
        c2 ← c3(user)
    Else
        c1 ← c1(user)
        c2 ← c2(user) ***if*** *c2(user) ≠ out* ***else*** *c3(user)*
    End If
    f ← dist(user, c1) + dist(user, in) ***if*** *dist(user, in) < dist(user, c2)* ***else*** *dist(user, c1) + dist(user, c2)*
    $f_{cur} \leftarrow max(f, f_{cur})$
End For Each

Return *f_{cur}, c_{out}*

Predložena implementacija se oslanja na algoritme i strukture iz rada [28], sa tim da je rešenje prošireno nizom c_3 koji sadrži treće-najbliže centre svih korisnika. Dodatna struktura i proširenja algoritama su uslovljena samom prirodom *p*-drugi centar problema, tj. zahtevom da je unapred poznat novi drugi-najbliži centar svakog od korisnika u slučaju da je jedan od najbliža dva eliminisan iz tekućeg rešenja. Proširenja ne utiču na ispravnost i efikasnost algoritma, tako da diskusija i teorijska osnova iz rada [28] ostaje potpuno primenljiva. U radu [28], autori analiziraju bipartitivni graf. To je urađeno da bi se napravila jasna razlika između centara i korisnika. Algoritam koji se predlaže u ovoj disertaciji nema takvu pretpostavku. Algoritam je primenljiv na opšti težinski graf bez ograničenja u pogledu broja grana i povezanosti. Ipak, ova razlika u postavci problema nikako ne utiče na ispravnost algoritma. Problem definisan preko bipartitivnog grafa može biti predstavljen kao ekvivalent opštem grafu gde je težina grana između korisnika i centra na istoj lokaciji 0. Prema tome, predloženi pristup i algoritam predstavljaju samo generalizaciju algoritma i diskusije iz [28] na opšte grafove i proširenje rešenja pomoćnom strukturom centara kako bi bilo moguće opslužiti korisnike i u slučaju ispada primarnih centara. U nastavku sledi pseudokod preostalih procedura koje implementiraju algoritam.

Algoritam 3.2: Ažuriranje najbližih, drugih-najbližih i trećih-najbližih centara

```
Update(x_cur, c_in, c_out, c1, c2, c3)
in ← x_cur(c_in)
out ← x_cur(c_out)
For Each user = x_cur(1), …, x_cur(n)
        **for users whose center is deleted, find new one**
        If c1(user) = out
                If dist(user, in) ≤ dist(user, c2(user))
                        c1(user) ← in
                Else
                        c1(user) ← c2(user)
                        If dist(user, in) ≤ dist(user, c3(user))
                                c2(user) ← in
                        Else
                                c2(user) ← c3(user)
                                **find third closest center for the user**
                                c3(user) ← select center
                                        from {x_cur(1), …, x_cur(p)} ∪{in} \ {c1(user), c2(user), out}
                                        where d(user, center) is minimum
                        End If
                End If
        Else
                If c2(user) = out
                        If dist(user, in) ≤ dist(user, c1(user))
                                c2(user) ← c1(user)
                                c1(user) ← in
                        Else
                                If dist(user, in) ≤ dist(user, c3(user))
                                        c2(user) ← in
                                Else
                                        c2(user) ← c3(user)
                                        **find third closest center for the user**
                                        c3(user) ← select center
                                                from {x_cur(1), …, x_cur(p)} ∪{in} \ {c1(user), c2(user), out}
                                                where d(user, center) is minimum
                                End If
                        End If
                Else
```

```
                    If dist(user, in) ≤ dist(user, c1(user))
                        c3(user) ← c2(user)
                        c2(user) ← c1(user)
                        c1(user) ← in
                    Else
                        If dist(user, in) ≤ dist(user, c2(user))
                            c3(user) ← c2(user)
                            c2(user) ← in
                        Else
                            If dist(user, in) ≤ dist(user, c3(user))
                                c3(user) ← in
                            Else If c3(user) = out
                                **find third closest center for the user**
                                c3(user) ← select center
                                from {x_cur(1), ..., x_cur(p)} ∪ {in} \ {c1(user), c2(user), out}
                                where d(user, center) is minimum
                            End If
                        End If
                    End If
                End If
            End If
        End For Each
        Return c1, c2, c3
```

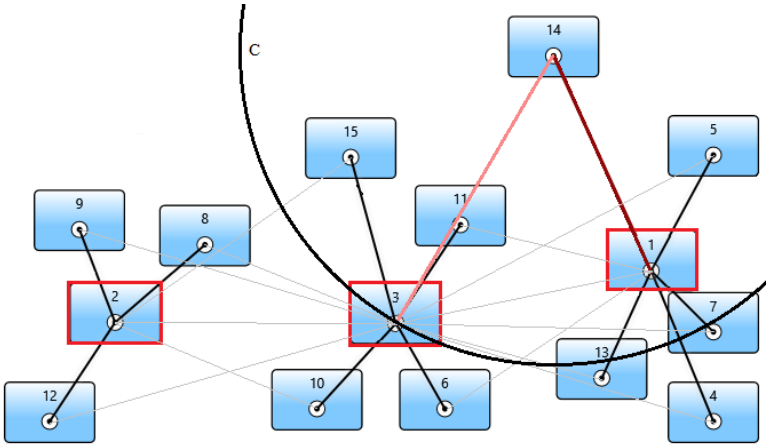
Algoritam 3.3: Procedura zamene čvorova tokom faze lokalne pretrage za *p*-drugi centar problem

```
LocalSearchVertexSubstitution(x_cur, c1, c2, c3, u_c, f_cur)
Main loop:
While True
    f' ← ∞
    For Each in = p +1, ..., n
        Investigate N_1(x_cur) neighborhood with center x_cur(in) as a new center and find the best deletion:
        If d(u_c, x_cur(in)) < d(u_c, c2(u_c))
            f, out ← Move(x_cur, in, c1, c2, c3)
            If f < f'
                f' ← f
                c_in ← in
                c_out ← out
            End If
        End If
    End For Each
    If f_cur ≤ f'
        There is not found improvement in the neighborhood:
        Break Main loop
    End If
    Update(x_cur, c_in, c_out, c1, c2 ,c3)
    f_cur ← f'
    x_cur(c_in) ↔ x_cur(c_out)
    u_c ← select user from {x_cur(1), ..., x_cur(n)} where d(user, c1(user)) + d(user, c2(user)) is maximum
End While
Return x_cur, u_c, f_cur
```

Update procedura (Algoritam 3.2) koristi strukture *c1*, *c2* i *c3* koje predstavljaju liste najbližih i drugih i trećih najbližih centara svih korisnika. To su i ulazne i izlazne promenljive, dok c_{in} i c_{out}, tj. indeksi centara koji se dodaju i brišu iz tekućeg rešenja, zajedno sa tekućim rešenjem x_{cur} predstavljaju samo ulazne promenljive. Korisnici u tekućem rešenju se ispituju jedan po jedan i proverava se da li će neki od njihovih najbližih centara biti obrisan (c_{out}) ili će možda novi centar c_{in} postati jedan od najbližih. Ukoliko se prethodno pomenuto dogodi, ažurira se odgovarajuća *c1*, *c2* i/ili *c3* lista najbližih centara. Kada se za ažuriranje trećih-najbližih centara koristi hip struktura podataka, vremenska složenost *Update* procedure je $O(n \log n)$.

LocalSearchVertexSubstitution procedura (Algoritam 3.3) se oslanja na Svojstvo 3.1 i na taj način ubrzava konvergenciju ka lokalnom optimumu. Ubrzanje redukuje konstantan faktor kompleksnosti, ali ne i složenost u najgorem slučaju. Složenost jedne iteracije *Main petlje (loop)* u najgorem slučaju ostaje $O(n^2) + O(n \log n) + O(n) \approx O(n^2)$. Bez obzira na to, prostor pretrage se sa svakom iteracijom smanjuje. Ulazne vrednosti procedure su tekuće rešenje, nizovi *c1*, *c2* i *c3*, kritični korisnik i tekuća vrednost funkcije cilja. Procedura lokalne pretrage pozivom *Move* procedure u svakoj iteraciji pronalazi optimalan par centara čija razmena rezultira najvećom redukcijom vrednosti funkcije cilja. Izvršava se dokle god je moguće pronaći takav par centara i na taj način rezultira "najboljim" rešenjem, tj. lokalnim optimumom. Uz ažuriranje pomoćnih struktura centara, procedura kao izlazne vrednosti daje novo tekuće rešenje, novog kritičnog korisnika i novu vrednost funkcije cilja.

Nakon iteracije predložene VNS implementacije (tj. prethodno opisanih algoritama) nad problemom sa Slike 3.4, dobija se novo tekuće rešenje $P = \{1, 2, 3\}$, predstavljeno Slikom 3.5. Otvoren je novi centar 3, a zatvoren centar 11. Novi kritični korisnik je $u_c = 14$, njemu najbliži centar 1, a drugi-najbliži centar 3. Vrednost funkcije cilja je smanjena, tj. $d(14, 1) + d(14, 3) < d(12, 2) + d(12, 11)$.



Sl. 3.5. Primer problema *p*-drugog centra sa $n = 15$ i $p = 3$; novo rešenja je $P = \{1, 2, 3\}$, a kritični korisnik $u_c = 14$

Pseudokod VNS algoritma za problem *p*-drugog centra je dat u Algoritmu 3.4. Inicijalna vrednost promenljive *k* je 1, što znači da se novo rešenje traži u N_1 okolini inicijalno tekućeg rešenja. U *Shaking operator* koraku bira se rešenje iz N_k okoline tekućeg rešenja. Kao i u proceduri lokalne pretrage, na osnovu Svojstva 3.1 redukuje se veličina skupa N_k, tj. bira se jedno od rešenja koje uključuje novi centar koji je bliži kritičnom centru u_c* od njemu prethodno drugog-najbližeg centra $c2(u_c^*)$. Nakon toga, izabrano rešenje x_{cur} postaje tekuće rešenje procedure lokalne pretrage (*Local search* korak). Ukoliko se lokalnom pretragom pronađe jednako ili bolje rešenje od x_{cur}, prihvata se novo rešenje kao "trenutno optimalno" x_{opt}, a pretraga se resetuje na početak, $k = 1$. U suprotnom, vrednost *k* se povećava za 1, tj. pretraga se nastavlja u skupu N_{k+1}, ili ukoliko je *k* dostiglo maksimalnu vrednost k_{max}, *k* se postavlja na 1. *Main step* se ponavlja sve dok se ne dostigne maksimalno dozvoljeno vreme izvršenja t_{max}. Algoritam vraća najbolje rešenje pronađeno tokom pretrage.

Algoritam 3.4: VNS algoritam za problem *p*-drugog centra

VariableNeighborhoodSearch(k_{max}, t_{max})

Initialization:

Randomly initialize x_{opt}; according to x_{opt} initialize arrays c1, c2 and c3, f_{opt}, u_c; copy initial solution into the current one, i.e., copy f_{opt}, x_{opt}, c1, c2, c3 and u_c* into f_{cur}, x_{cur}, $c1_{cur}$, $c2_{cur}$, $c3_{cur}$ and u_{cur}*, respectively.*

Repeat the Main step until the stopping condition is met (e.g., time ≤ t_{max})

Main step:

$k \leftarrow 1$

While $k \leq k_{max}$

Shaking operator:

*****generate a solution at random from kth neighborhood*****

For Each $j = 1, ..., k$

- *Take center to be inserted (c_{in}) at random if $d(u_{cur}^*, x_{cur}(c_{in})) < d(u_{cur}^*, c2_{cur}(u_{cur}^*))$ is true;*
- *Find center to be deleted (c_{out}) at random;*
- *Update x_{cur}, $c1_{cur}$, $c2_{cur}$ and $c3_{cur}$,i.e., execute:*
  Update(x_{cur}, c_{in}, c_{out}, $c1_{cur}$, $c2_{cur}$, $c3_{cur}$)
  $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
- *Update f_{cur} and u_{cur}* according to x_{cur}, $c1_{cur}$, $c2_{cur}$ and $c3_{cur}$*

End For Each

Local search:

If *any potentially better solution found*

$x_{cur}, u_{cur}^*, f_{cur} \leftarrow LocalSearchVertexSubstitution(x_{cur}, c1, c2, c3, u_{cur}^*, f_{cur})$

Move or not:

If $f_{cur} \leq f_{opt}$

*****Save current solution as the optimal; return to N_1*****

$x_{opt} \leftarrow x_{cur}$; $f_{opt} \leftarrow f_{cur}$; $u_c^* \leftarrow u_{cur}^*$; $c1 \leftarrow c1_{cur}$; $c2 \leftarrow c2_{cur}$; $c3 \leftarrow c3_{cur}$

$k \leftarrow 1$

Else

*****Current solution is the optimal; change the neighborhood*****

$x_{cur} \leftarrow x_{opt}$; $f_{cur} \leftarrow f_{opt}$; $u_{cur}^* \leftarrow u_c^*$; $c1_{cur} \leftarrow c1$; $c2_{cur} \leftarrow c2$; $c3_{cur} \leftarrow c3$

$k \leftarrow k + 1$

End If

End If

End While

Return x_{opt}, f_{opt}

3.3. Modifikacija algoritma za prepoznavanje egzaktnih rešenja problema *p*-drugog centra

Svojstvo 3.2. Neka je u_c kritični korisnik, $c1(u_c)$ njegov najbliži, a $c2(u_c)$ drugi-najbliži centar u tekućem rešenju P, ($c1(u_c) \in P$, $c2(u_c) \in P$). Tada, ukoliko u skupu potencijalno novih centara ne postoji centar c_{in} $\left(c_{in} \neq c1(u_c)\right)$ koji je bliži kritičnom korisniku u_c od njegovog prethodno drugog-najbližeg centra $c_2(u_c)$, tekuće rešenje *P* je globalno optimalno rešenje problema *p*-drugog centra.

Dokaz. Na osnovu Svojstva 3.1, važi da je $d(u_c, c_{in}) < d\left(u_c, c2(u_c)\right)$. Ukoliko nije moguće pronaći novi centar c_{in} koji zadovoljava prethodnu nejednakost, to znači da nije moguće unaprediti tekuće rešenje, tj. tekuće rešenje je globalno optimalno rešenje problema *p*-drugog centra. ■

Posledica. Ukoliko ne postoji nijedan novi centar čijim otvaranjem bi se unapredila vrednost funkcije kritičnog korisnika u_c, kritični korisnik je već izabran da bude centar u tekućem rešenju P.

Svojstvo 3.2 se može predstaviti kao uopštenje Svojstva 3.1, tj. ukoliko postoji bolje rešenje P' od tekućeg rešenja P, rešenje P' mora sadržati novi centar c_{in} ($c_{in} \notin P$) koji je bliži kritičnom korisniku u_c od njegovog prethodno drugog-najbližeg centra $c2(u_c)$, gde $c2(u_c) \in P$.

Algoritam 3.5: VNS algoritam za prepoznavanje egzaktnih rešenja problema *p*-drugog centra

VariableNeighborhoodSearch(k_{max}, t_{max})
Initialization:
…
Repeat the Main step until the stopping condition is met (e.g., time ≤ t_{max})
Main step:
k ← 1
While *k ≤ k_{max}*
Shaking operator:
*****generate a solution from kth neighborhood*****
count ← 0
For Each *j = 1, ..., k*
c_{in} ← ***find*** *center at random* ***from*** *{$x_{cur}(p + 1)$, …, $x_{cur}(n)$}*
where *d(u_{cur}*, center) < d(u_{cur}*, $c2_{cur}$(u_{cur}*))*
If *c_{in} is not found*
If *count = 0*
*****the optimal solution has been found*****
Return *x_{opt}, f_{opt}*
End If
Else
- *Increment count*
- *Find center to be deleted (c_{out}) at random;*
- *Update x_{cur}, $c1_{cur}$, $c2_{cur}$ and $c3_{cur}$,i.e., execute:*
  Update(x_{cur}, c_{in}, c_{out}, $c1_{cur}$, $c2_{cur}$, $c3_{cur}$)
  $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
- *Update f_{cur} and u_{cur}* according to x_{cur}, $c1_{cur}$, $c2_{cur}$ and $c3_{cur}$*

End If
End For Each
Local search:
…
End While
Return *x_{opt}, f_{opt}*

Na osnovu Svojstva 3.2, predložena ja modifikacija VNS algoritma iz prethodnog poglavlja sposobna da u nekim slučajevima prepozna egzaktna rešenja problema *p*-drugog centra. Suština je, pre faze lokalne pretrage, proveriti da li postoji barem jedan centar koji potencijalno unapređuje tekuće rešenje. Ukoliko ne postoji, izvršenje se zaustavlja i algoritam vraća tekuće rešenje kao optimalno (Algoritam 3.5).

Kao budući rad, interesantno bi bilo pokušati ubrzati konvergenciju ka optimalnom rešenju pretragom istorije prethodnih rešenja. Npr., rešenje $P1 = \{1, 2, 11\}$ sa Slike 3.4 se unapređuje otvaranjem centra 3 i nakon zatvaranja centra 11 postaje $\{1, 2, 3\}$. Skup svih potencijalno novih centara čine centri unutar kruga C, tj. $C1_{in} = \{3, 6, 8, 9, 10, 12, 15\}$. Dakle, rešenje $\{1, 2\}$ je prošireno jednim od centara iz skupa $C1_{in}$. Na Slici 3.5, tekuće rešenje je $P2 = \{1, 2, 3\}$, a skup potencijalno novih centara $C2_{in} = \{5, 11, 14, 15\}$. U slučaju zatvaranja centra 3, tekuće rešenje se eventualno može proširiti centrima iz skupa $C2_{in} \cap C1_{in} = \{15\}$, što smanjuje kardinalnost skupa pretrage sa 4 na 1. Za memorisanje prethodnih rešenja se može iskoristiti nova struktura, npr. stablo koje u svojim listovima čuva skup svih potencijalnih centara koji mogu proširiti rešenje definisano čvorovima na putanji od korena do tog lista stabla.

3.4. Rezultati testiranja algoritma za problem *p*-drugog centra

Algoritam je implementiran u programskom jeziku C++, a sva testiranja su izvršena na *Intel Core i7-8700K (3.7GHz) CPU* sa *32GB RAM*-a konfiguraciji. Za potrebe testiranja, preuzet je *OR-Library* [2] skup podataka koji sadrži 40 test instanci sa 100 do 900 čvorova i p između 5 i 200 (ne više od $n/3$, gde je n broj čvorova). Dodatno, generisana su dva skupa podataka sa većim test instancama. Prvi sadrži 44 instance sa 1000 do 2500 čvorova i p između 5 i 200, a drugi 48 instanci (500 - 1000 čvorova) definisanih nad grafovima različitih gustina (50% - 80%) i p između 5 i 200.

Predloženi algoritmi su izvršeni po 20 puta nad svakom test instancom, uvek počevši od različitog inicijalnog rešenja. Isprobane su različite kombinacije parametara $k_{max} = p/4$, $k_{max} = p/2$, $k_{max} = p$, kao i $t_{max} = n$ i $t_{max} = 2n$. Ispostavilo se da su rezultati neznatno bolji sa većim vrednostima parametra k_{max}. Sa druge strane, algoritam je pronalazio najbolje rešenje znatno pre isteka vremenskog limita izvršenja. Stoga su prezentovani rezultati samo za $k_{max} = p$ i $t_{max} = n$ sekundi. Zbirni rezultati su predstavljeni u narednim tabelama, dok su detaljni rezultati dostupni u Dodatku B.1.

3.4.1. Rezultati testiranja nad *OR-Library* instancama

Tabela 3.1 prikazuje dobijene rezultate za izvorne instance *OR-Library* test biblioteke. Prva kolona tabele sadrži naziv instance, naredne tri kolone predstavljaju vrednosti p (broj centara), n (broj korisnika) i m (broj grana grafa). U kolonama "*Best Value*", "*AVG Value*" i "*Worst Value*" prikazuje se najbolja, prosečna i najgora vrednost rešenja koje je algoritam pronašao tokom 20 izvršenja. Kolona "*Time*" (ili *time-to-target*) prikazuje prosečno vreme u sekundama koje je bilo potrebno da se prvi put pronađe najbolje rešenje. "*Time*" nije ukupno vreme izvršenja. Algoritam se izvršava sve dok ne istekne vremenski limit, tj. n sekundi. Poslednja kolona "*#Best*" daje broj koliko puta je algoritam pronašao najbolje rešenje u 20 izvršenja. Takođe, u Dodatku B.1, uključene su dodatne kolone koje sadrže procentualno odstupanje prosečnog i najgoreg rešenja u odnosu na najbolje poznato rešenje dato u koloni "*Best-Known Value*". Kako *OR-Library* instance još uvek nisu korišćene za testiranje problema *p*-drugog centra, kao najbolje poznato rešenje uzeto je najbolje rešenje koje je pronašao predloženi algoritam.

Na osnovu rezultata iz Tabele 3.1, primećuje se da je većina manjih instanci rešena vrlo brzo. Samo u 3 od 15 slučajeva za $n \leq 300$, prosečno vreme pronalaska najboljeg rešenja je veće od 4 sekunde. Sa porastom veličine test instanci, očekivano raste i vreme potrebno da se pronađe najbolje rešenje. Prosečno vreme pronalaska najboljeg rešenja nad kompletnim test skupom podataka je 31.32 sekunde. Sa druge strane, u pogledu stabilnosti algoritma i kvaliteta rešenja, svako od 20 izvršenja nije pronašlo najbolje poznato rešenje samo za 4 od 40 slučajeva. Štaviše, samo za dva primera (pmed37 i pmed40), najbolje rešenje je pronađeno manje od osamnaest puta u 20 izvršenja. Za instancu pmed40, samo je jednom pronađeno najbolje rešenje, ali je u svim ostalim izvršenjima pronađeno rešenje čija je vrednost samo za 1 veća od najboljeg. Algoritam je

pronašao najbolje poznato rešenje u proseku u 19.20 od 20 slučajeva ili u 96%. Što se tiče odstupanja prosečnih i najgorih rešenja, iz Tabele B.1.1 (Dodatak B.1) može se zaključiti da nema značajnih razlika između najgorih/prosečnih i najboljih vrednosti, samo 0.23% i 0.12% u proseku, respektivno.

Tabela 3.1. Rezultati za pSCP-VNS algoritam testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
pmed1-pmed5	5-33	100	200	193.80	193.80	193.80	3.30	20.00
pmed6-pmed10	5-67	200	800	120.00	120.02	120.20	4.33	19.60
pmed11-pmed15	5-100	300	1800	83.80	83.80	83.80	2.23	20.00
pmed16-pmed20	5-133	400	3200	65.00	65.00	65.00	34.49	20.00
pmed21-pmed25	5-167	500	5000	58.60	58.61	58.80	48.15	19.80
pmed26-pmed30	5-200	600	7200	56.00	56.00	56.00	35.03	20.00
pmed31-pmed34	5-140	700	9800	53.00	53.00	53.00	19.83	20.00
pmed35-pmed37	5-80	800	1280	51.67	51.83	52.00	149.49	16.67
pmed38-pmed40	5-90	900	16200	54.67	54.98	55.00	29.12	13.67
AVG							**31.32s**	**19.20**

U cilju verifikacije kvaliteta novog sofisticiranog metoda lokalnog pretraživanja u okviru VNS algoritma za pSCP, primenjen je i testiran i jednostavan metod lokalne pretrage sa istim parametrima i test podacima. Za razliku od predloženog rešenja, jednostavna procedura lokalnog pretraživanja ne filtrira centre koji ne zadovoljavaju Svojstvo 3.1. Procedura samo traga za centrima koji poboljšavaju tekuće rešenje, kao što je prikazano u Algoritmu 3.6.

Algoritam 3.6: Jednostavna procedura lokalne pretrage

SimpleLocalSearch(x_{cur})
Main loop:
While *True*
    $P \leftarrow \{x_{cur}(1), ..., x_{cur}(p)\}$
    (in, out) ← *select (user, center)*
        where user in $\{x_{cur}(p+1), ..., x_{cur}(n)\}$ *and center in* $\{x_{cur}(1), ..., x_{cur}(p)\}$
        and $P \cup \{user\} \setminus \{center\}$ *is better than P*
    If *(in, out) found*
        Exchange(in, out)
    Else
        Break Main loop
    End If
End While
Return x_{cur}

Algoritam 3.6 je takođe izvršen 20 puta i rezultati su predstavljeni u Tabeli 3.2. Za VNS koji koristi jednostavan algoritam lokalne pretrage prikazano je najbolje rešenje dobijeno u 20 izvršenja (kolona "*Best-Found Value*") i prosečno vreme pronalaska najboljeg rešenja (kolona "*Time*"). Kolona "*#Best-Known*" sadrži broj izvršenja algoritma sa jednostavnom procedurom lokalne pretrage koja su rezultirala pronalaskom najboljeg poznatog rešenja, tj. najboljeg rešenja koje je pronašao algoritam sa sofisticiranom lokalnom pretragom. Konačno, poslednja kolona daje procentualno odstupanje najboljeg pronađenog rešenja algoritma sa jednostavnom procedurom lokalne pretrage od najboljeg poznatog rešenja. Procenat odstupanja ("*percentage gap*") se računa kao $\frac{Best\ found - Best\ known}{Best\ known} * 100$.

Tabela 3.2. VNS sa jednostavnom procedurom lokalne pretrage

	P	*N*	*Best-Known Value (1)*	*Best-Found Value (2)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1-pmed5	5-33	100	193.80	193.80	16.15	16.80	0.00
pmed6-pmed10	5-67	200	120.00	120.20	64.97	11.60	0.24
pmed11-pmed15	5-100	300	83.80	86.00	183.92	5.60	3.80
pmed16-pmed20	5-133	400	65.00	68.60	260.17	6.00	7.72
pmed21-pmed25	5-167	500	58.60	62.00	337.51	6.60	8.41
pmed26-pmed30	5-200	600	56.00	57.40	340.79	5.40	3.63
pmed31-pmed34	5-140	700	53.00	55.25	328.85	10.25	6.43
pmed35-pmed37	5-80	800	51.67	55.67	487.02	1.67	12.12
pmed38-pmed40	5-90	900	54.67	58.33	360.99	10.33	12.64
AVG					**246.92s**	**8.43**	**5.47%**

Tabela 3.2 pokazuje da VNS sa jednostavnom procedurom lokalne pretrage nije u stanju da vrati zadovoljavajuće rezultate. Algoritam nije pronašao najbolje poznato rešenje za 14 od 40 instanci, a u proseku, najbolje poznato rešenje je pronašao samo u 8.43 od 20 slučajeva. Ispostavlja se da je vrednost najboljeg pronađenog rešenja VNS algoritma sa sofisticiranom procedurom lokalne pretrage bolja 5.47% u proseku. Štaviše, nekoliko instanci, kao što su pmed19, pmed24, pmed33, pmed37 i pmed40, rezultovale su značajno većim odstupanjem od najboljeg poznatog rešenja. Takođe, prosečno vreme do pronalaska najboljeg rešenja je poraslo skoro osam puta, čak do 246.92 sekunde. Sve ove činjenice ukazuju na prednosti sofisticiranog algoritma lokalne pretrage, tj. mnogo bolja rešenja za značajno manje procesorskog vremena.

Tabela 3.3 sadrži rezultate izvršenja "egzaktnog" algoritma (Algoritam 3.5) nad svim instancama *OR-Library* test seta. U odnosu na prethodne, tabela je proširena kolonom "*Exact Value*", koja prikazuje da li je pronađeno tačno rešenje. Algoritam je takođe izvršen 20 puta sa istim vrednostima parametara ($k_{max} = p$ i $t_{max} = n$ sekundi).

Tabela 3.3. Rezultati VNS algoritma za prepoznavanje egzaktnih rešenja *OR-Library* test instanci

	P	*N*	*Best-Known Value (1)*	*Best-Found Value (2)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$	*Exact Value*
pmed1	5	100	268	268	0.02	3	0	
pmed2	10	100	220	220	0.11	4	0	
pmed3	10	100	208	208	0.09	3	0	
pmed4	20	100	163	163	0.07	1	0	
pmed5	33	100	110	110	0.02	16	0	
pmed6	5	200	180	180	0.23	10	0	
pmed7	10	200	143	143	0.51	3	0	
pmed8	20	200	122	122	0.48	2	0	
pmed9	40	200	85	85	0.20	3	0	
pmed10	67	200	70	70	0.10	20	0	✓
pmed11	5	300	125	125	0.52	16	0	
pmed12	10	300	112	112	1.20	5	0	
pmed13	30	300	78	78	1.00	3	0	
pmed14	60	300	60	60	0.63	1	0	✓
pmed15	100	300	44	44	0.36	20	0	✓
pmed16	5	400	98	98	1.61	16	0	
pmed17	10	400	83	83	4.26	9	0	
pmed18	40	400	62	63	2.70	0	1.61	
pmed19	80	400	42	43	1.51	0	2.38	
pmed20	133	400	40	40	0.81	20	0	✓
pmed21	5	500	85	85	5.00	14	0	
pmed22	10	500	80	80	20.38	3	0	
pmed23	50	500	49	50	6.17	0	2.04	
pmed24	100	500	35	35	2.37	1	0	
pmed25	167	500	44	44	1.48	20	0	✓
pmed26	5	600	80	80	20.52	5	0	
pmed27	10	600	67	67	20.04	7	0	
pmed28	60	600	57	57	2.86	20	0	✓
pmed29	120	600	36	36	3.02	20	0	✓
pmed30	200	600	40	40	3.06	20	0	✓
pmed31	5	700	64	64	7.32	20	0	

pmed32	10	700	72	72	4.58	20	0	✓
pmed33	70	700	35	35	16.87	8	0	
pmed34	140	700	41	41	4.50	20	0	✓
pmed35	5	800	64	64	66.86	6	0	
pmed36	10	800	58	58	50.63	8	0	
pmed37	80	800	33	33	24.95	2	0	✓
pmed38	5	900	61	61	32.37	16	0	
pmed39	10	900	74	74	9.82	20	0	✓
pmed40	90	900	29	30	18.08	0	3.45	
AVG					8.43s	9.63	0.24%	***Total: 12***

Tabela 3.3 pokazuje da "egzaktan" algoritam, iako u proseku smanjuje vreme pronalaska najboljeg rešenja, nije tako uspešan kao inicijalni algoritam. Pronašao je najbolje rešenje u 9.63 od 20 izvršenja u proseku. Bilo je nekoliko instanci (pmed18, pmed19, pmed23 i pmed40) za koje nije pronađeno najbolje poznato rešenje. Što se tiče najboljih rešenja, inicijalni algoritam je prosečno uspešniji samo za 0.24%. Sa druge strane, algoritam je uspeo da identifikuje globalno optimalna rešenja za 12 od 40 instanci, što je naznačeno u poslednjoj koloni Tabele 3.3. Značajno je primetiti da su tačno rešene uglavnom veće instance problema, tj. problemi sa većim brojem centara (većim p vrednostima). U slučaju većih p vrednosti, verovatnije je da postoji centar koji je istovremeno kritični korisnik i nije moguće pronaći bliži *backup* centar čijim dodavanjem tekućem rešenju bi se smanjila vrednost funkcije cilja (Posledica Svojstva 3.2).

Na kraju, predloženi VNS algoritam za pSCP je upoređen sa rezultatima algoritama za probleme p-centra [28], p-sledećeg centra [31] i p-medijana [15], dobijenim izvršenjem nad istim *OR-Library* test instancama. Rezultati su prikazani u Dodatku B.2.

3.4.2. Rezultati testiranja nad većim instancama problema

U cilju dalje procene performansi algoritma, generisane su test instance sa 1000, 1500, 2000 i 2500 čvorova i takođe nove instance definisane nad grafovima do 1000 čvorova sa različitim vrednostima gustina. Nove test instance su generisane kao *random k-regularni* grafovi sa unapred zadatim vrednostima broja čvorova i grana. Predloženi VNS algoritam je ponovo izvršen 20 puta sa istim vrednostima parametara ($k_{max} = p$ i $t_{max} = n$ sekundi) nad novim test primerima i rezultati su prezentovani u tabelama koje slede.

Tabela 3.4 prikazuje rezultate izvršenja nad većim test instancama i primetno je da algoritam nije bio tako uspešan kao u slučaju ostalih instanci. Pronašao je najbolje rešenje u 17.84 od 20 izvršenja u proseku. Bilo je nekoliko instanci (rndkreg13, rndkreg34 i rndkreg38) za koje su pronađena najbolja rešenja samo tri ili manje puta, ali apsolutno odstupanje najgorih i najboljih rešenja nije bilo veće od 1. Prosečno vreme pronalaska najboljeg rešenja je bilo 269.58 sekundi.

Dodatno u odnosu na prethodno objašnjene kolone, Tabela 3.5 sadrži novu kolonu ("*Density*") koja prikazuje gustinu grafa nad kojim je definisana test instanca. Na osnovu rezultata iz Tabele 3.5, uzevši u obzir nešto veće instance problema, prosečno vreme pronalaska najboljeg rešenja je u odnosu na *OR-Library* instance očekivano poraslo na 43.07 sekundi, a prosečan broj pronalazaka najboljih rešenja pao na 18.46 od 20 izvršenja. Sa druge strane, gustina grafa nije uticala na efikasnost algoritma. To je i očekivano s obzirom da algoritam ne uzima u obzir broj grana grafa, već nova rešenja za pretragu iz odgovarajućih okolina generiše samo zamenom čvorova (centara). Bilo je 8 (od ukupno 48) instanci za koje nije pronađeno najbolje rešenje u svakom od 20 izvršenja, ali je apsolutno odstupanje najgoreg i najboljeg rešenja u svim tim slučajevima samo 1.

Tabela 3.4. Rezultati za velike test instance

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
rndkreg1-rndkreg11	5-200	1000	50000	14.82	14.90	15.00	82.85	18.36
rndkreg12-rndkreg22	5-200	1500	112500	12.45	12.56	12.64	183.81	17.82
rndkreg23-rndkreg33	5-200	2000	200000	11.64	11.64	11.64	235.01	20.00
rndkreg34-rndkreg44	5-200	2500	312500	10.27	10.51	10.64	576.65	15.18
AVG							**269.58s**	**17.84**

Tabela 3.5. Rezultati za test instance definisane nad grafovima sa različitim gustinama

	P	*N*	*Density*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
rnddnskreg1-rnddnskreg4	5-200	500	50.10	8.00	8.00	8.00	13.07	20.00
rnddnskreg5-rnddnskreg8	5-200	500	60.12	7.00	7.00	7.00	4.35	20.00
rnddnskreg9-rnddnskreg12	5-200	500	80.16	6.00	6.23	6.25	17.13	15.50
rnddnskreg13-rnddnskreg16	5-200	600	50.08	7.25	7.36	7.75	67.27	17.75
rnddnskreg17-rnddnskreg20	5-200	600	60.10	7.00	7.00	7.00	6.45	20.00
rnddnskreg21-rnddnskreg24	5-200	600	80.13	5.75	5.99	6.25	51.02	15.25
rnddnskreg25-rnddnskreg28	5-200	800	50.06	6.75	6.83	7.00	97.05	18.50
rnddnskreg29-rnddnskreg32	5-200	800	60.08	6.50	6.50	6.50	37.24	20.00
rnddnskreg33-rnddnskreg36	5-200	800	80.10	6.00	6.00	6.00	28.90	20.00
rnddnskreg37-rnddnskreg40	5-200	1000	50.05	6.50	6.56	6.75	105.35	18.75
rnddnskreg41-rnddnskreg44	5-200	1000	60.06	6.50	6.50	6.50	27.76	20.00
rnddnskreg45-rnddnskreg48	5-200	1000	80.08	5.50	5.71	5.75	61.31	15.75
AVG			63.43%				**43.07s**	**18.46**

3.5. Zaključak

Problem *p*-drugog centra predstavlja uopštenje poznatog i veoma proučavanog problema *p*-centra. Dizajniran je i implementiran metod promenljivih okolina kao metaheuristički pristup rešavanju problema. Rešenje problema *p*-drugog centra predstavlja identifikaciju *p* centara sa ciljem minimizacije maksimalnog rastojanja *n* korisnika do najbližeg plus drugog-najbližeg centra.

Predložena je nova implementacija VNS metoda koja je testirana nad *OR-Library* instancama iz literature do 900 čvorova i dobijeni eksperimentalni rezultati su potvrdili visoku efikasnost algoritma u razumnom intervalu vremena. Algoritam je pronašao najbolje rešenje u proseku u 19.20 od 20 slučajeva ili u 96%. U radu je predstavljena i modifikacija inicijalno predloženog algoritma kao VNS implementacija koja je sposobna da potencijalno identifikuje globalno optimalna rešenja problema *p*-drugog centra. Testirana je nad istim *OR-Library* test setom. Ispostavilo se da "egzaktni" algoritam u proseku daje lošije rezultate u odnosu na inicijalni algoritam, ali je uspeo da identifikuje 12 tačnih rešenja od ukupno 40 *OR-Library* instanci.

U cilju potvrde efikasnosti izvornog algoritma, takođe su generisane veće test instance sa 1000, 1500, 2000 i 2500 čvorova i instance definisane nad grafovima različitih gustina (50% - 80%). Pokazalo se da gustina grafa ne utiče na efikasnost algoritma. Pronađeno je najbolje rešenje za čak 40 od ukupno 48 instanci u svakom od 20 izvršenja algoritma, ili u 92.29% u proseku. Za veće instance problema (do $n = 2500$), algoritam nije bio tako uspešan, ali je i dalje bio stabilan. Pronašao je najbolje rešenje u 89.20% izvršenja uz apsolutno odstupanje najgorih od najboljih rešenja ne veće od 1.

4. Problemi p-α-sledećih i p-α-najbližih centara

Problem p-centra je poznat i veoma proučavan problem koji podrazumeva identifikaciju p od potencijalnih n lokacija centara na takav način da se minimizuje najveće rastojanje između korisnika i najbližeg centra. Vremenom je kao potencijalni problem detektovan otkaz najbližeg centra i u tom slučaju je potrebno pronaći sledeći najbliži centar. Poznate definicije ovog problema su p-sledeći i p-drugi centar. Problem p-sledećeg centra predstavlja uopštenje p-centar problema tako što identifikuje p od potencijalnih n lokacija u cilju minimizacije maksimalnog rastojanja između korisnika i njemu najbližeg centra plus rastojanja tog centra do njemu najbližeg centra. Sa druge strane, p-drugi centar identifikuje p od potencijalnih n lokacija u cilju minimizacije maksimalne sume rastojanja između korisnika i njemu najbližeg i drugog-najbližeg centra. Generalizacijom ovih problema dolazi se do definicija problema p-α-sledećih i p-α-najbližih centara čiji je cilj minimizacija sume rastojanja korisnika do najbližeg centra i rastojanja između, odnosno do, preostalih α - *1* najbližih centara. Ukoliko α ima vrednost p, u obzir se uzimaju rastojanja svih p centara. Kao rešenje novih problema p-α-sledećih i p-α-najbližih centara u disertaciji je predložen heuristički algoritam zasnovan na metodu promenljivih okolina koji uključuje metod filtriranja potencijalnih rešenja koji donekle ubrzava konvergenciju ka lokalnom optimumu. Predloženi algoritam je testiran nad u literaturi već poznatim skupom test instanci i rezultati pokazuju da pronalazi optimalno ili približno optimalno rešenje u vremenskom intervalu srazmernom veličini problema.

4.1. Uvod

Problem p-centra (pCP) je predstavljen 1965. godine [14] kao problem identifikacije lokacija p centara u cilju minimizacije najvećeg rastojanja između korisnika i njemu dodeljenog, tj. najbližeg centra. Predstavlja formalizaciju problema određivanja lokacija za izgradnju različitih uslužnih centara, kao što su bolnice, toplane, zabavni parkovi itd., na takav način da budu što bliže svojim korisnicima.

Vremenom se pokazalo da često može doći do otkaza centra, recimo kvara u sistemu grejanja ili preopterećenja bolničkih kapaciteta. U tom slučaju korisnici moraju biti preusmereni u neki drugi centar. Kao mogući odgovori na krizne situacije definisani su problemi p-sledećeg (pNCP) i p-drugog centra (pSCP). U osnovi je ideja o unapred dodeljenom zamenskom centru koji će opsluživati korisnika u slučaju ispada primarnog centra.

Problem p-sledećeg centra, definisan u radu [1] iz 2015. godine, predstavlja identifikaciju p od mogućih n centara u cilju minimizacije maksimalne sume rastojanja korisnika do najbližeg centra i rastojanja tog centra do njemu najbližeg centra. Problem p-sledećeg centra podrazumeva najpre "posetu" primarno dodeljenom centru pa tek onda zamenskom, u slučaju otkaza primarnog. U današnje vreme se informacije brzo šire, tako da je za očekivati da se unapred sazna za kvar primarnog centra. U tom slučaju odmah se posećuje zamenski centar. Problem p-drugog centra upravo identifikuje p od potencijalnih n lokacija centara u cilju minimizacije maksimalne sume rastojanja između korisnika i njemu najbližeg i drugog-najbližeg centra. Međutim, postavlja se pitanje šta u slučaju otkaza i primarnog i zamenskog centra. Kao potencijalni odgovor, moguće je posmatrati proširenja p-sledeći i p-drugi centar problema na probleme kojima se ukupan broj primarnog i zamenskih centara prosleđuje kao parametar α. Tako se dolazi do definicija p-α-sledećih i p-α-najbližih centara kao problema kod kojih je potrebno minimizirati rastojanje do α centara.

Problem p-α-sledećih centara (*eng. p-α-next centers problem*, pαNCP) kao generalizacija problema p-sledećeg centra predstavlja identifikaciju lokacija p od mogućih n centara u cilju minimizacije maksimalne sume rastojanja korisnika do najbližeg centra, rastojanja tog centra do njemu najbližeg kao i međusobnih rastojanja narednih α - *2* parova centara i njima najbližih centara. Formalno, neka je $G = (V, E)$ neusmeren težinski graf gde su težine grana određene rastojanjem između svojih krajeva, V skup svih čvorova, a E skup grana grafa. Centri, kao i ostali korisnici predstavljaju čvorove grafa, a $d(i, j)$ je najkraće rastojanje između čvorova i i j, izračunato kao rezultat algoritma određivanja najkraćih rastojanja u grafu G. Ukoliko u skupu E ne postoje grane koje posredno ili neposredno povezuju i i j čvorove, onda je $d(i, j) = \infty$. Problem p-α-sledećih centara se definiše kao:

$$f_n(P,V) = \max_{i \in V} \left\{ \min_{j_1 \in P} d(i,j_1) + \min_{\substack{k_1 \in \left\{ \arg\min_{j_1 \in P} d(i,j_1) \right\} \\ j_2 \in P \setminus \{k_1\}}} d(k_1,j_2) + \sum_{l=3}^{\alpha} \min_{\substack{k_1 \in \left\{ \arg\min_{j_1 \in P} d(i,j_1) \right\} \\ k_2 \in \left\{ \arg\min_{j_2 \in P \setminus \{k_1\}} d(k_1,j_2) \right\} \\ \vdots \\ k_{l-1} \in \left\{ \arg\min_{j_{l-1} \in P \setminus \{k_1,k_2,\ldots,k_{l-2}\}} d(k_{l-2},j_{l-1}) \right\} \\ j_l \in P \setminus \{k_1,k_2,\ldots,k_{l-1}\}}} d(k_{l-1},j_l) \right\},$$

$$p\alpha NCP(V) = \min_{\substack{P \subset V \\ |P|=p}} f_n(P,V). \tag{4.1}$$

Problem *p*-*α*-najbližih centara (*eng. p-α-closest centers problem*, pαCCP) je generalizacija problema *p*-drugog centra i predstavlja identifikaciju lokacija *p* od mogućih *n* centara u cilju minimizacije maksimalne sume rastojanja korisnika do α najbližih centara. Formalno, definisan nad istim grafom kao u slučaju pαNCP, rešenje problema *p*-*α*-najbližih centara je skup čvorova $P \subset V$, kardinalnosti *p*, koji sadrži centre tako da je, uzevši sve korisnike u obzir, maksimalna suma udaljenosti korisnika do α najbližih centara minimizovana:

$$f_c(P,V) = \max_{i \in V} \left\{ \min_{j_1 \in P} d(i,j_1) + \min_{\substack{k_1 \in \left\{ \arg\min_{j_1 \in P} d(i,j_1) \right\} \\ j_2 \in P \setminus \{k_1\}}} d(i,j_2) + \sum_{l=3}^{\alpha} \min_{\substack{k_1 \in \left\{ \arg\min_{j_1 \in P} d(i,j_1) \right\} \\ k_2 \in \left\{ \arg\min_{j_2 \in P \setminus \{k_1\}} d(i,j_2) \right\} \\ \vdots \\ k_{l-1} \in \left\{ \arg\min_{j_{l-1} \in P \setminus \{k_1,k_2,\ldots,k_{l-2}\}} d(i,j_{l-1}) \right\} \\ j_l \in P \setminus \{k_1,k_2,\ldots,k_{l-1}\}}} d(i,j_l) \right\},$$

$$p\alpha CCP(V) = \min_{\substack{P \subset V \\ |P|=p}} f_c(P,V). \tag{4.2}$$

U disertaciji je predstavljen heuristički algoritam zasnovan na metodu promenljivih okolina za rešavanje novih problema *p*-*α*-sledećih i *p*-*α*-najbližih centara. Algoritam proizvodi optimalna ili približno optimalna rešenja za proizvoljno izabran broj dodeljenih centara ($2 \leq \alpha \leq p$). Shodno tome, u narednim poglavljima sledi opis algoritma, uz poseban osvrt na metod filtriranja potencijalnih rešenja, kao i rezultati testiranja nad u literaturi poznatim skupom test instanci, *OR-Library* [2].

4.2. Algoritam za rešavanje problema *p*-*α*-sledećih i *p*-*α*-najbližih centara

Algoritam za rešavanje problema *p*-*α*-sledećih i *p*-*α*-najbližih centara je takođe izgrađen na, u Poglavlju 1.2 već opisanom, metaheurističkom metodu promenljivih okolina (VNS). Predloženi algoritam za pαNCP i pαCCP se može predstaviti kao generalizacija *"Filtered variable neighborhood search method for the p-next center problem"* algoritma iz rada [31]. U osnovi je preuzeta ideja iz rada [31] o filtriranju potencijalno novih rešenja za dalju pretragu. Nakon uspostavljenog kriterijuma filtriranja, rešenja iz skupa $N_k(P)$ koja ne zadovoljavaju kriterijum se odbacuju, što smanjuje prostor pretrage i na taj način u srednjem slučaju ubrzava konvergenciju ka optimalnom rešenju.

Problemi *p*-*α*-sledećih i *p*-*α*-najbližih centara predstavljaju probleme minimizacije maksimalnog rastojanja korisnika do dodeljenih centara. Korisnik koji odgovara maksimalnom rastojanju označen je kao kritični korisnik, tako da vrednost funkcije cilja odgovara rastojanju kritičnog korisnika do svojih centara. U cilju optimizacije, tj. smanjenja vrednosti funkcije cilja, potrebno je uključivanjem novog centra u rešenje smanjiti ovo rastojanje. Kriterijum pretrage novih centara je definisan upravo tako da ga zadovoljavaju samo potencijalno novi centri koji smanjuju vrednost funkcije cilja kritičnog korisnika.

Svojstvo 4.1. Neka je u_c kritični korisnik, P tekuće rešenje koje sadrži p centara, a $f(P, u_c)$ vrednost funkcije cilja kritičnog korisnika. Tada, ukoliko postoji bolje rešenje P' u okolini $N_1(P)$, važi da je $f(P', u_c) < f(P, u_c)$.

Dokaz. Neka je u_c' novi kritični korisnik u rešenju P'. Suprotno Svojstvu 4.1, važilo bi da je $f(P, u_c) \leq f(P', u_c) \leq f(P', u_c')$, što je u kontradikciji sa činjenicom da je novo rešenje P' bolje od tekućeg rešenja P. ■

Na osnovu Svojstva 4.1, u svakoj iteraciji algoritma pretrage teži se smanjenju vrednosti funkcije cilja kritičnog korisnika u_c, tako što se jedan od centara tekućeg rešenja P zamenjuje novim centrom c_{in} ($c_{in} \notin P$). Centar c_{in} mora da zadovolji kriterijum filtriranja, tj. $\exists(c_{out} \in P)\big(f(P \setminus \{c_{out}\} \cup \{c_{in}\}, u_c) < f(P, u_c)\big)$. Predloženi algoritam za probleme *p*-*α*-sledećih i *p*-*α*-najbližih centara implementira VNS metod uz filtriranje potencijalnih rešenja na osnovu pomenutog kriterijuma. U cilju efikasne implementacije metoda filtriranja i izračunavanja vrednosti funkcije cilja, za svakog od korisnika održava se samo-balansirajuće binarno stablo pretrage koje sadrži centre tekućeg rešenja sortirane po udaljenosti od korisnika.

Algoritam podrazumeva da tekuće rešenje P predstavlja prvih p elemenata niza x_{cur}, a ostali korisnici su $n - p$ elemenata sa kraja istog niza. Dodatno, prezentovani algoritam koristi sledeće strukture i funkcije:

- *dist(u, v)* – najkraće rastojanje između čvorova *u* i *v*,
- *centers(u)* – binarno stablo pretrage centara tekućeg rešenja korisnika *u*,
- *f(α, u, c_{in}, c_{out}, centers)* – vrednost funkcije cilja korisnika *u* nakon što se centar c_{out} tekućeg rešenja zameni novim centrom c_{in}.

Algoritam 4.1: Zamena čvorova u N_1 okolini u kontekstu problema *p*-*α*-sledećih i *p*-*α*-najbližih centara

Move(α, x_{cur}, c_{in}, centers)
Initialization:
$f \leftarrow \infty$; $u_c \leftarrow$ *null*; *in* $\leftarrow x_{cur}(c_{in})$; $c_{out} \leftarrow$ *null*
Best deletion:
For Each *i = {1, ..., p}*
    out $\leftarrow x_{cur}(i)$
    $f_{max} \leftarrow 0$
    For Each *user* = $x_{cur}(1)$, …, $x_{cur}(n)$
        value $\leftarrow f(\alpha,$ *user, in, out, centers)*
        If $f_{max} <$ *value*
            $f_{max} \leftarrow$ *value*
            $u_c' \leftarrow$ *user*
        End If
    End For Each
    If $f > f_{max}$
        $f \leftarrow f_{max}$
        $c_{out} \leftarrow i$
        $u_c \leftarrow u_c'$
    End If
End For Each
Return *f*, c_{out}, u_c

Move procedura (Algoritam 4.1) identifikuje centar tekuće pretraživanog rešenja koji bi trebalo zameniti novim centrom u cilju maksimalnog unapređenja vrednosti funkcije cilja, a ujedno i određuje vrednost funkcije cilja novog rešenja, kao i novog kritičnog korisnika. Pored vrednosti α i niza stabala *centers*, ulazne vrednosti su tekuće rešenje i indeks novog centra. Uz pretpostavku brisanja jednog po jednog centra, izračunava se vrednost funkcije cilja svih korisnika. Na osnovu toga, vremenska složenost je $O(pn) * O(f)$, gde je $O(f)$ vremenska kompleksnost algoritma izračunavanja vrednosti funkcije cilja za jednog korisnika.

Algoritam 4.2: Ažuriranje binarnih stabala pretrage koja sadrže centre tekućeg rešenja

***Update*(x_{cur}, c_{in}, c_{out}, *centers*)**
in ← $x_{cur}(c_{in})$
out ← $x_{cur}(c_{out})$
For Each *user* = $x_{cur}(1)$, …, $x_{cur}(n)$
    erase(out, centers(user))
    insert(in, centers(user))
End For Each
Return *centers*

Update procedura (Algoritam 4.2), za sve korisnike ažurira samo-balansirajuća binarna stabla pretrage koja sadrže centre tekućeg rešenja. Kako svaki od korisnika ima svoje stablo, održava se niz tih stabala. U svako od balansiranih stabala se unosi novi centar c_{in} i briše centar c_{out}. Shodno tome, vremenska složenost *Update* procedure je $O(n \log p)$.

Algoritam 4.3: Provera da li je moguće pronaći potencijalno bolje rešenje nakon otvaranja novog centra c_{in}

***ExistsRelaxedDistance*(α, x_{cur}, u_c, c_{in}, f_{cur}, *centers*)**
in ← $x_{cur}(c_{in})$
For Each *out* = $x_{cur}(1)$, …, $x_{cur}(p)$
    If $f(\alpha, u_c, in, out, centers) < f_{cur}$
        Return *True*
    End If
End For Each
Return *False*

ExistsRelaxedDistance procedura (Algoritam 4.3) proverava da li je moguće unaprediti vrednost funkcije cilja dodavanjem centra c_{in} i brisanjem nekog od centara tekućeg rešenja. Vremenska složenost je $O(p) * O(f)$, gde je $O(f)$ vremenska složenost algoritma izračunavanja vrednosti funkcije cilja kritičnog korisnika.

Ulazne vrednosti *LocalSearchVertexSubstitution* procedure (Algoritam 4.4) su vrednost α, tekuće rešenje, kritični korisnik, tekuća vrednost funkcije cilja i niz balansiranih binarnih stabala pretrage koja sadrže centre. Metod lokalne pretrage pomoću *Move* procedure u svakoj iteraciji pronalazi optimalan par centara čija razmena rezultira najvećom redukcijom vrednosti funkcije cilja. Izvršava se dokle god je moguće pronaći takav par centara i na taj način rezultira lokalnim optimumom. Nakon razmene centara i ažuriranja stabala pretrage za svakog od korisnika, procedura kao izlazne vrednosti daje novo tekuće rešenje, novog kritičnog korisnika i novu vrednost funkcije cilja.

LocalSearchVertexSubstitution procedura se oslanja na Svojstvo 4.1 i u srednjem slučaju ubrzava konvergenciju ka lokalnom optimumu. Ipak, složenost u najgorem slučaju ostaje nepromenjena. Složenost jedne iteracije *Main petlje (loop)* se izračunava kao $O(pn) * O(f) + k * O(pn) * O(f) + O(n \log p)$, gde je k broj potencijalnih centara koji zadovoljavaju kriterijum pretrage. U najgorem slučaju $k = n - p$, te je složenost $O(pn) * O(f) + n * O(pn) * O(f) + O(n \log p) \approx O(pn^2) * O(f)$. Sa druge strane, prostor pretrage se sa svakom iteracijom smanjuje, samim tim i vrednost k pa tako i vremenska složenost iteracije. U krajnjem slučaju k je 0 pa je vremenska kompleksnost u najboljem slučaju redukovana na $O(pn) * O(f) + O(n \log p) \approx O(pn) * O(f)$.

Algoritam 4.4: Procedura zamene čvorova u fazi lokalne pretrage za probleme *p*-α-sledećih i *p*-α-najbližih centara

LocalSearchVertexSubstitution(α, x_{cur}, u_c, f_{cur}, centers)
Main loop:
While *True*
    $f' \leftarrow \infty$; $u_c' \leftarrow null$

    For Each *in = p +1, ..., n*
        Find the optimal objective function value improvement within $N_1(x_{cur})$ neighborhood:
        If *ExistsRelaxedDistance(α, x_{cur}, u_c, in, f_{cur}, centers)*
            f'', out, u_c'' ← Move(α, x_{cur}, in, centers)
            If $f'' < f'$
                $f' \leftarrow f''$
                $c_{in} \leftarrow in$
                $c_{out} \leftarrow out$
                $u_c' \leftarrow u_c''$
            End If
        End If
    End For Each

    If $f_{cur} \leq f'$
        There is not found improvement in the neighborhood:
        Break Main loop
    End If

    Update(x_{cur}, c_{in}, c_{out}, centers)
    $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
    $u_c \leftarrow u_c'$
    $f_{cur} \leftarrow f'$
End While

Return x_{cur}, u_c, f_{cur}

VNS algoritam za probleme *p*-α-sledećih i *p*-α-najbližih centara je predstavljen Algoritmom 4.5. VNS (kao i *LocalSearchVertexSubstitution*) implementacija je što se strukture koda tiče identična i preuzeta iz prethodnih poglavlja na temu problema *p*-sledećeg i *p*-drugog centra. Ulazne vrednosti *VariableNeighborhoodSearch* procedure su α, k_{max} i t_{max} kao vremenski limit izvršavanja algoritma. Pretraga počinje u N_1 okolini ($k = 1$) slučajno izabranog rešenja, uz filtriranje rešenja kao i u slučaju *LocalSearchVertexSubstitution* procedure. Ukoliko lokalna pretraga rezultira pronalaskom jednakog ili boljeg rešenja, to rešenje postaje optimalno, a vrednost k se resetuje na 1, tj. naredno rešenje se traži kao slučajno izabrano rešenje iz okoline N_1 novog rešenja. Sa druge strane, u slučaju da se nakon faze lokalne pretrage ne pronađe bolje ili jednako rešenje, pretraga se nastavlja u N_{k+1} okolini trenutno optimalnog (najboljeg do sada pronađenog) rešenja. Ukoliko k prevaziđe svoju maksimalno dozvoljenu vrednost k_{max}, resetuje se na 1. *Main step* se ponavlja sve dok se ne dostigne maksimalno dozvoljeno vreme izvršenja t_{max}. Algoritam vraća najbolje rešenje pronađeno tokom pretrage.

Algoritam 4.5: VNS algoritam za probleme *p*-*α*-sledećih i *p*-*α*-najbližih centara

VariableNeighborhoodSearch(α, k_{max}, t_{max})
Initialization:
Randomly initialize x_{opt}; according to x_{opt} initialize array "centers", f_{opt}, u_c; copy initial solution into the current one, i.e., copy f_{opt}, x_{opt}, centers and u_c* into f_{cur}, x_{cur}, $centers_{cur}$ and u_{cur}*, respectively.*
Repeat the Main step until the stopping condition is met (e.g., time ≤ t_{max})
Main step:
k ← 1
While *k ≤ k_{max}*
Shaking operator:
*****generate a solution at random from kth neighborhood*****
For Each *j = 1, ..., k*
- *Take center to be inserted (c_{in}) at random if ExistsRelaxedDistance(α, x_{cur}, u_{cur}*, c_{in}, f_{cur}, $centers_{cur}$);*
- *Find center to be deleted (c_{out}) at random;*
- *Update x_{cur}, $centers_{cur}$ i.e., execute:*
  Update(x_{cur}, c_{in}, c_{out}, $centers_{cur}$)
  $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
- *Update f_{cur} and u_{cur}* according to x_{cur}, $centers_{cur}$*

End For Each
Local search:
If *any potentially better solution found*
x_{cur}, u_{cur}, f_{cur} ← LocalSearchVertexSubstitution(α, x_{cur}, u_{cur}*, f_{cur}, $centers_{cur}$)*
Move or not:
If *$f_{cur} \leq f_{opt}$*
*****Save current solution as the optimal; return to N_1*****
$x_{opt} \leftarrow x_{cur}$; $f_{opt} \leftarrow f_{cur}$; u_c ← u_{cur}*; centers ← $centers_{cur}$;*
k ← 1
Else
*****Current solution is the optimal; change the neighborhood*****
$x_{cur} \leftarrow x_{opt}$; $f_{cur} \leftarrow f_{opt}$; u_{cur} ← u_c*; $centers_{cur}$ ← centers;*
k ← k + 1
End If
End If
End While
Return *x_{opt}, f_{opt}*

Konačno, u nastavku slede i procedure za izračunavanje vrednosti funkcije cilja na osnovu održavanih binarnih stabala pretrage. Za svakog od korisnika, stablo sadrži centre tekućeg rešenja sortirane na osnovu udaljenosti od korisnika, počev od najbližeg centra.

Algoritam 4.6: Izračunavanje vrednosti funkcije cilja za problem *p*-*α*-sledećih centara

f(α, user, in, out, centers)
u ← user; value ← 0; P ← {}
While *|P| < α*
*c ← **closest** center **from** centers(u) ∪{in} \ {out} **where** center ∉ P*
value ← value + dist(u, c)
P ← P ∪{c}
u ← c
End While
Return *value*

Algoritam 4.6 izračunava vrednost funkcije cilja korisnika *user* u slučaju problema *p*-*α*-sledećih centara. U *i*-toj iteraciji *While* petlje, u najgorem slučaju posećuje se *i* centara, tako da je vremenska složenost *i*-te iteracije $O(i)$. Na osnovu toga, vremenska složenost procedure izračunavanja vrednosti funkcije cilja u najgorem slučaju je $O(1 + 2 + ... + \alpha) = O\left(\frac{\alpha*(\alpha+1)}{2}\right) \approx O(\alpha^2)$. U najboljem slučaju je $O(\alpha)$. Prema tome, složenost jedne iteracije *Main petlje* procedure lokalne pretrage (Algoritam 4.4) za problem *p*-*α*-sledećih centara u najgorem slučaju je $O(\alpha^2 pn^2)$, ili $O(p^3 n^2)$ ukoliko je $\alpha = p$. U najboljem slučaju, složenost je $O(\alpha pn)$, odnosno $O(p^2 n)$ za $\alpha = p$. Kod velikih instanci problema, važi da je $p << n$, pa je vremenska složenost jedne iteracije procedure lokalne pretrage $O(n^2)$ u najgorem slučaju, odnosno $O(n)$ u najboljem slučaju.

Algoritam 4.7: Izračunavanje vrednosti funkcije cilja za problem *p*-*α*-najbližih centara

```
f(α, user, in, out, centers)
value ← 0
P ← take α closest centers from centers(user) ∪ {in} \ {out}
For Each center = P(1), ..., P(α)
        value ← value + dist(user, center)
End For Each
Return value
```

Algoritam 4.7 izračunava vrednost funkcije cilja korisnika *user* u slučaju problema *p*-*α*-najbližih centara. Vremenska složenost algoritma je $O(\alpha)$. Prema tome, složenost jedne iteracije *Main petlje* procedure lokalne pretrage (Algoritam 4.4) za problem *p*-*α*-najbližih centara u najgorem slučaju je $O(\alpha pn^2)$, ili $O(p^2 n^2)$ ukoliko je $\alpha = p$. U najboljem slučaju, složenost je $O(\alpha pn)$, odnosno $O(p^2 n)$ za $\alpha = p$. Za velike instance problema ($p << n$), isto kao i kod problema *p*-*α*-sledećih centara, vremenska složenost jedne iteracije procedure lokalne pretrage je $O(n^2)$ u najgorem slučaju, odnosno $O(n)$ u najboljem slučaju.

4.3. Rezultati algoritma za probleme *p*-*α*-sledećih i *p*-*α*-najbližih centara

Algoritam je implementiran u programskom jeziku C++, a sva testiranja su izvršena na *Intel Core i7-8700K (3.7GHz) CPU* sa *32GB RAM*-a konfiguraciji. Za potrebe testiranja, preuzet je *OR-Library* [2] skup podataka, inicijalno namenjen za testiranje *p*-medijan problema. Biblioteka sadrži 40 test instanci sa 100 do 900 čvorova i *p* između 5 i 200 (ne više od *n/3*, gde je *n* broj čvorova).

Predloženi algoritmi su izvršeni po 20 puta nad svakom test instancom, uvek počevši od različitog inicijalnog rešenja. Isprobane su različite kombinacije parametara $\alpha = 2$ i $\alpha = p$, $k_{max} = p/2$, $k_{max} = p$, kao i $t_{max} = n$, $t_{max} = 2n$ i $t_{max} = 5n$. Ispostavilo se da je u slučaju većih problema potrebno znatno više vremena da se pronađe najbolje rešenje. Stoga, algoritmi su ponovo izvršeni po 20 puta nad svakom test instancom, ali za vrednosti parametara $\alpha = 2$ i $\alpha = p$, $k_{max} = p$ i $t_{max} = \alpha n$ sekundi. Dobijeni rezultati su predstavljeni u narednim tabelama.

4.3.1. Rezultati testiranja nad *OR-Library* instancama za $\alpha = p$

Tabele 4.1 i 4.2 prikazuju zbirne rezultate testiranja algoritma za probleme *p*-*α*-sledećih i *p*-*α*-najbližih ($\alpha = p$) centara nad instancama *OR-Library* test biblioteke. Detaljni rezultati su dati u Tabelama C.1 i C.2 u okviru Dodatka C. Prva kolona tabela sadrži naziv test instance, naredne tri kolone predstavljaju vrednosti *p* (broj centara), *n* (broj korisnika) i *m* (broj grana grafa). U kolonama "*Best Value*", "*AVG Value*" i "*Worst Value*" prikazana je najbolja, prosečna i najgora vrednost rešenja koje je algoritam pronašao tokom 20 izvršenja. Kolona "*Time*" (ili *time-to-target*) prikazuje prosečno vreme u sekundama koje je bilo potrebno da se prvi put pronađe najbolje rešenje. Dakle, "*Time*" nije ukupno vreme izvršavanja algoritma, već vreme pronalaska najboljeg rešenja. Algoritam se izvršava sve dok ne istekne vremenski limit, tj. *αn* sekundi, ili dok se ne izvrši poslednja faza lokalne pretrage. Kolona "*#Best*" daje broj koliko puta je algoritam pronašao najbolje rešenje u 20 izvršenja. Takođe, u Dodatku C, uključene su dodatne kolone koje sadrže procentualno odstupanje prosečnog

i najgoreg rešenja u odnosu na najbolje poznato rešenje dato u koloni "*Best-Known Value*". Kako *OR-Library* instance još uvek nisu korišćene za testiranje problema *p*-*α*-sledećih i *p*-*α*-najbližih centara za $\alpha = p$, kao najbolje poznato rešenje uzeto je najbolje rešenje koje je pronašao predloženi VNS algoritam.

Tabela 4.1 sadrži rezultate algoritma za problem *p*-*α*-sledećih centara, gde je $\alpha = p$. Primetno je da sa porastom veličine test instanci, očekivano raste i vreme pronalaska najboljeg rešenja. Prosečno vreme nad kompletnim test skupom podataka je 22439.83 sekunde, dok je za najmanje instance (pmed1 - pmed5) 296.98 sekundi. Algoritam je iz 20 izvršenja uspeo da pronađe najbolje poznato rešenje prosečno u 8.85 slučajeva. Prosečna odstupanja najgorih i srednjih rešenja u odnosu na najbolja rešenja su 18.92% i 7.05% (iz Tabele C.1 u Dodatku C). Prethodno ukazuje na to da bi samo najbolja rešenja trebalo uzimati u obzir, a naročito u slučaju većih instanci problema *p*-*α*-sledećih centara sa većim *p* i *n* vrednostima. Na osnovu dobijenih rezultata, da se zaključiti da predloženi algoritam, pronalazi optimalna ili skoro-optimalna rešenja u vremenskom intervalu srazmernom veličini problema i da se, uz višestruko izvršenje, može uspešno koristiti za rešavanje problema *p*-*α*-sledećih centara veličine do 900 čvorova.

Tabela 4.1. Rezultati za pαNCP algoritam ($\alpha = p$, $t_{max} = \alpha n$) testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
pmed1-pmed5	5-33	100	200	399.00	403.54	406.40	296.98	16.40
pmed6-pmed10	5-67	200	800	303.80	318.92	349.20	3052.68	9.40
pmed11-pmed15	5-100	300	1800	339.80	362.61	413.60	8450.52	5.00
pmed16-pmed20	5-133	400	3200	377.00	424.06	498.40	16375.86	6.20
pmed21-pmed25	5-167	500	5000	401.40	458.38	515.20	32340.45	7.60
pmed26-pmed30	5-200	600	7200	468.00	544.29	628.00	56892.34	5.40
pmed31-pmed34	5-140	700	9800	286.50	338.63	409.00	41349.02	8.75
pmed35-pmed37	5-80	800	1280	152.67	176.67	206.00	20156.68	9.67
pmed38-pmed40	5-90	900	16200	171.67	196.27	238.00	28227.64	13.33
AVG							**22439.83s**	**8.85**

Tabela 4.2 sadrži rezultate algoritma za problem *p*-*α*-najbližih centara, gde je $\alpha = p$. Primetno je da su rezultati bolji u odnosu na problem *p*-*α*-sledećih centara, prvenstveno u pogledu stabilnosti i broja pronalazaka najboljih rešenja, u proseku 13.38 od 20 izvršenja. Sa druge strane, vreme pronalaska najboljeg rešenja ostaje prilično veliko, 13040.57 sekundi. Prema tome, očekivano je da sa daljim povećanjem vremenskog limita za izvršenje algoritma, rezultati algoritma i nad instancama problema *p*-*α*-najbližih, kao i *p*-*α*-sledećih centara, budu još bolji. To potvrđuju i rezultati testiranja za manje vrednosti vremenskog limita $t_{max} = n$, $t_{max} = 2n$ i $t_{max} = 5n$. Ovi rezultati su izostavljeni iz rada zbog lošijih vrednosti u odnosu na rezultate testiranja sa limitom $t_{max} = \alpha n$ sekundi. Odstupanja vrednosti najgorih i srednjih rešenja u odnosu na najbolja pronađena rešenja potvrđuju stabilnost algoritma za rešavanje problema *p*-*α*-najbližih centara. Srednje vrednosti odstupanja predstavljene u Tabeli C.2 (Dodatak C) iznose samo: 0.33% u slučaju najgorih i 0.13% u slučaju vrednosti srednjih rešenja.

Tabela 4.2. Rezultati za pαCCP algoritam ($\alpha = p$, $t_{max} = \alpha n$) testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
pmed1-pmed5	5-33	100	200	2707.80	2708.18	2708.60	250.85	17.00
pmed6-pmed10	5-67	200	800	3060.60	3061.50	3063.60	956.64	16.00
pmed11-pmed15	5-100	300	1800	3485.60	3488.24	3493.00	5432.34	13.00
pmed16-pmed20	5-133	400	3200	3463.80	3468.64	3477.00	12741.40	9.80
pmed21-pmed25	5-167	500	5000	3895.80	3902.19	3912.80	25090.23	7.20
pmed26-pmed30	5-200	600	7200	4710.40	4712.28	4715.40	26166.34	14.00
pmed31-pmed34	5-140	700	9800	3086.50	3089.71	3091.75	22337.63	15.25
pmed35-pmed37	5-80	800	1280	1425.33	1425.33	1425.33	5140.81	20.00
pmed38-pmed40	5-90	900	16200	1425.33	1427.33	1429.67	21220.28	9.67
AVG							**13040.57s**	**13.38**

4.3.2. Rezultati testiranja nad *OR-Library* instancama za $\alpha = 2$

U cilju upoređivanja rezultata sa algoritmima iz prethodnih poglavlja, predloženi algoritmi za probleme *p*-α-sledećih i *p*-α-najbližih centara su testirani i za vrednost parametra $\alpha = 2$ ($k_{max} = p$ i $t_{max} = 2n$ sekundi). Problem *p*-*2*-sledećih centara je ekvivalentan problemu *p*-sledećeg centra, a problem *p*-*2*-najbližih centara problemu *p*-drugog centra. Stoga, dobijeni rezultati su upoređeni sa do sada najefikasnijim heurističkim algoritmima za rešavanje problema *p*-sledećeg (Poglavlje 2) i *p*-drugog (Poglavlje 3) centra. Svi pomenuti algoritmi su testirani na istoj hardverskoj konfiguraciji, tako da su sva poređenja potpuno relevantna.

Predloženi algoritmi za probleme *p*-α-sledećih i *p*-α-najbližih centara, gde je $\alpha = 2$, izvršeni su po 20 puta i zbirni rezultati su predstavljeni u tabelama koje slede. Tabele su proširene kolonom "*Best-Known Value*" koja sadrži do sada najbolja poznata rešenja problema, tj. prikazuje najbolja rešenja VNS algoritma iz Poglavlja 2 ove disertacije za problem *p*-sledećeg centra i najbolja rešenja VNS algoritma iz Poglavlja 3 za problem *p*-drugog centra. U tabelama, prikazano je takođe najbolje rešenje predloženog algoritma dobijeno u 20 izvršenja (kolona "*Best-Found Value*") i prosečno vreme pronalaska najboljeg rešenja (kolona "*Time*"). Kolona "*#Best-Known*" sadrži broj izvršenja algoritma koja su rezultirala pronalaskom najboljeg poznatog rešenja. Konačno, poslednja kolona daje procentualno odstupanje najboljeg pronađenog u odnosu na najbolje poznato rešenje. Procenat odstupanja se računa kao $\frac{Best\ found - Best\ known}{Best\ known} * 100$. U Tabelama C.3, C.4 i C.5 (Dodatak C) dati su detaljni rezultati izvršenja nad svakom instancom *OR-Library* test skupa, uključujući vrednosti najgorih i srednjih rešenja, kao i njihova odstupanja u odnosu na vrednosti najboljih poznatih rešenja.

Tabela 4.3. Rezultati za pαNCP algoritam ($\alpha = 2$, $t_{max} = 2n$) testiran nad *OR-Library* test instancama

	P	*N*	*Best-Known Value*	*Best-Found Value*	*Time*	*#Best-Known*	*Gap (best found-best known) /best known*100*
pmed1-pmed5	5-33	100	131.00	131.20	9.02	16.00	0.17
pmed6-pmed10	5-67	200	82.60	82.60	82.00	16.00	0.00
pmed11-pmed15	5-100	300	58.60	58.80	142.05	15.80	0.43
pmed16-pmed20	5-133	400	44.40	45.20	318.72	11.80	2.50
pmed21-pmed25	5-167	500	41.20	42.20	342.48	10.60	3.13
pmed26-pmed30	5-200	600	43.60	43.80	172.97	16.00	0.53
pmed31-pmed34	5-140	700	42.50	44.75	224.53	15.00	10.23
pmed35-pmed37	5-80	800	37.00	37.00	256.23	18.67	0.00
pmed38-pmed40	5-90	900	45.67	47.00	332.94	13.33	5.80
AVG					**200.05s**	**14.68**	**2.30%**

U poređenju sa rezultatima algoritma iz Poglavlja 2, predloženi algoritam za problem *p*-sledećeg centra ($\alpha = 2$) pronalazi najbolja poznata rešenja prosečno u 14.68 od 20 izvršenja (Tabela 4.3), ili u 73.38% slučajeva. Bilo je čak 7 instanci (pmed4, pmed13, pmed19, pmed23, pmed27, pmed33 i pmed40) za koje nisu pronađena najbolja poznata rešenja. Prosečno odstupanje najboljih pronađenih u odnosu na najbolja poznata rešenja je 2.30%. Iz Tabele C.3 (Dodatak C) se vidi da su odstupanja najgorih i srednjih rešenja, od 6.60% i 3.79%, respektivno, takođe velika. Prema tome, rezultati izvršenja algoritma sa vremenskim ograničenjem $t_{max} = 2n$ sekundi nisu bili tako dobri. Stoga, u narednoj tabeli su prikazani i rezultati testiranja algoritma za isti problem, ali sa povećanim vremenskim ograničenjem izvršenja, $t_{max} = 5n$ sekundi.

Tabela 4.4. Rezultati za pαNCP algoritam ($\alpha = 2$, $t_{max} = 5n$) testiran nad *OR-Library* test instancama

	P	*N*	*Best-Known Value*	*Best-Found Value*	*Time*	*#Best-Known*	*Gap (best found-best known) /best known*100*
pmed1-pmed5	5-33	100	131.00	131.00	67.75	18.80	0.00
pmed6-pmed10	5-67	200	82.60	82.60	153.76	13.40	0.00
pmed11-pmed15	5-100	300	58.60	58.60	166.89	17.40	0.00
pmed16-pmed20	5-133	400	44.40	44.60	196.68	11.80	0.43
pmed21-pmed25	5-167	500	41.20	41.40	314.11	10.80	0.63
pmed26-pmed30	5-200	600	43.60	43.60	238.27	15.00	0.00
pmed31-pmed34	5-140	700	42.50	43.25	520.16	14.50	3.41
pmed35-pmed37	5-80	800	37.00	37.00	115.01	19.67	0.00
pmed38-pmed40	5-90	900	45.67	45.67	681.86	18.00	0.00
AVG					**253.96s**	**15.18**	**0.47%**

Tabela 4.4 sadrži rezultate algoritma za *p*-sledeći centar problem ($\alpha = 2$), izvršen za vremenski limit $t_{max} = 5n$ sekundi (i $k_{max} = p$). Sa povećanjem dozvoljenog vremena izvršavanja dobijeni su bolji rezultati. U poređenju sa rezultatima algoritma iz Poglavlja 2, najbolja poznata rešenja pronađena su prosečno u 15.18 od 20 izvršenja, ili u 75.88% slučajeva. Ostalo je nekoliko instanci (pmed17, pmed23 i pmed33) za koje nisu pronađena najbolja poznata rešenja, ali je prosečno odstupanje najboljih pronađenih u odnosu na najbolja poznata rešenja samo 0.47%. Iz Tabele C.4 (Dodatak C) se vidi da su čak i odstupanja najgorih i srednjih rešenja, od 3.31% i 1.35%, respektivno, sada smanjena. To potvrđuje stabilnost i efikasnost algoritma u rešavanju problema *p*-sledećeg centra, ali uz veći utrošak procesorskog vremena. Srednje vreme pronalaska najboljeg rešenja je 253.96 sekundi, što je znatno više od 20.86 sekundi koliko je potrebno algoritmu iz Poglavlja 2.

Tabela 4.5. Rezultati za pαCCP algoritam ($\alpha = 2$, $t_{max} = 2n$) testiran nad *OR-Library* test instancama

	P	*N*	*Best-Known Value*	*Best-Found Value*	*Time*	*#Best-Known*	*Gap (best found-best known) /best known*100*
pmed1-pmed5	5-33	100	193.80	193.80	6.70	19.20	0.00
pmed6-pmed10	5-67	200	120.00	120.00	26.19	19.20	0.00
pmed11-pmed15	5-100	300	83.80	83.80	48.41	20.00	0.00
pmed16-pmed20	5-133	400	65.00	65.20	112.92	13.00	0.32
pmed21-pmed25	5-167	500	58.60	58.80	222.50	13.00	0.41
pmed26-pmed30	5-200	600	56.00	56.00	134.39	19.60	0.00
pmed31-pmed34	5-140	700	53.00	53.00	145.39	15.25	0.00
pmed35-pmed37	5-80	800	51.67	52.33	364.71	12.33	2.02
pmed38-pmed40	5-90	900	54.67	55.00	183.62	13.33	1.15
AVG					**124.55s**	**16.45**	**0.33%**

Rezultati poređenja sa algoritmom iz Poglavlja 3, dati u Tabelama 4.5 i C.5 (za $\alpha = 2$ i $t_{max} = 2n$ sekundi), potvrđuju efikasnost i primenljivost predloženog algoritma za rešavanje problema *p*-drugog centra. U odnosu na *p*-sledeći centar problem sa istim vrednostima parametara, algoritam je bio uspešniji u rešavanju problema *p*-drugog centra. U proseku, najbolje poznato rešenje pronađeno je u 16.45 od 20 izvršenja, tj. u 82.25% slučajeva. Prosečna odstupanja najgorih, srednjih i najboljih pronađenih rešenja u odnosu na najbolja rešenja algoritma iz Poglavlja 3 su relativno mala: 1.00%, 0.64% i 0.33%, respektivno. Sa druge strane, vreme potrebno da se pronađe najbolje rešenje ostaje veliko, 124.55 sekundi, u odnosu na 31.32 sekunde koliko je potrebno algoritmu za rešavanje problema *p*-drugog centra iz Poglavlja 3.

4.4. Zaključak

U ovom poglavlju razmatrani su problemi *p*-*α*-sledećih i *p*-*α*-najbližih centara, koji predstavljaju uopštenja problema *p*-sledećeg i *p*-drugog centra. Dizajniran je i implementiran metod promenljivih okolina kao metaheuristički pristup rešavanju problema. Rešenje problema *p*-*α*-sledećih centara predstavlja identifikaciju *p* od *n* potencijalnih centara sa ciljem minimizacije maksimalne dužine puta koji polazi od korisnika i obilazi *α* međusobno najbližih centara. Svaki naredni centar na putu je najbliži neposećeni centar prethodnog centra. Nasuprot tome, rešenje problema *p*-*α*-najbližih centara predstavlja identifikaciju *p* od *n* potencijalnih centara sa ciljem minimizacije maksimalne sume rastojanja od korisnika do *α* neposredno njemu najbližih centara.

Predložena je nova implementacija VNS metoda koja je testirana nad *OR-Library* instancama iz literature do 900 čvorova. Dobijeni eksperimentalni rezultati su pokazali da je algoritam, uz značajan utrošak procesorskog vremena, primenljiv za rešavanje konkretnih problema. Algoritam je pokazao stabilnost u rešavanju "najvećih" problema *p*-*α*-najbližih centara, kod kojih je potrebno identifikovati *p* ($\alpha = p$) najbližih centara. Iako je prosečan broj pronađenih najboljih rešenja u 20 izvršenja algoritma bio relativno mali 13.38, odstupanja vrednosti srednjih rešenja u odnosu na najbolja rešenja su bila samo 0.13%. Sa druge strane, ispostavilo se da su prihvatljiva samo najbolja rešenja algoritma za problem *p*-*α*-sledećih centara. Odstupanja najgorih i srednjih rešenja problema *p*-*α*-sledećih centara u odnosu na najbolja rešenja su čak 18.92% i 7.05% u proseku.

Predloženi algoritam je testiran i nad "najmanjim" problemima, kod kojih je potrebno minimizirati rastojanje samo do 2 najbliža centra ($\alpha = 2$). Ovako definisani problemi su poznati kao *p*-sledeći i *p*-drugi centar problem. Dobijeni rezultati primene nad ovim problemima su približno jednaki najboljim poznatim rezultatima. U poređenju sa trenutno najefikasnijim algoritmima, iz Poglavlja 2 i 3 ove disertacije, predloženi algoritam je bio prosečno lošiji samo za 0.47% u slučaju problema *p*-sledećeg i 0.33% u slučaju problema *p*-drugog centra. Sa druge strane, prosečna vremena pronalazaka najboljih rešenja su bila znatno veća, 253.96 sekundi za problem *p*-sledećeg i 124.55 sekundi za problem *p*-drugog centra. Algoritmi iz Poglavlja 2 i 3, oslanjajući se na teorijska razmatranja i specifičnosti samih problema, povećavaju kvalitet rešenja i brzinu konvergencije ka optimalnom rešenju konkretnog problema. Sa druge strane, novo-predloženi algoritam predstavlja generičko rešenje primenljivo na više različitih problema uz kompromis vremena potrebnog da se pronađe najbolje rešenje. Sa veličinom problema, raste i potrebno vreme pronalaska rešenja. Prosečno vreme pronalaska najboljih rešenja najvećih problema kod kojih je $\alpha = p$, bilo je čak 22439.83 sekunde za problem *p*-*α*-sledećih i 13040.57 sekundi za problem *p*-*α*-najbližih centara.

Predloženi algoritam uspešno rešava probleme *p*-*α*-sledećih i *p*-*α*-najbližih centara iz *OR-Library* test seta, ali uz značajan utrošak procesorskog vremena. Interesantno bi bilo primeniti algoritam i na veće instance problema od *OR-Library* test instanci, što bi zahtevalo značajno unapređenje vremena izvršenja.

5. Problemi *p*-sledećeg medijana i *p*-α-sledećeg medijana

Problem *p*-medijana je poznat i dugo proučavan problem koji podrazumeva identifikaciju *p* od potencijalnih *n* lokacija centara ($p \leq n$) na takav način da se minimizuje suma rastojanja između korisnika i najbližih centara. Za razliku od *p*-medijana, problem *p*-sledećeg medijana minimizuje sumu udaljenosti korisnika do najbližeg centra plus rastojanje između tog centra i njemu najbližeg centra. U disertaciji, predložen je heuristički algoritam za rešavanje problema *p*-sledećeg medijana kao nova implementacija poznatog generičkog okvira promenljivih okolina za izgradnju algoritama pretrage. Dalje, predstavljen je algoritam za minimizaciju suma rastojanja od korisnika do α najbližih centara, gde je $2 \leq \alpha \leq p$. Problem predstavlja uopštenje *p*-sledeći medijan problema i nazvan je *p*-α-sledeći medijan problem. Predloženi algoritmi su testirani nad u literaturi poznatim *OR-Library* skupom test instanci i rezultati pokazuju da u vremenskom intervalu srazmernom veličini problema vraćaju optimalna ili približno optimalna rešenja.

5.1. Uvod

U uvodnom poglavlju, naglašeno je da se disertacija bavi problemom otkaza primarno dodeljenih centara koji opslužuju korisnike u kontekstu različitih *p*-problema. Postoje dve klase ovakvih problema: prva kada se korisnik, u slučaju otkaza, odmah obraća zamenskom centru i druga kada najpre poseti primarni pa tek onda zamenski centar. U ovom poglavlju, razmatra se upravo drugi tip u kontekstu *p*-medijan problema, tj. problem *p*-sledećeg medijana.

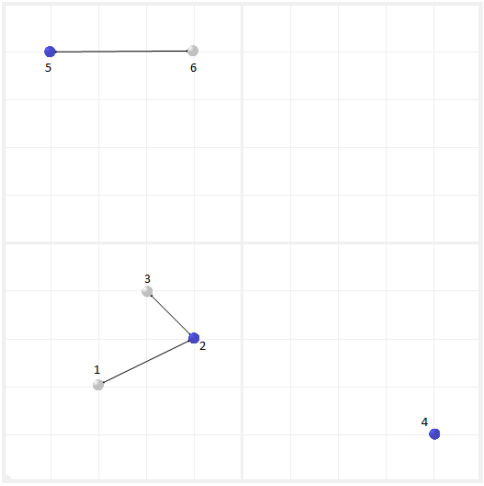
Problemom otkaza primarnog centra u kontekstu *p*-centar problema bavi se problem *p*-sledećeg centra (Poglavlje 2). Analogno problemu *p*-sledećeg centra, problem koji se bavi otkazom centra u slučaju *p*-medijan problema nazivan je *p*-sledeći medijan problem. Problem *p*-sledećeg medijana (*eng. p-next median problem*, pNMP) je proširenje NP-teškog [20] *p*-medijan problema (pMP) i predstavlja identifikaciju *p* od potencijalnih *n* lokacija centara u cilju minimizacije sume rastojanja korisnika do najbližeg centra plus rastojanja između tog centra i njemu najbližeg. Svakom korisniku u kontekstu *p*-sledeći medijan problema se unapred dodeljuje primarni i zamenski centar, nazvani kao i kod problema *p*-sledećeg centra, *reference* i *backup* centar. Formalno, pNMP se može definisati nad neusmerenim težinskim grafom $G = (V, E)$, gde su težine grana određene rastojanjem između svojih krajeva, V skup svih čvorova, a E skup grana grafa. Centri, kao i ostali korisnici predstavljaju čvorove grafa, a $d(i,j)$ je najkraće rastojanje između čvorova i i j (ukoliko u skupu E ne postoje grane koje posredno ili neposredno povezuju i i j čvorove, podrazumeva se da je $d(i,j) = \infty$). Tada je:

$$pNMP(V) = \min_{\substack{P \subset V \\ |P|=p}} \sum_{i \in V} \left\{ \min_{j \in P} d(i,j) + \min_{k \neq j' \in \left\{ \arg\min_{j \in P} d(i,j) \right\}} \min_{k \in P} d(j',k) \right\}. \tag{5.1}$$

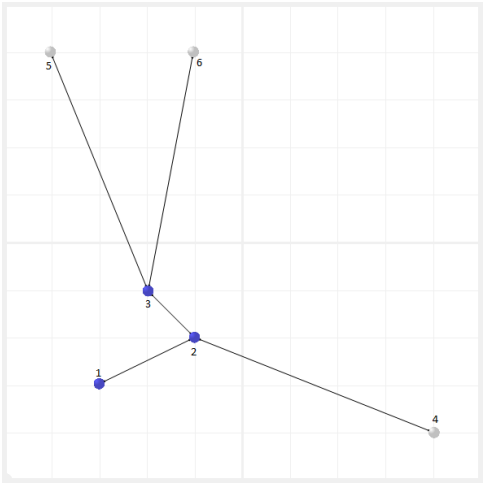
Da bi se ilustrovala razlika između pMP i pNMP, na Slikama 5.1 i 5.2 predstavljen je jednostavan primer rešenja problema pMP i pNMP. Definisan je problem za $n = 6$ korisnika i $p = 3$ centra. Optimalno rešenje za pMP je 6.65 jedinica i centri su locirani na pozicijama 2, 4 i 5. Vrednost funkcije cilja, 6.65 jedinica, izračunata je kao suma rastojanja svih korisnika do svojih *reference* centara, što je u ovom slučaju suma euklidskih rastojanje između čvorova 1 i 2, 2 i 2, 3 i 2, 4 i 4, 5 i 5 i još 6 i 5. Sa druge strane, vrednost optimalnog rešenja za pNMP je 25.17 jedinica i centri su postavljeni na pozicijama 1, 2 i 3. Vrednost funkcije, 25.17 jedinica, izračunata je kao suma rastojanja između svakog od korisnika i njegovog *reference* centra kao i između *reference* i *backup* centara. U konkretnom slučaju, to je suma dužina od čvora 1 do čvora 1 s obzirom da je to njegov *reference* centar plus dužina od čvora 1 do čvora 2, dalje dužina između čvorova 2 i 2 i između 2 i 3,

između čvorova 3 i 3 i između 3 i 2, između čvorova 4 i 2 i između 2 i 3, između čvorova 5 i 3 i između 3 i 2 i između čvorova 6 i 3 i između 3 i 2.

Kada su svi centri funkcionalni, korisnici dolaze do najbližeg centra (pMP). Na Slici 5.1, korisnici iz centara 2, 4 i 5 ostaju u ovim centrima s obzirom da su to *reference* centri, a korisnici iz 1 i 3 odlaze u centar 2, odnosno korisnik 6 odlazi u centar 5 s obzirom da je to najbliži centar. Kako god, može se ispostaviti da kada korisnik stigne do centra da tada taj centar ne bude funkcionalan. U tom slučaju, korisnici dolaze do najbližih *backup* centara (pNMP), kao što je predstavljeno na Slici 5.2. U primeru, korisnici iz centara 1, 2 i 3 ostaju u ovim centrima s obzirom da su to *reference* centri, ali u slučaju da neki od njih nije više funkcionalan, korisnici nastavljaju do sledećih najbližih centara, tj. *backup* centara. *Backup* centar centara 1 i 3 je centar 2, a centra 2 *backup* je centar 3. Iako su ovo veoma slični problemi, očigledno je da se optimalna rešenja za pMP i pNMP razlikuju. Primetno je da kod pNMP dolazi do grupisanja centara bliže jedan drugome, pa se algoritmi za rešavanje pNMP mogu koristiti i kao algoritmi klasterizacije čvorova grafa.



Sl. 5.1. Optimalno rešenje za *p*-medijan problem ($n = 6$ i $p = 3$)



Sl. 5.2. Optimalno rešenje za *p*-sledeći medijan problem ($n = 6$ i $p = 3$)

Problem *p*-sledećeg medijana je realan problem, međutim usled humanitarnih katastrofa širokih razmera očekivati je da dođe do otkaza i zamenskog centra. U tom slučaju, trebalo bi iskoristiti sledeći najbliži centar itd. Ukoliko α predstavlja unapred određen broj centara, *p*-sledeći medijan problem se može proširiti na *p*-α-sledeći medijan problem (*eng. p-α-next median problem*, pαNMP) kod koga je potrebno identifikovati p od mogućih n lokacija centara sa ciljem minimizacije sume rastojanja korisnika ne samo do 2 već do proizvoljnog broja α ($2 \leq \alpha \leq p$) centara. Formalno, nad istim neusmerenim težinskim grafom $G = (V, E)$, kao i u slučaju problema *p*-sledećeg medijana, *p*-α-sledeći medijan problem se definiše kao:

$$f(P,V)=\sum_{i\in V}\left\{\min_{j_1\in P} d(i,j_1)+\min_{\substack{k_1\in\left\{\arg\min_{j_1\in P} d(i,j_1)\right\}\\ j_2\in P\setminus\{k_1\}}} d(k_1,j_2)+\sum_{l=3}^{\alpha}\min_{\substack{k_1\in\left\{\arg\min_{j_1\in P} d(i,j_1)\right\}\\ k_2\in\left\{\arg\min_{j_2\in P\setminus\{k_1\}} d(k_1,j_2)\right\}\\ \vdots\\ k_{l-1}\in\left\{\arg\min_{j_{l-1}\in P\setminus\{k_1,k_2,\ldots,k_{l-2}\}} d(k_{l-2},j_{l-1})\right\}\\ j_l\in P\setminus\{k_1,k_2,\ldots,k_{l-1}\}}} d(k_{l-1},j_l)\right\},$$

$$p\alpha NMP(V)=\min_{\substack{P\subset V\\ |P|=p}} f(P,V). \tag{5.2}$$

U ovom radu, predstavljeni su algoritmi za rešavanje problema *p*-sledećeg medijana i *p*-*α*-sledećeg medijana. Cilj je pronaći optimalna ili skoro-optimalna rešenja većih instanci problema u što kraćem vremenskom intervalu. Poznato je da sa značajnim porastom veličine problema, egzaktni algoritmi postaju neupotrebljivi za rešavanje problema prvenstveno zbog velike vremenske a i memorijske zahtevnosti. Sve to ukazuje na potrebu za heurističkim algoritmima. U radu [15] je predstavljen heuristički algoritam za rešavanje *p*-medijan problema kao implementacija generičkog okvira promenljivih okolina za izgradnju algoritama pretrage. U Poglavlju 2 ove disertacije je dat do sada najefikasniji heuristički algoritam za rešavanje problema *p*-sledećeg centra, a u Poglavlju 4 algoritam za problem *p*-*α*-sledećih centara. Algoritmi predloženi u ovom poglavlju se oslanjaju na prethodno pomenute algoritme, tako da je implementacija preuzeta iz Poglavlja 2, Poglavlja 4 i rada [15], i prilagođena *p*-sledeći medijan i *p*-*α*-sledeći medijan problemima.

5.2. Algoritam

Predloženi algoritmi za rešavanje *p*-sledeći medijan i *p*-*α*-sledeći medijan problema su izgrađeni na metaheurističkom metodu promenljivih okolina (VNS), o kojem je bilo reči u uvodnom poglavlju disertacije. Zasnovani su na VNS algoritmima za *p*-sledeći centar (pNCP) i *p*-*α*-sledećih centara (pαNCP) probleme, predstavljenim u Poglavljima 2 i 4. Suštinska razlika u implementaciji novih algoritama u odnosu na pomenute pNCP i pαNCP algoritme je u strukturama za čuvanje međurezultata i implementaciji procedura za izračunavanje vrednosti funkcije cilja.

Pseudokod VNS implementacije za posmatrane probleme dat je u Algoritmu 5.1. Sličan je pseudokodu iz prethodnih poglavlja, a ovde je predstavljen zbog kompletnosti i objašnjenja pNMP i pαNMP rešenja. Pretraga počinje od N_1 okoline predefinisanog rešenja. Neka je rešenje koje se trenutno pretražuje nazvano tekućim rešenjem, a najbolje rešenje pronađeno tokom pretrage optimalnim rešenjem. U fazi lokalne pretrage, pretražuje se N_1 okolina tekućeg rešenja sa ciljem da se pronađe bolje rešenje od tekućeg, i u tom slučaju novo rešenje postaje optimalno rešenje.

VNS algoritmi za *p*-sledeći medijan i *p*-*α*-sledeći medijan probleme (Algoritam 5.1) pretragu potencijalnih rešenja počinju u N_1 okolini tekućeg rešenja, tako da inicijalna vrednost promenljive *k* jeste 1. Tekuće rešenje predstavlja prvih *p* centara niza x_{cur}. Optimalno rešenje se čuva u nizu x_{opt}. Nizovi imaju po *n* elemenata, tako da preostalih $n - p$ elemenata predstavlja samo korisnike i kandidate za nove centre, tj. centre koji bi u nekoj od narednih iteracija pretrage mogli biti uključeni u rešenje. U svakoj *While* iteraciji, najpre se metodom slučajnog uzorka bira rešenje iz N_k okoline tekućeg rešenja, tj. tačno *k* centara tekućeg rešenja se zamenjuje novim centrima. Izabrano rešenje postaje tekuće i kao takvo predstavlja ulaz u fazu lokalne pretrage, tj. *LocalSearchVertexSubstitution* proceduru. Za svakog od korisnika, održava se i niz sortiranih struktura centara koji služi za efikasno izračunavanje vrednosti funkcije cilja. Niz centara i trenutna vrednost funkcije cilja se takođe prosleđuju kao ulazne vrednosti *LocalSearchVertexSubstitution* proceduri. Ukoliko nakon lokalne

pretrage bude pronađeno rešenje bolje ili jednako trenutno optimalnom rešenju, novo rešenje postaje optimalno i pretraga se nastavlja u sledećoj iteraciji petlje u N_1 okolini novog rešenja ($k = 1$). U suprotnom, novo rešenje se odbacuje, a pretraga se nastavlja u N_{k+1} (tj. $k = k + 1$) okolini prethodnog rešenja. Ukoliko se prevaziđe maksimalna vrednost (k_{max}) za vrednost promenljive k, vrednost se resetuje na 1 ($k = 1$). Algoritam se izvršava sve dok ne istekne vremensko ograničenje t_{max}.

Algoritam 5.1: VNS algoritam za p-sledeći medijan i p-α-sledeći medijan probleme

VariableNeighborhoodSearch(k_{max}, t_{max})
Initialization:
Randomly initialize x_{opt}; according to x_{opt} initialize arrays of centers and f_{opt};
Copy initial solution into the current one, i.e., copy f_{opt}, x_{opt}, centers into f_{cur}, x_{cur}, $centers_{cur}$, respectively.

Repeat the Main step until the stopping condition is met (e.g., time ≤ t_{max})
Main step:
$k \leftarrow 1$
While $k \leq k_{max}$

Shaking operator:
*****generate a solution at random from kth neighborhood*****
For Each $j = 1, ..., k$
- *Take center to be inserted (c_{in}) at random*
- *Find center to be deleted (c_{out}) at random*
- *Update x_{cur} and $centers_{cur}$*
- *Update f_{cur} according to x_{cur} and $centers_{cur}$*

End For Each

Local search:
$x_{cur}, f_{cur} \leftarrow LocalSearchVertexSubstitution(x_{cur}, f_{cur}, centers_{cur})$
Move or not:
If $f_{cur} \leq f_{opt}$
*****Save current solution as the optimal*****
$x_{opt} \leftarrow x_{cur}$; $f_{opt} \leftarrow f_{cur}$; $centers \leftarrow centers_{cur}$
$k \leftarrow 1$
Else
*****Current solution is the optimal; change the neighborhood*****
$x_{cur} \leftarrow x_{opt}$; $f_{cur} \leftarrow f_{opt}$; $centers_{cur} \leftarrow centers$
$k \leftarrow k + 1$
End If

End While
Return x_{opt}, f_{opt}

U fazi lokalne pretrage, tekuće rešenje se iterativno unapređuje. U svakoj iteraciji, tekuće rešenje se zamenjuje optimalnim rešenjem iz N_1 okoline tekućeg rešenja, na taj način da se vrednost funkcije cilja maksimalno unapredi, tj. jedan od centara tekućeg rešenja se zamenjuje novim centrom sa ciljem minimizacije vrednosti funkcije cilja. Pretraga se nastavlja sve dok je moguće pronaći bar jedan par centara čijom razmenom se smanjuje vrednost funkcije tekućeg rešenja. Pseudokod *LocalSearchVertexSubstitution* procedure koja implementira fazu lokalne pretrage je dat u Algoritmu 5.2. Optimalan par centara za razmenu zavisi od tipa problema, p-sledeći medijan ili p-α-sledeći medijan, tako da se određuje pozivom *Move* procedure na osnovu specifične strukture centara. Nakon promene tekućeg rešenja, potrebno je ažurirati i strukture centara pozivom *Update* procedure. Svaki od problema ima svoju implementaciju *Move* i *Update* procedura kao i sortiranu strukturu centara tekućeg rešenja za svakog od korisnika.

Algoritam 5.2: Procedura zamene čvorova u fazi lokalne pretrage za probleme *p*-sledećeg i *p*-α-sledećeg medijana

LocalSearchVertexSubstitution(x_{cur}, f_{cur}, centers)
Main loop:
While *True*
    $f' \leftarrow \infty$
    $c_{in} \leftarrow null$; $c_{out} \leftarrow null$
    For Each *in = p +1, ..., n*
        Find the optimal objective function value improvement within N_1(xcur) neighborhood:
        f, out ← *Move(x_{cur}, in, centers)*
        If $f < f'$
            $f' \leftarrow f$
            $c_{in} \leftarrow in$
            $c_{out} \leftarrow out$
        End If
    End For Each
    If $f_{cur} \leq f'$
        There is not found improvement in the neighborhood:
        Break Main loop
    End If
    $x_{cur}(c_{in}) \leftrightarrow x_{cur}(c_{out})$
    Update(x_{cur}, centers)
    $f_{cur} \leftarrow f'$
End While
Return x_{cur}, f_{cur}

5.2.1. Algoritam za *p*-sledeći medijan problem

Slično algoritmu iz rada [15], algoritam za izračunavanje vrednosti funkcije cilja koristi nizove *c1* i *c2* koji evidentiraju najbliži i drugi-najbliži centar svakog od korisnika. Takođe, da bi se tokom faze lokalne pretrage pronašao optimalan centar za brisanje koristi se još i pomoćna struktura centara *z*, pri čemu element $z(p_i)$ sadrži promenu (bilo pozitivnu bilo negativnu) u vrednosti funkcije cilja ukoliko se u tekućem rešenju zatvori centar p_i.

U implementaciji rešenja *p*-sledeći medijan problema, kao pomoćne koriste se i sledeće funkcije:

- *nc1(v, in, c1, c2)* – vraća *reference* (najbliži) centar korisnika *v*. Ukoliko su centri *c1(v)* i *c2(v)* na jednakoj udaljenosti od korisnika *v*, prednost ima centar sa bližim *backup* centrom. Novi centar *in* takođe može biti *backup* centar,
- *nc2(v, in, c1, c2)* – vraća *backup* centar korisnika *v*,
- *nc1Dist(v, p, c1)* - vraća vrednost funkcije cilja za korisnika *v* i *reference* centar *p*,
- *nc2_1Dist(v, in, c1, c2)* – vraća novu vrednost funkcije cilja korisnika *v* nakon što se u tekućem rešenju njegov *reference* centar zameni novim centrom *in*,
- *nc2_2Dist(v, in, c1, c2)* - vraća novu vrednost funkcije cilja korisnika *v* nakon što se u tekućem rešenju njegov *backup* centar zameni novim centrom *in*,
- *dist(v, u)* – vraća dužinu najkraćeg puta između čvorova *v* i *u*.

Da bi se pojednostavio pseudokod, u algoritmima su izostavljeni *c1* i *c2* nizovi iz liste argumenata prilikom poziva prethodno pomenutih funkcija.

Algoritam 5.3: Zamena čvorova u N_1 okolini (*eng. 1-interchange move*) u kontekstu *p*-sledećeg medijana

Move(x_{cur}, c_{in}, c1, c2)
Initialization:
$f_{value} \leftarrow 0$; *Set* $z(x_{cur}(i)) \leftarrow 0$ *for all* $i = 1, ..., p$

Add center:
$in \leftarrow x_{cur}(c_{in})$
For Each $user = x_{cur}(1), \ldots, x_{cur}(n)$
*****calculate function value if center in is a new reference or backup center*****
f_{in}, *center* ← *GetFunctionValueForNewCentarIn(user, in, c1, c2) [Algorithm 5.3.1]*
*****calculate function value if center in is a new reference or backup center and "center" is deleted*****
$f_{out} \leftarrow$ *GetBackupFunctionValueForNewCentarIn(user, in, center, c1, c2) [Algorithm 5.3.2]*
*****calculate function value if center in is not a new reference center*****
$f_{cur} \leftarrow$ *GetFunctionValue(user, in, c1, c2) [Algorithm 5.3.3]*

Update f_{value} and z values:
If $f_{in} \leq f_{cur}$
******center in is either a new reference or backup center*******
$f_{value} \leftarrow f_{value} + f_{in}$
*****calculate function value if center in is not a new reference center and "center" is deleted*****
$f'_{out} \leftarrow$ *GetBackupFunctionValue(user, in, center, c1, c2) [Algorithm 5.3.4]*
*****in case that "center" is deleted*****
$z(center) \leftarrow z(center) + \min(f_{out}, f'_{out}) - f_{in}$
Else
*****nc1(user, in) is the reference center; c1(nc1(user, in)) is the backup center*****
$f_{value} \leftarrow f_{value} + f_{cur}$
*****in case that the reference center is deleted*****
$f'_{out} \leftarrow$ *GetBackupFunctionValue(user, in, nc1(user, in), c1, c2) [Algorithm 5.3.4]*
$f \leftarrow f_{in}$ ***if*** $nc1(user, in) \neq center$ ***else*** f_{out}
$z(nc1(user, in)) \leftarrow z(nc1(user, in)) + \min(f, f'_{out}) - f_{cur}$
*****in case that the backup center is deleted*****
$f''_{out} \leftarrow$ *GetBackupFunctionValue(user, in, c1(nc1(user, in)), c1, c2) [Algorithm 5.3.4]*
$f \leftarrow f_{in}$ ***if*** $c1(nc1(user, in)) \neq center$ ***else*** f_{out}
$z(c1(nc1(user, in))) \leftarrow z(c1(nc1(user, in))) + \min(f, f''_{out}) - f_{cur}$
End If
End For Each

Best deletion:
$min \leftarrow \infty$
For Each $i = \{1, ..., p\}$
If $min > z(x_{cur}(i))$
$min \leftarrow z(x_{cur}(i))$
$c_{out} \leftarrow i$
End If
End For Each

$f_{value} \leftarrow f_{value} + min$

Return f_{value}, c_{out}

Algoritam 5.3.1: Izračunavanje vrednosti pNMP funkcije korisnika *u* sa novim *reference* ili *backup* centrom *in* ("*counterpart center*" je novi *backup* ukoliko je *in* novi *reference* centar i obrnuto)

GetFunctionValueForNewCentarIn(u, in, c1, c2)
*****in as the new reference center*****
f ← *nc1Dist(u, in)* **if** *dist(u, in)* ≤ *dist(u, nc1(u, in))* **else** ∞
counterpart_center ← *c1(in)*

*****in as the new backup center*****
f' ← *dist(u, nc1(u, in))* + *dist(nc1(u, in), in)* **if** *dist(u, nc1(u, in))* ≤ *dist(u, in)* **else** ∞
If *f* > *f'*
    f ← *f'*
    counterpart_center ← *nc1(u, in)*
End If

Return *f, counterpart_center*

Algoritam 5.3.2: Izračunavanje vrednosti pNMP funkcije korisnika *u* nakon zatvaranja centra *out* i dodele novog *reference* ili *backup* centra *in*

GetBackupFunctionValueForNewCentarIn(u, in, out, c1, c2)
*****"center" is the closest center (not including in), after center out has been deleted*****
center ← *nc1(u, in)* **if** *nc1(u, in)* ≠ *out* **else** *nc2*(u, in)

*****in as the new reference center*****
If *c1(in)* ≠ *out*
    f' ← *nc1Dist(u, in)* **if** *dist(u, in)* ≤ *dist(u, center)* **else** ∞
Else
    f' ← *dist(u, in)* + *dist(in, c2(in))* **if** *dist(u, in)* ≤ *dist(u, center)* **else** ∞
End If

*****in as the new backup center*****
f'' ← *dist(u, center)* + *dist(center, in)* **if** *dist(u, center)* ≤ *dist(u, in)* **else** ∞

Return *min(f', f'')*

Algoritam 5.3.3: Izračunavanje vrednosti pNMP funkcije korisnika *u* sa potencijalno novim *backup* centrom *in* (ukoliko je *in* novi *reference* centar, kao rezultat se vraća ∞)

GetFunctionValue(u, in, c1, c2)
f ← *nc1Dist(u, nc1(u, in))* **if** *dist(u, nc1(u, in))* ≤ *dist(u, in)* **else** ∞
Return *f*

Algoritam 5.3.4: Izračunavanje vrednosti pNMP funkcije korisnika *u* nakon zatvaranja centra *out* (novi centar *in* može ostati nedodeljen ili postati *backup* centar, a ako je *in* ipak novi *reference* centar vrednost funkcije je ∞)

```
GetBackupFunctionValue(u, in, out, c1, c2)
If out ≠ nc1(u, in) and out ≠ c1(nc1(u, in))
        **neither the reference nor the backup center is deleted**
        f ← GetFunctionValue(u, in, c1, c2) [Algorithm 5.3.3]
Else
        If out = nc1(u, in)
                **the reference center is deleted**
                f ← nc2_1Dist(u, in) if dist(u, nc2(u, in)) ≤ dist(u, in) else ∞
        Else
                **the backup center is deleted**
                f ←nc2_2Dist(u, in) if dist(u, nc1(u, in)) ≤ dist(u, in) else ∞
        End If
End If
Return f
```

Algoritam 5.3 sadrži pseudokod *Move* procedure u kontekstu *p*-sledeći medijan problema. Kako koraci algoritma nisu tako očigledni, sledi kratko objašnjenje. *Move* procedura identifikuje optimalan centar u tekućem rešenju koji bi trebalo zameniti novim centrom *in*, ali tako da vrednost funkcije cilja bude što manja. Nova vrednost funkcije cilja se računa kao suma f_{value} vrednosti i minimalne vrednosti *z* niza. Promenljiva f_{value} sadrži vrednost funkcije cilja u tekućem rešenju proširenom centrom $in = x_{cur}(c_{in})$, dok $z(p_i)$ predstavlja pozitivan ili negativan porast vrednosti funkcije u slučaju da se iz ovog rešenja isključi centar p_i.

Nakon inicijalizacije f_{value} i *z* vrednosti, za svakog od korisnika, označenog sa *user*, najpre se proverava da li otvaranjem novog centra *in*, ovaj centar postaje novi *reference* ili možda *backup* centar korisnika *user*. U tom slučaju, izračunava se f_{in} vrednost funkcije koja odgovara *user*-u i određuje nedostajući dodeljeni centar, tj. *reference* centar korisnika *user* ukoliko je *in* novi *backup* centar i obrnuto. Dodeljeni centar koji je različit od *in* se označava sa *center*. Osnovna ideja iza ovog koraka algoritma je da se istovremeno sa izračunavanjem vrednosti funkcije cilja tekućeg rešenja (proširenog centrom *in*), bez uvećanja kompleksnosti, odredi i vrednost funkcije u slučaju da se neki od korisniku dodeljenih centara zatvori, i na taj način ubrza konvergencija ka lokalnom optimumu. Činjenica je da novi centar *in* neće biti zatvoren u istoj iteraciji, tako da je potrebno odrediti još samo vrednost funkcije koja odgovara *user*-u u slučaju zatvaranja dodeljenog centra *center*. Nakon zatvaranja ovog centra, *in* ponovo može biti dodeljen korisniku *user* kao *reference* ili *backup* centar, i u tom slučaju nova vrednost funkcije cilja koja odgovara *user*-u je f_{out}. Takođe, može se desiti da *user* za svoj *reference* centar povrati jedan od prethodno dodeljenih centara iz tekućeg rešenja, što odgovara vrednosti f'_{out}. Svakako, teži se minimalnoj vrednosti, tako da će nova vrednost funkcije biti $min(f_{out},f'_{out})$.

U slučaju da *in* nije novi *reference* niti *backup* centar korisnika *user*, f_{in} dobija vrednost ∞. Sa druge strane, ukoliko *user* zadrži svoj *reference* centar i nakon otvaranja centra *in*, f_{cur} predstavlja vrednost funkcije koja odgovara korisniku *user* u novom rešenju koje uključuje i centar *in*. U suprotnom, f_{cur} dobija vrednost ∞.

Upoređivanjem f_{in} i f_{cur} vrednosti, zaključuje se da li je centar *in* dodeljen korisniku *user* u novom rešenju. Ukoliko važi da je $f_{in} \leq f_{cur}$, *in* je novi *reference* ili *backup* centar korisnika *user*. To znači da korisnik *user* u ukupnoj vrednosti funkcije cilja "učestvuje" sa f_{in} ($f_{value} = f_{value} + f_{in}$) i da je dovoljno ažurirati *z* vrednost koja odgovara promeni vrednosti funkcije cilja u slučaju zatvaranja korisniku dodeljenog centra različitog od *in*. Promena vrednosti funkcije se računa kao $min(f_{out},f'_{out}) - f_{in}$, i odgovarajući *z* element se uvećava upravo za ovu vrednost. U suprotnom, kada je $f_{cur} < f_{in}$, *user* zadržava svoje centre, ukupnoj vrednosti funkcije cilja "doprinosi" sa f_{cur} i potrebno je ažurirati *z* vrednosti koje odgovaraju potencijalnom zatvaranju i *reference* i *backup* centra.

Konačno, optimalan centar za zatvaranje odgovara minimalnom elementu niza *z*, tj. minimalnom (ili maksimalno negativnom) porastu vrednosti funkcije cilja tekućeg rešenja proširenog centrom *in*. Nova vrednost funkcije cilja se računa kao f_{value} plus pomenuti porast vrednosti funkcije.

Algoritam 5.4: Ažuriranje nizova najbližih i drugih-najbližih centara svih korisnika

Update(x_{cur}, c1, c2)
For each user, properly update the closest and the second-closest center:
For Each *user = $x_{cur}(1), \ldots, x_{cur}(n)$*
    c1_dist ← ∞; c1_center ← null
    c2_dist ← ∞; c2_center ← null
    For Each *center = $x_{cur}(1), \ldots, x_{cur}(p)$*
        d ← dist(user, center)
        If *d < c1_dist or d = c1_dist and dist(center, c1(center)) < dist(c1_center, c1(c1_center))*
            c2_center ← c1_center
            c2_dist ← c1_dist
            c1_center ← center
            c1_dist ← d
        Else If *d < c2_dist or d = c2_dist and*
            dist(center, c1(center)) < dist(c2_center, c1(c2_center))
            c2_center ← center
            c2_dist ← d
        End If
    End For Each
    c1(user) ← c1_center
    c2(user) ← c2_center
End For Each
Return *c1, c2*

Nakon svake zamene centra tekućeg rešenja novim centrom, potrebno je i ažurirati nizove najbližih centara *c1* i *c2*. Pseudokod *Update* procedure je predstavljen u Algoritmu 5.4.

Svojstvo 5.1. Vremenska složenost *Move* algoritma (Algoritam 5.3) je $O(n+p)$.

Dokaz. Neka je:

- $P = \{p_1, p_2, \ldots, p_p\}$ – tekuće rešenje *p*-sledeći medijan problema,
- $V = \{v_1, v_2, \ldots, v_n\}$ – skup svih korisnika,
- $P' = P \cup \{c_{in}\}$, $c_{in} \in V \backslash P$ – rešenje koje se dobija dodavanjem centra c_{in} tekućem rešenju,
- $P_i = P' \backslash \{p_i\} = P \cup \{c_{in}\} \backslash \{p_i\}$, $p_i \in P$ – rešenje koje se dobija brisanjem centra p_i iz rešenja P',
- $f(X)$ – vrednost funkcije cilja u rešenju $X \in \{P', P_i\}$,
- $z(p_i)$, $p_i \in P$ – promena (bilo pozitivna bilo negativna) vrednosti funkcije cilja u rešenju P' nakon isključenja centra p_i,
- $rc_X(v_j)$, $j = 1, \ldots, n$ – *reference* centar korisnika v_j u rešenju $X \in \{P', P_i\}$,
- $c1_X(v_j)$, $j = 1, \ldots, n$ – najbliži centar korisniku/centru v_j u rešenju $X \in \{P, P', P_i\}$,
- $c2_X(v_j)$, $j = 1, \ldots, n$ – drugi-najbliži centar korisniku/centru v_j u rešenju $X \in \{P, P'\}$,
- $d(u, v)$ – najkraće rastojanje između čvorova u i v.

Posmatrajmo najpre skup centara P', tj. slučaj kada je novi centar c_{in} uključen u tekuće rešenje P, a da pri tom nije zatvoren niti jedan centar. Za svako $v_i \in V$, u rešenju P' važi:

$$c1_{P'}(v_i) = \begin{cases} c_{in}, \ if\ cndCP_1(v_i) \vee \left(cndCP_2(v_i) \wedge cndCP_3(v_i)\right) \\ c1_P(v_i), \ if\ cndCP_4(v_i) \vee \left(cndCP_5(v_i) \wedge cndCP_6(v_i)\right) \\ c2_P(v_i), \ otherwise \end{cases}, \tag{5.3}$$

gde je:

$$cndCP_1(v_i) = d(v_i, c_{in}) < d\left(v_i, c1_P(v_i)\right),$$

$$cndCP_2(v_i) = \left[d(v_i, c_{in}) = d\left(v_i, c1_P(v_i)\right)\right] \wedge d\left(c_{in}, c1_P(c_{in})\right) < min\left\{d\left(c1_P(v_i), c1_P\left(c1_P(v_i)\right)\right), d\left(c1_P(v_i), c_{in}\right)\right\},$$

$$cndCP_3(v_i) = d(v_i, c_{in}) < d\left(v_i, c2_P(v_i)\right) \vee d\left(c_{in}, c1_P(c_{in})\right) < min\left\{d\left(c2_P(v_i), c1_P\left(c2_P(v_i)\right)\right), \left(c2_P(v_i), c_{in}\right)\right\},$$

$$cndCP_4(v_i) = d\left(v_i, c1_P(v_i)\right) < d\left(v_i, c2_P(v_i)\right),$$

$$cndCP_5(v_i) = \left[d\left(v_i, c1_P(v_i)\right) = d\left(v_i, c2_P(v_i)\right)\right],$$

$$cndCP_6(v_i) = min\left\{d\left(c1_P(v_i), c1_P(c1_P(v_i))\right), d(c1_P(v_i), c_{in})\right\} \leq min\left\{d\left(c2_P(v_i), c1_P(c2_P(v_i))\right), (c2_P(v_i), c_{in})\right\}.$$

Tada je:

$$rc_{P'}(v_i) = \begin{cases} c1_{P'}(v_i), v_i \notin P' \\ v_i, v_i \in P' \end{cases}. \quad (5.4)$$

Pretpostavimo da je:

1) $c1_{P'}(v_i) = c_{in}$, tada je:

$$c2'_{P'}(v_i) = \begin{cases} c1_P(v_i),\ cndCP'_1(v_i) \vee \left(cndCP'_2(v_i) \wedge cndCP'_3(v_i)\right) \\ c2_P(v_i),\ otherwise \end{cases}, \quad (5.5)$$

gde je:

$$cndCP'_1(v_i) = d(v_i, c1_P(v_i)) < d(v_i, c2_P(v_i)),$$
$$cndCP'_2(v_i) = \left[d(v_i, c1_P(v_i)) = d(v_i, c2_P(v_i))\right],$$
$$cndCP'_3(v_i) = min\left\{d\left(c1_P(v_i), c1_P(c1_P(v_i))\right), d(c1_P(v_i), c_{in})\right\} \leq min\left\{d\left(c2_P(v_i), c1_P(c2_P(v_i))\right), (c2_P(v_i), c_{in})\right\}.$$

2) $c1_{P'}(v_i) = c1_P(v_i)$:

$$c2''_{P'}(v_i) = \begin{cases} c_{in},\ cndCP''_1(v_i) \vee cndCP''_2(v_i) \\ c2_P(v_i),\ otherwise \end{cases}, \quad (5.6)$$

gde je:

$$cndCP''_1(v_i) = d(v_i, c_{in}) < d(v_i, c2_P(v_i)),$$
$$cndCP''_2(v_i) = \left[d(v_i, c_{in}) = d(v_i, c2_P(v_i))\right] \wedge d(c_{in}, c1_P(c_{in})) < min\left\{d\left(c2_P(v_i), c1_P(c2_P(v_i))\right), (c2_P(v_i), c_{in})\right\}.$$

3) $c1_{P'}(v_i) = c2_P(v_i)$:

$$c2'''_{P'}(v_i) = \begin{cases} c_{in},\ cndCP'''_1(v_i) \wedge cndCP'''_2(v_i) \\ c1_P(v_i),\ otherwise \end{cases}, \quad (5.7)$$

gde je:

$$cndCP'''_1(v_i) = \left[d(v_i, c_{in}) = d(v_i, c1_P(v_i))\right],$$
$$cndCP'''_2(v_i) = d(c_{in}, c1_P(c_{in})) < min\left\{d\left(c1_P(v_i), c1_P(c1_P(v_i))\right), (c1_P(v_i), c_{in})\right\}.$$

Iz (5.5), (5.6) i (5.7) sledi:

$$c2_{P'}(v_i) = \begin{cases} c2'_{P'}(v_i), c1_{P'}(v_i) = c_{in} \\ c2''_{P'}(v_i), c1_{P'}(v_i) = c1_P(v_i). \\ c2'''_{P'}(v_i), c1_{P'}(v_i) = c2_P(v_i) \end{cases} \quad (5.8)$$

Na osnovu prethodnih izraza sledi:

$$f(P') = \sum_{v_i \in V}\left\{d(v_i, rc_{P'}(v_i)) + d\left(rc_{P'}(v_i), c1_{P'}(rc_{P'}(v_i))\right)\right\}. \quad (5.9)$$

Posmatrajmo sada skup P_i, tj. rešenje koje se dobija kada se iz skupa P' isključi centar p_i. Upoređujući rešenja P' i P_i, skup korisnika V se deli na dva disjunktna podskupa:

- V'_i, korisnici koji u rešenju P_i zadržavaju svoj par centara iz rešenja P':

$$z'(p_i) = 0, \quad (5.10)$$

- V''_i, korisnici koji nakon zatvaranja centra p_i gube svoj *reference* i/ili *backup* centar iz rešenja P':

$$z''(p_i) = \sum_{v_j \in V''_i}\left\{d\left(v_j, rc_{P_i}(v_i)\right) + d\left(rc_{P_i}(v_i), c1_{P_i}\left(rc_{P_i}(v_i)\right)\right) - d\left(v_j, rc_{P'}(v_i)\right) - d\left(rc_{P'}(v_i), c1_{P'}(rc_{P'}(v_i))\right)\right\}. \quad (5.11)$$

Iz (5.10) i (5.11) sledi:

$$z(p_i) = z''(p_i),$$

tj.

$$z(p_i) = \sum_{v_j \in V''_i}\left\{d\left(v_j, rc_{P_i}(v_i)\right) + d\left(rc_{P_i}(v_i), c1_{P_i}\left(rc_{P_i}(v_i)\right)\right) - d\left(v_j, rc_{P'}(v_i)\right) - d\left(rc_{P'}(v_i), c1_{P'}(rc_{P'}(v_i))\right)\right\}. \quad (5.12)$$

Takođe, za svako $v_i \in V$, u rešenju P_i važi:

$$c1_{P_i}(v_i) = \begin{cases} c1_{P'}(v_i),\ cndCP_1^{iv}(v_i) \vee \left(cndCP_2^{iv}(v_i) \wedge \left(cndCP_3^{iv}(v_i) \vee cndCP_4^{iv}(v_i)\right)\right) \\ c2_{P'}(v_i),\ otherwise \end{cases}, \quad (5.13)$$

gde je:
$cndCP_1^{iv}(v_i) = [c2_{P'}(v_i) = p_i]$,
$cndCP_2^{iv}(v_i) = c1_{P'}(v_i) \neq p_i$,
$cndCP_3^{iv}(v_i) = d(v_i, c1_{P'}(v_i)) < d(v_i, c2_{P'}(v_i))$,
$cndCP_4^{iv}(v_i) = d\left(c1_{P'}(v_i), bc_{P_i}(c1_{P'}(v_i))\right) < d\left(c2_{P'}(v_i), bc_{P_i}(c2_{P'}(v_i))\right)$, $bc_{P_i}(c) = \begin{cases} c1_{P'}(c), c1_{P'}(c) \neq p_i \\ c2_{P'}(c), otherwise \end{cases}$, $c \in \{c1_{P'}(v_i), c2_{P'}(v_i)\}$.

Tada je:

$$rc_{P_i}(v_i) = \begin{cases} c1_{P_i}(v_i), v_i \notin P_i \\ v_i, v_i \in P_i \end{cases}. \quad (5.14)$$

Na osnovu (5.9) i (5.12), vrednost funkcije cilja u rešenju P_i izračunavamo kao:

$$f(P_i) = f(P') + z(p_i). \quad (5.15)$$

Konačno, *Best deletion* predstavljamo kao što sledi:

$$f(P^{(best)}) = \min_{i=1,\dots,p} f(P_i) = \min_{i=1,\dots,p} \{f(P') + z(p_i)\} = f(P') + \min_{i=1,\dots,p} z(p_i). \quad (5.16)$$

Poslednji izraz opisuje proceduru predstavljenu *Best deletion* korakom u Algoritmu 5.3.
Iz (5.3), (5.4), (5.13) i (5.14) sledi da se $c1_X(v_i)$ i $rc_X(v_i)$ vrednosti, gde $X \in \{P', P_i\}$, pronalaze u konstantnoj vremenskoj složenosti. Na osnovu toga, jednostavno je primetiti da izračunavanja $f(P')$ i z vrednosti u (5.9) i (5.12) imaju $O(n)$, tj. linearnu vremensku kompleksnost u odnosu na broj korisnika (*Add center* korak u Algoritmu 5.3). Dalje, iz (5.16), *Best deletion* korak se izvršava za $O(p)$, pa prema tome ukupna vremenska složenost *Move* procedure je $O(n + p)$. ■

Svojstvo 5.2. Pomoću algoritma *Move*, *p* različitih rešenja iz okoline $N_1(P)$ je posećeno.

Dokaz. U *Best deletion* koraku, *z* struktura sadrži *p* vrednosti funkcije cilja. Te vrednosti odgovaraju rešenjima gde je novi centar c_{in} otvoren, a svaki od *p* centara tekućeg rešenja *P* pojedinačno zatvoren. ■

Svojstvo 5.3. Vremenska složenost jedne iteracije *LocalSearchVertexSubstitution* algoritma (Algoritam 5.2) u slučaju *p*-sledeći medijan problema je $O(n^2)$.

Dokaz. Da bi se odredila vremenska složenost iteracije *LocalSearchVertexSubstitution* algoritma, najpre je potrebno utvrditi složenost ostalih procedura. Na osnovu Svojstva 5.1, vremenska složenost *Move* procedure je $O(n + p)$. *Update* procedura (Algoritam 5.4) ažurira nizove *c1* i *c2* dužine *p* za svakog od *n* korisnika tako da vremenska složenost iznosi $O(pn)$. *LocalSearchVertexSubstitution* algoritam koristi *Move* proceduru $n - p$ puta i jednom *Update* proceduru, pa prema tome ukupna vremenska složenost je $O(n^2 - p^2) + O(pn) \approx O(n^2)$. ■

5.2.2. Algoritam za *p*-*α*-sledeći medijan problem

Za razliku od algoritma za *p*-sledeći medijan koji koristi strukturu samo najbližeg i drugog-najbližeg centra, u implementaciji algoritma za rešavanje *p*-*α*-sledeći medijan problema, slično kao u algoritmu iz Poglavlja 4, za svakog korisnika održava se samo-balansirajuće binarno stablo pretrage svih centara tekućeg rešenja. Centri su sortirani počev od korisniku najbližeg centra, što u velikoj meri pojednostavljuje implementaciju *Move* i *Update* procedura. Strukture centara i implementacije ovih procedura su identične kao kod algoritma za probleme *p*-*α*-sledećih i *p*-*α*-najbližih centara iz Poglavlja 4. Ovde su ponovljene da bi se predstavilo kompletno rešenje *p*-*α*-sledeći medijan problema. Razlika je samo u algoritmu za izračunavanje vrednosti funkcije cilja. U implementaciji, kao pomoćne strukture i funkcije, koriste se:

- *dist(u, v)* – najkraće rastojanje između čvorova *u* i *v*,
- *centers(v)* – binarno stablo pretrage centara tekućeg rešenja korisnika *v*,
- *αNM(α, v, in, out, centers)* – vrednost funkcije cilja korisnika *v* nakon što se centar *out* tekućeg rešenja zameni novim centrom *in*.

Algoritam 5.5: Zamena čvorova u N_1 okolini tekućeg rešenja u kontekstu p-α-sledeći medijan problema

Move(α, x_{cur}, c_{in}, centers)
Initialization:
$f \leftarrow \infty$; $u_c \leftarrow null$; $in \leftarrow x_{cur}(c_{in})$; $c_{out} \leftarrow null$
Best deletion:
For Each $i = \{1, ..., p\}$
    $out \leftarrow x_{cur}(i)$
    $value \leftarrow 0$
    For Each $user = x_{cur}(1), \ldots, x_{cur}(n)$
        $value \leftarrow value + \alpha NM(\alpha, user, in, out, centers)$
    End For Each
    If $f > value$
        $f \leftarrow value$
        $c_{out} \leftarrow i$
    End If
End For Each
Return f, c_{out}

Move procedura (Algoritam 5.5) identifikuje centar tekuće pretraživanog rešenja koji bi trebalo zameniti novim centrom u cilju maksimalnog unapređenja vrednosti funkcije cilja i ujedno vraća vrednost funkcije cilja novog rešenja.

Algoritam 5.6: Ažuriranje binarnih stabala pretrage koja sadrže centre tekućeg rešenja

Update(x_{cur}, c_{in}, c_{out}, centers)
$in \leftarrow x_{cur}(c_{in})$
$out \leftarrow x_{cur}(c_{out})$
For Each $user = x_{cur}(1), \ldots, x_{cur}(n)$
    $erase(out, centers(user))$
    $insert(in, centers(user))$
End For Each
Return $centers$

Update procedura (Algoritam 5.6), za sve korisnike ažurira samo-balansirajuća binarna stabla pretrage koja sadrže centre tekućeg rešenja. Kako svaki od korisnika ima svoje stablo, održava se niz tih stabala. U svako od balansiranih stabala se unosi novi centar c_{in} i briše centar c_{out}.

Algoritam 5.7: Izračunavanje vrednosti funkcije cilja p-α-sledeći medijan problema

αNM (α, user, in, out, centers)
$u \leftarrow user$; $value \leftarrow 0$; $P \leftarrow \{\}$
While $|P| < \alpha$
    $c \leftarrow$ ***closest*** $center$ ***from*** $centers(u) \cup \{in\} \setminus \{out\}$ ***where*** $center \notin P$
    $value \leftarrow value + dist(u, c)$
    $P \leftarrow P \cup \{c\}$
    $u \leftarrow c$
End While
Return $value$

Konačno, Algoritam 5.7 prikazuje *αNM* proceduru za izračunavanje vrednosti funkcije cilja na osnovu održavanih binarnih stabala pretrage. Stablo sadrži centre tekućeg rešenja sortirane na osnovu udaljenosti od korisnika, počev od najbližeg centra, tako da se za izračunavanje vrednosti funkcije cilja uzima u obzir prvih α različitih centara.

Svojstvo 5.4. Vremenska složenost *Move* algoritma (Algoritam 5.5) u najgorem slučaju je $O(\alpha^2 pn)$.

Dokaz. Neka je:

- $P = \{p_1, p_2, \dots, p_p\}$ – tekuće rešenje *p*-*α*-sledeći medijan problema,
- $V = \{v_1, v_2, \dots, v_n\}$ – skup svih korisnika,
- $P_i = P \cup \{c_{in}\} \backslash \{p_i\}$, $c_{in} \in V \backslash P$, $p_i \in P$ – rešenje koje se dobija uključivanjem novog centra c_{in} u tekuće rešenje P i zatvaranjem centra p_i,
- $f(P_i)$, $i = 1, \dots, p$ – vrednost funkcije cilja u rešenju P_i,
- $f_{P_i}(v_j)$, $v_j \in V$, $j = 1, \dots, n$ – vrednost funkcije cilja u rešenju P_i za korisnika v_j,
- $d(u, v)$ – najkraće rastojanje između čvorova u i v.

Korisniku v_j u rešenju P_i su dodeljeni sledeći centri:

$$c_{P_i}^{(1)}(v_j) = \underset{c_k \in P_i}{arg\,min} \{d(v_j, c_k)\}, \tag{5.17}$$

$$c_{P_i}^{(l)}(v_j) = \underset{c_k \in P_i \backslash \left\{c_{P_i}^{(1)}(v_j), c_{P_i}^{(2)}(v_j), \dots, c_{P_i}^{(l-1)}(v_j)\right\}}{arg\,min} \left\{d(c_{P_i}^{(l-1)}(v_j), c_k)\right\}, l = 2, \dots, \alpha. \tag{5.18}$$

Tada je:

$$f_{P_i}(v_j) = d\left(v_j, c_{P_i}^{(1)}(v_j)\right) + \sum_{l=2}^{\alpha} d\left(c_{P_i}^{(l-1)}(v_j), c_{P_i}^{(l)}(v_j)\right), \tag{5.19}$$

tj.

$$f(P_i) = \sum_{j=1}^{n} f_{P_i}(v_j). \tag{5.20}$$

Na osnovu prethodnog, *Best deletion* određujemo kao:

$$f\left(P^{(best)}\right) = \min_{i=1,\dots,p} f(P_i) = \min_{i=1,\dots,p} \left\{\sum_{j=1}^{n} f_{P_i}(v_j)\right\}. \tag{5.21}$$

Poslednji izraz opisuje proceduru predstavljenu *Best deletion* korakom u Algoritmu 5.5.
Ukoliko se svakom korisniku dodeli sortirana struktura centara iz rešenja P_i, iz (5.17), (5.18) i (5.19), vremenska složenost izračunavanja vrednosti $f_{P_i}(v_j)$ je $O(1 + 2 + \dots + \alpha) = O\left(\frac{\alpha*(\alpha+1)}{2}\right) \approx O(\alpha^2)$ u najgorem slučaju (i $O(\alpha)$ u najboljem slučaju). Na osnovu prethodnog, jednostavno je primetiti da se $f(P_i)$ u najgorem slučaju izračunava za $O(\alpha^2 n)$ vremensku kompleksnost (odnosno za $O(\alpha n)$ u najboljem slučaju). Dalje, iz (5.21), *Best deletion* korak se izvršava za $O(p\alpha^2 n)$, pa je prema tome vremenska složenost *Move* procedure $O(\alpha^2 pn)$ u najgorem slučaju. ■

Svojstvo 5.5. Vremenska složenost jedne iteracije *LocalSearchVertexSubstitution* algoritma (Algoritam 5.2) u najgorem slučaju za *p*-*α*-sledeći medijan problem je $O(\alpha^2 pn^2)$.

Dokaz. Da bi se odredila vremenska složenost iteracije *LocalSearchVertexSubstitution* algoritma, najpre je potrebno utvrditi složenost ostalih procedura. Na osnovu Svojstva 5.4, vremenska složenost *Move* procedure je $O(\alpha^2 pn)$ u najgorem slučaju. *Update* procedura (Algoritam 5.6) u sortiranu strukturu za svakog od *n* korisnika dodaje i briše po jedan centar, tako da vremenska složenost iznosi $O(n \log p)$. *LocalSearchVertexSubstitution* algoritam koristi *Move* proceduru *n* – *p* puta i jednom *Update* proceduru, pa prema tome vremenska složenost u najgorem slučaju je $O(\alpha^2 pn^2) + O(n \log p) \approx O(\alpha^2 pn^2)$. ■

5.3. Rezultati testiranja

Algoritmi su implementirani u programskom jeziku C++, a sva testiranja izvršena na *Intel Core i7-8700K (3.7GHz) CPU* sa *32GB RAM*-a konfiguraciji. Za potrebe testiranja, preuzet je *OR-Library* [2] skup podataka, inicijalno namenjen za testiranje rešenja *p*-medijan problema. Biblioteka sadrži 40 test instanci sa 100 do 900 čvorova i *p* između 5 i 200 (ne više od *n/3*, gde je *n* broj čvorova).

Predloženi algoritmi su izvršeni po 20 puta nad svakom *OR-Library* instancom, uvek počevši od različitog inicijalnog rešenja. Isprobane su različite kombinacije parametara $k_{max} = p/2$, $k_{max} = p$, kao i $t_{max} = n$, $t_{max} = 2n$, $t_{max} = 5n$ i $t_{max} = 10n$. Ispostavilo se da su rezultati neznatno bolji sa većim vrednostima parametra k_{max}. Sa druge strane, algoritam je za manje instance problema pronalazio najbolja rešenja znatno pre isteka vremenskog limita, ali za instance sa većim p i n vrednostima dobijani su bolji rezultati sa porastom t_{max} ograničenja. Stoga, algoritmi su testirani i sa vrednostima parametara $k_{max} = p$ i $t_{max} = pn$ sekundi. U tom slučaju su dobijeni najbolji rezultati, tako da su u nastavku predstavljeni samo rezultati dobijeni za ovaj skup ulaznih vrednosti parametara.

5.3.1. Rezultati testiranja algoritma za *p*-sledeći medijan problem

Predloženi algoritam za rešavanje *p*-sledeći medijan problema je testiran nad svim primerima iz *OR-Library* test seta. Tabela 5.1 prikazuje zbirne rezultate testiranja, a detaljni rezultati za svaku izvornu instancu *OR-Library* test biblioteke su dati u Tabeli D.1 (Dodatak D). Prva kolona Tabele 5.1 sadrži naziv instance, naredne tri kolone predstavljaju vrednosti *p* (broj centara), *n* (broj korisnika) i *m* (broj grana grafa). U kolonama "*Best Value*", "*AVG Value*" i "*Worst Value*" prikazane su najbolja, prosečna i najgora vrednost rešenja koje je algoritam pronašao tokom 20 izvršenja. Kolona "*Time*" (ili *time-to-target*) prikazuje prosečno vreme u sekundama koje je bilo potrebno da se prvi put pronađe najbolje rešenje. "*Time*" nije ukupno vreme izvršenja. Algoritam se izvršava sve dok ne istekne vremenski limit, tj. *pn* sekundi, ili dok se ne izvrši poslednja faza lokalne pretrage. Kolona "*#Best*" daje broj koliko puta je algoritam pronašao najbolje rešenje u 20 izvršenja. Takođe, u Dodatak D uključene su dodatne kolone koje sadrže procentualno odstupanje prosečnog i najgoreg rešenja u odnosu na najbolje poznato rešenje dato u koloni "*Best-Known Value*". Kako *OR-Library* instance još uvek nisu korišćene za testiranje *p*-sledeći medijan problema, kao najbolje poznato rešenje uzima se najbolje rešenje koje je pronašao novi algoritam.

Tabela 5.1. Rezultati za pNMP algoritam testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
pmed1-pmed5	5-33	100	200	5726.20	5726.20	5726.20	9.70	20.00
pmed6-pmed10	5-67	200	800	6429.00	6439.17	6534.00	211.69	16.80
pmed11-pmed15	5-100	300	1800	6651.40	6651.98	6657.60	2373.24	17.60
pmed16-pmed20	5-133	400	3200	7072.00	7076.67	7108.80	5625.80	12.60
pmed21-pmed25	5-167	500	5000	7678.40	7679.83	7683.20	8479.06	11.80
pmed26-pmed30	5-200	600	7200	7695.60	7705.31	7735.40	16489.42	12.20
pmed31-pmed34	5-140	700	9800	9088.75	9096.56	9126.50	25679.07	13.25
pmed35-pmed37	5-80	800	1280	10918.67	10949.10	11055.00	16840.08	8.33
pmed38-pmed40	5-90	900	16200	10947.33	10966.43	11076.33	22992.21	9.33
AVG							**9703.94s**	**14.03**

Na osnovu Tabele 5.1, primetno je da sa porastom veličine test instanci, očekivano raste i vreme potrebno da se pronađe najbolje rešenje. Prosečno vreme pronalaska najboljeg rešenja nad kompletnim test skupom podataka je 9703.94 sekunde, dok je za najmanje instance (pmed1 - pmed5) samo 9.70 sekundi. Sa druge strane, u pogledu stabilnosti algoritma i kvaliteta rešenja, iz 20 izvršenja algoritam uspe da pronađe najbolje rešenje prosečno u 14.03 slučaja, ali prosečna odstupanja vrednosti najgorih i srednjih rešenja u odnosu na najbolja pronađena rešenja su veoma mala. Stabilnost algoritma potvrđuju rezultati procentualnih odstupanja iz Tabele D.1 (Dodatak D), od 0.49% odstupanja najgorih i samo 0.08% odstupanja prosečnih u odnosu na najbolja rešenja. Na osnovu dobijenih rezultata, zaključuje se da predloženi algoritam pronalazi optimalna ili skoro-optimalna rešenja u vremenskom intervalu srazmernom veličini problema i da se može uspešno koristiti za rešavanje *p*-sledeći medijan problema veličine do 900 čvorova.

5.3.2. Rezultati testiranja algoritma za *p*-α-sledeći medijan problem

Algoritam za rešavanje *p*-α-sledeći medijan problema je testiran za α vrednosti: $\alpha = p$, kao najširi mogući problem; i $\alpha = 2$, kao problem ekvivalentan *p*-sledeći medijan problemu. Vrednosti preostalih parametara su ostale nepromenjene, tj. $k_{max} = p$ i $t_{max} = pn$ sekundi. Algoritam je takođe izvršen po 20 puta i rezultati su prezentovani u tabelama koje slede.

Tabela 5.2. Rezultati za pαNMP algoritam ($\alpha = p$) testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*
pmed1-pmed5	5-33	100	200	30416.60	31892.82	35564.40	402.79	11.20
pmed6-pmed10	5-67	200	800	48457.40	52022.13	61273.60	3302.16	8.20
pmed11-pmed15	5-100	300	1800	86292.20	96226.94	113513.00	9047.10	7.80
pmed16-pmed20	5-133	400	3200	137473.00	152500.97	173298.40	17487.47	4.20
pmed21-pmed25	5-167	500	5000	179730.80	204663.59	232164.20	34090.43	8.00
pmed26-pmed30	5-200	600	7200	240748.80	294873.41	360044.00	63718.10	5.60
pmed31-pmed34	5-140	700	9800	168372.00	200945.71	247040.75	52885.52	4.75
pmed35-pmed37	5-80	800	1280	109836.67	124646.38	140105.00	22948.22	5.67
pmed38-pmed40	5-90	900	16200	111965.33	144515.17	173834.33	34028.83	3.00
AVG							**25567.84s**	**6.75**

Tabela 5.2 sadrži rezultate algoritma za *p*-α-sledeći medijan problem, gde je $\alpha = p$. Iz 20 izvršenja algoritam uspe da pronađe najbolje poznato rešenje prosečno u 6.75 slučajeva. Srednje vreme potrebno da se pronađe najbolje rešenje je 25567.84 sekunde. Kao i u slučaju *p*-sledeći medijan problema, veće instance problema zahtevaju više vremena, a takođe i prosečan broj pronalazaka najboljih poznatih rešenja opada sa porastom veličine problema. Na osnovu Tabele D.2 (Dodatak D), primetno je da su odstupanja vrednosti najgorih i srednjih rešenja u odnosu na najbolja pronađena rešenja prosečno čak 25.03%, odnosno 9.17%, što

ukazuje na to da bi trebalo da budu odbačena. Dakle, algoritam se uz višestruko izvršenje može primeniti za rešavanje problema veličine do 900 čvorova, ali u obzir je moguće uzeti samo najbolja rešenja.

Da bi se uporedili rezultati sa algoritmom za *p*-sledeći medijan, algoritam za rešavanje *p*-α-sledeći medijan problema je izvršen i sa parametrom $\alpha = 2$. Rezultati poređenja su predstavljeni u Tabeli 5.3 i Tabeli D.3 (Dodatak D). Tabele su proširene kolonom "*Best-Known Value*" koja sadrži do sada najbolja poznata rešenja problema, tj. najbolja rešenja dobijena algoritmom za *p*-sledeći medijan problem. U Tabelama 5.3 i D.3, takođe se prikazuje najbolje rešenje predloženog algoritma dobijeno u 20 izvršenja (kolona "*Best-Found Value*") i prosečno vreme pronalaska najboljeg rešenja (kolona "*Time*"). Kolona "*#Best-Known*" sadrži broj izvršenja algoritma koja su rezultirala pronalaskom najboljeg poznatog rešenja. Konačno, poslednja kolona daje procentualno odstupanje najboljeg pronađenog rešenja od najboljeg poznatog rešenja. Procenat odstupanja se računa kao $\frac{Best\ found - Best\ known}{Best\ known} * 100$. U Tabeli D.3 dati su detaljni rezultati izvršenja nad svakom instancom *OR-Library* test seta, uključujući vrednosti najgorih i srednjih rešenja, kao i njihova odstupanja u odnosu na vrednosti najboljih poznatih rešenja.

Tabela 5.3. Rezultati za pαNMP algoritam ($\alpha = 2$) testiran nad *OR-Library* test instancama

	P	*N*	*Best-Known Value*	*Best-Found Value*	*Time*	*#Best-Known*	*Gap (best found-best known) /best known*100*
pmed1-pmed5	5-33	100	5726.20	5726.20	54.69	20.00	0.00
pmed6-pmed10	5-67	200	6429.00	6429.00	3228.33	11.60	0.00
pmed11-pmed15	5-100	300	6651.40	6654.40	5812.60	8.60	0.07
pmed16-pmed20	5-133	400	7072.00	7096.80	12196.23	7.20	0.45
pmed21-pmed25	5-167	500	7678.40	7704.60	11444.44	6.60	0.54
pmed26-pmed30	5-200	600	7695.60	7732.60	17181.75	3.80	0.64
pmed31-pmed34	5-140	700	9088.75	9138.50	18393.04	7.75	0.83
pmed35-pmed37	5-80	800	10918.67	10979.00	8686.21	5.33	0.81
pmed38-pmed40	5-90	900	10947.33	10997.67	13585.19	6.33	0.67
AVG					**9749.41s**	**8.88**	**0.41%**

Tabele 5.3 i D.3 prikazuju rezultate algoritma za *p*-α-sledeći medijan problem gde je $\alpha = 2$, i primetno je da su rezultati nešto lošiji u odnosu na rezultate algoritma za ekvivalentni *p*-sledeći medijan problem. Algoritam za *p*-α-sledeći medijan problem pronalazi najbolja poznata rešenja prosečno u 8.88 od 20 izvršenja. Bilo je čak 15 instanci za koje nisu pronađena najbolja poznata rešenja, ali je prosečno odstupanje najboljih pronađenih u odnosu na najbolja poznata rešenja samo 0.41%. Iz Tabele D.3 (Dodatak D) se vidi da su čak i odstupanja najgorih i srednjih rešenja, od 1.77% i 0.94%, respektivno, relativno mala. To potvrđuje stabilnost i efikasnost algoritma u rešavanju *p*-sledeći medijan problema uz neznatno veći utrošak procesorskog vremena. Srednje vreme pronalaska najboljeg rešenja je 9749.41 sekund.

5.4. Zaključak

Problem *p*-sledećeg medijana predstavlja proširenje poznatog *p*-medijan problema. Rešenje ovog novog problema identifikuje lokacije *p* centara sa ciljem minimizacije sume rastojanja *n* korisnika do najbližeg centra plus rastojanja između tog centra i njemu najbližeg centra. Takođe, u radu je predstavljen i *p*-α-sledeći medijan problem kao uopštenje *p*-sledeći medijan problema na taj način što se minimizira rastojanje do α centara, gde je $2 \leq \alpha \leq p$. Dizajniran je i implementiran metod promenljivih okolina kao metaheuristički pristup rešavanju *p*-sledeći medijan i *p*-α-sledeći medijan problema.

Predloženi algoritmi su testirani nad *OR-Library* instancama iz literature do 900 čvorova i dobijeni eksperimentalni rezultati su pokazali da algoritmi pronalaze optimalna ili skoro-optimalna rešenja u intervalu vremena srazmernom veličini problema. Algoritam za *p*-sledeći medijan problem je pronašao najbolje rešenje u proseku u 14.03 od 20 slučajeva, ali su odstupanja vrednosti srednjih i najgorih rešenja u odnosu na najbolja rešenja bila manja od 0.5%. To potvrđuje stabilnost predloženog algoritma, tj. da se može uspešno koristiti za rešavanje *p*-sledeći medijan problema veličine do 900 čvorova. Prosečno vreme potrebno da se pronađe najbolje rešenje je ipak prilično veliko, 9703.94 sekunde.

Algoritam za rešavanje *p*-α-sledeći medijan problema je testiran za vrednosti $\alpha = p$ i $\alpha = 2$. Rezultati testiranja najširih problema ($\alpha = p$) pokazuju da algoritam u proseku identifikuje relativno mali broj najboljih rešenja (6.75 od 20), uz značajno odstupanje vrednosti najgorih i srednjih rešenja u odnosu na najbolja pronađena rešenja (25.03% i 9.17%, respektivno). Prema tome, samo najbolja rešenja višestruko izvršenog algoritma za *p*-α-sledeći medijan problem, gde je $\alpha = p$, bi trebalo uzimati u obzir. Takođe, vreme potrebno da se pronađe najbolje rešenje je veliko, 25567.84 sekunde. Algoritam je testiran i nad "najmanjim" problemima, kod kojih je potrebno minimizirati rastojanje samo do 2 najbliža centra ($\alpha = 2$). Problem je ekvivalentan *p*-sledeći medijan problemu što daje mogućnost da se uporede rezultati algoritama za *p*-sledeći medijan i *p*-α-sledeći medijan probleme. Ispostavilo se da je *p*-sledeći medijan algoritam nešto uspešniji, 0.41% u proseku. Vrednosti odstupanja rešenja algoritma za *p*-α-sledeći ($\alpha = 2$) medijan problem su relativno mala u odnosu na algoritam za *p*-sledeći medijan problem, što potvrđuje stabilnost i primenljivost algoritma za rešavanje *p*-α-sledeći medijan problema veličine do 900 čvorova. Vreme pronalaska najboljeg rešenja ostaje srazmerno veličini problema, ali ipak značajno veliko, 9749.41 sekund.

Kao što se vidi iz prethodnog izlaganja, predloženi algoritmi uspešno rešavaju *p*-sledeći medijan i *p*-α-sledeći medijan probleme, ali uz značajan utrošak procesorskog vremena. Interesantno bi bilo pokušati smanjiti to vreme, tj. ubrzati konvergenciju ka (lokalnom-)optimumu, recimo filtriranjem potencijalnih rešenja kao u algoritmu iz Poglavlja 2. Predloženi algoritmi tokom lokalne pretrage uzimaju u obzir sva rešenja iz N_1 okoline tekućeg rešenja. Neka od njih bi mogla biti isključena iz pretrage ukoliko bi se unapred znalo da neće unaprediti tekuće rešenje.

6. Za kraj disertacije

U disertaciji, zajedno sa predloženim rešenjima, predstavljeno je nekoliko "*p*-problema" nad skupom čvorova grafa koji kao deo svog rešenja daju odgovor i na eventualni otkaz dodeljenih centara. Prvi takav problem, definisan 2015. godine u radu [1], nazvan je *p*-sledeći centar problem. Predstavlja identifikaciju *p* od potencijalno *n* centara sa ciljem minimizacije maksimalne sume rastojanja korisnika do najbližeg centra i rastojanja između tog i njemu najbližeg centra, kao zamenskog centra koji preuzima ulogu opsluživanja korisnika u slučaju otkaza primarnog centra. Svi problemi su definisani na osnovu opšteg neusmerenog težinskog grafa čiji čvorovi predstavljaju korisnike i centre, a težine grana rastojanja između njih. Rešenje problema predstavlja promocija tačno *p* od ukupno *n* čvorova grafa u tzv. centre koji funkcionalno opslužuju sve korisnike, opet pozicionirane u čvorovima grafa. U osnovi, disertacija se bavi optimizacionim problemima, tako da se centri biraju sa ciljem da odgovarajuća rastojanja među njima, odnosno korisnicima, budu minimalna. Svi posmatrani problemi su NP-teški i stoga su predložena rešenja kao implementacija heurističkih algoritama zasnovanih na generičkom metodu promenljivih okolina.

Pored problema *p*-sledećeg centra u disertaciji je predloženo i heurističko rešenje *p*-drugi centar problema, koji za cilj ima identifikaciju *p* centara tako da je minimizovana maksimalna suma rastojanja od korisnika do najbližeg i drugog-najbližeg centra. Predloženi algoritmi koji rešavaju ove probleme implementiraju efikasne procedure filtriranja potencijalnih centara zasnovane na *Whitaker* strukturi podataka i teorijskim razmatranjima delimično preuzetim iz rada [28]. Teorijska osnova je dala čvrstu potporu za izgradnju veoma efikasnih algoritama koji za kratko vreme identifikuju najbolja poznata rešenja problema *p*-sledećeg i *p*-drugog centra. U literaturi je već poznato nekoliko algoritama za rešavanje *p*-sledeći centar problema. Predloženi algoritam se u poređenju sa njima, primenjen nad istim instancama problema, pokazao znatno efikasnijim, a naročito u kontekstu instanci većih problema, gde je većinom ustanovio nova najbolja poznata rešenja. U disertaciji su predložene i modifikacije algoritama za prepoznavanje egzaktnih rešenja *p*-sledeći i *p*-drugi centar problema. Primenjeni nad u literaturi poznatim *OR-Library* test skupom, algoritmi su identifikovali 22 egzaktna rešenja *p*-sledeći centar, odnosno 12 egzaktnih rešenja *p*-drugi centar problema, od ukupno 40 instanci koliko sadrži posmatrana test platforma. S obzirom da je većina u literaturi poznatih matematičkih modela i algoritama za rešavanje ovih problema testirana nad izvornim ili modifikovanim *OR-Library* test skupom podataka, identifikovana globalno optimalna rešenja su od posebnog značaja za dalje proučavanje i procenu predloženih metoda za rešavanje problema *p*-sledećeg i *p*-drugog centra.

Definisan je i *p*-sledeći medijan problem koji za cilj ima identifikaciju *p* centara tako da je ukupna suma rastojanja svih korisnika do najbližih centara plus rastojanja između tih centara i njima najbližih centara minimizovana. Predloženi algoritam koji rešava ovaj problem je takođe testiran nad *OR-Library* test skupom, tako da je ustanovio inicijalna rešenja *p*-sledeći medijan problema definisanih ovim test instancama. U disertaciji se razmatra i potencijalni otkaz većeg broja dodeljenih centara i stoga su u kontekstu *p*-centar i *p*-medijan problema predstavljena rešenja koja svakom od korisnika dodeljuju proizvoljan broj (α) centara, tj. jedan primarni i α - *1* zamenskih centara. Ovi problemi su nazvani kao problemi *p*-α-sledećih centara, *p*-α-najbližih centara i *p*-α-sledeći medijan problem, i predstavljaju identifikaciju *p* od potencijalno *n* centara, tako da je minimizovana odgovarajuća suma rastojanja korisnika do, odnosno između, α dodeljenih centara. Problemi su definisani kao uopštenja *p*-sledeći centar, *p*-drugi centar i *p*-sledeći medijan problema, a rešenja predstavljena kao generički algoritam, izgrađen na metodu promenljivih okolina, koji uključuje jednostavnu proceduru evaluacije vrednosti funkcije cilja konkretnog problema. Algoritam pronalazi optimalna ili skoro-optimalna rešenja posmatranih problema, ali uz značajan utrošak procesorskog vremena. Akcenat u implementaciji je na generičkom rešenju koje pokriva više problema bez mnogo teorijskih razmatranja. Upravo u poređenju sa algoritmima za probleme *p*-sledećeg i *p*-drugog centra se vidi značaj teorijske potpore u implementaciji rešenja. Algoritmi za *p*-sledeći i *p*-drugi centar probleme jesu stabilniji i efikasniji, ali njihova superiornost u odnosu na ponuđene algoritme za opštije probleme značajno dolazi do izražaja tek u poređenju utrošenih vremena do prvog pronalaska najboljih rešenja. Nad *OR-Library* test instancama, algoritmu za problem *p*-sledećeg centra je prosečno potrebno 20.86 sekundi, odnosno 31.32 sekunde za rešenje problema *p*-drugog centra, a recimo za *p*-sledeći medijan problem 9703.94 sekunde ili čak 13040.57 i 22439.83 sekunde za probleme *p*-α-najbližih i *p*-α-sledećih centara. Na osnovu toga, nameće se zaključak da je implementacija algoritma promenljivih okolina ali

zasnovana na dovoljnom nivou ekspertskog znanja, tj. dobro razrađenim teorijskim osnovama problema kojim se bavi, izuzetno moćan metod za rešavanje optimizacionih problema primenljiv i u okolnostima koje zahtevaju brz odgovor. Kao tema za buduće istraživanje, interesantno bi bilo podržati proceduru za filtriranje potencijalnih rešenja problema p-α-najbližih centara, p-α-sledećih centara i p-α-sledeći medijan problema odgovarajućom teorijskom osnovom koja bi efikasno eliminisala neobećavajuća rešenja.

Dodatak A

Dodatak A sadrži tabele sa detaljnim rezultatima testiranja predloženog VNS algoritma za rešavanje problema *p*-sledećeg centra.

Tabela A.1. Detaljni rezultati za sofisticirani VNS algoritam testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1	5	100	200	166	166.00	166	0.14	20	0.00	0.00
pmed2	10	100	200	135	135.00	135	1.47	20	0.00	0.00
pmed3	10	100	200	151	151.00	151	0.49	20	0.00	0.00
pmed4	20	100	200	118	118.00	118	0.78	20	0.00	0.00
pmed5	33	100	200	85	85.00	85	0.10	20	0.00	0.00
pmed6	5	200	800	107	107.00	107	1.91	20	0.00	0.00
pmed7	10	200	800	84	84.00	84	10.22	20	0.00	0.00
pmed8	20	200	800	81	81.00	81	17.29	20	0.00	0.00
pmed9	40	200	800	71	71.00	71	0.41	20	0.00	0.00
pmed10	67	200	800	70	70.00	70	0.16	20	0.00	0.00
pmed11	5	300	1800	70	70.00	70	0.76	20	0.00	0.00
pmed12	10	300	1800	72	72.00	72	0.85	20	0.00	0.00
pmed13	30	300	1800	47	47.00	47	17.91	20	0.00	0.00
pmed14	60	300	1800	60	60.00	60	0.74	20	0.00	0.00
pmed15	100	300	1800	44	44.00	44	1.68	20	0.00	0.00
pmed16	5	400	3200	54	54.00	54	57.07	20	0.00	0.00
pmed17	10	400	3200	46	46.35	47	110.24	13	2.17	0.76
pmed18	40	400	3200	50	50.00	50	1.67	20	0.00	0.00
pmed19	80	400	3200	32	32.00	32	10.61	20	0.00	0.00
pmed20	133	400	3200	40	40.00	40	3.05	20	0.00	0.00
pmed21	5	500	5000	48	48.05	49	10.90	19	2.08	0.10
pmed22	10	500	5000	49	49.00	49	72.78	20	0.00	0.00
pmed23	50	500	5000	32	32.00	32	50.16	20	0.00	0.00
pmed24	100	500	5000	33	33.00	33	6.89	20	0.00	0.00
pmed25	167	500	5000	44	44.00	44	1.99	20	0.00	0.00
pmed26	5	600	7200	47	47.00	47	14.34	20	0.00	0.00
pmed27	10	600	7200	38	38.10	39	124.89	18	2.63	0.26
pmed28	60	600	7200	57	57.00	57	2.71	20	0.00	0.00
pmed29	120	600	7200	36	36.00	36	4.14	20	0.00	0.00

pmed30	200	600	7200	40	40.00	40	2.65	20	0.00	0.00
pmed31	5	700	9800	35	35.00	35	18.77	20	0.00	0.00
pmed32	10	700	9800	72	72.00	72	4.07	20	0.00	0.00
pmed33	70	700	9800	22	22.75	23	193.32	5	4.55	3.41
pmed34	140	700	9800	41	41.00	41	4.39	20	0.00	0.00
pmed35	5	800	12800	36	36.00	36	17.96	20	0.00	0.00
pmed36	10	800	12800	42	42.00	42	6.95	20	0.00	0.00
pmed37	80	800	12800	33	33.00	33	6.49	20	0.00	0.00
pmed38	5	900	16200	40	40.00	40	10.16	20	0.00	0.00
pmed39	10	900	16200	74	74.00	74	9.29	20	0.00	0.00
pmed40	90	900	16200	23	23.00	23	34.15	20	0.00	0.00
AVG							**20.86s**	**19.38**	**0.29%**	**0.11%**

Tabela A.2. Detaljni rezultati poređenja sa Filtriranim VNS metodom iz rada Ristića i sar.

	P	*N*	*Best Known Value*	*Best Found Value*	*Time*	*#Best Known*	*Gap (best found-best known) /best known*100*
pmed1	5	100	166	166	0.50	19	0.00
pmed2	10	100	135	135	100.46	16	0.00
pmed3	10	100	151	151	87.24	15	0.00
pmed4	20	100	118	118	177.84	5	0.00
pmed5	33	100	85	85	38.63	20	0.00
pmed6	5	200	107	107	146.09	14	0.00
pmed7	10	200	84	84	390.52	2	0.00
pmed8	20	200	81	84	332.01	0	3.70
pmed9	40	200	71	71	261.44	2	0.00
pmed10	67	200	70	70	9.74	20	0.00
pmed11	5	300	70	70	168.67	19	0.00
pmed12	10	300	72	72	544.26	16	0.00
pmed13	30	300	47	52	658.49	0	10.64
pmed14	60	300	60	60	515.14	8	0.00
pmed15	100	300	44	44	811.39	13	0.00
pmed16	5	400	54	55	106.52	0	1.85
pmed17	10	400	46	47	791.43	0	2.17
pmed18	40	400	50	50	1101.69	6	0.00
pmed19	80	400	32	40	1113.12	0	25.00
pmed20	133	400	40	40	1079.11	4	0.00

pmed21	5	500	48	48	377.36	5	0.00
pmed22	10	500	49	52	849.43	0	6.12
pmed23	50	500	32	42	1371.95	0	31.25
pmed24	100	500	33	35	1196.55	0	6.06
pmed25	167	500	44	44	242.45	20	0.00
pmed26	5	600	47	47	905.53	7	0.00
pmed27	10	600	38	40	946.39	0	5.26
pmed28	60	600	57	57	20.40	20	0.00
pmed29	120	600	36	36	1461.31	9	0.00
pmed30	200	600	40	40	108.86	20	0.00
pmed31	5	700	35	35	1057.44	7	0.00
pmed32	10	700	72	72	15.29	20	0.00
pmed33	70	700	22	33	1611.42	0	50.00
pmed34	140	700	41	41	72.14	20	0.00
pmed35	5	800	36	36	927.56	4	0.00
pmed36	10	800	42	42	223.52	20	0.00
pmed37	80	800	33	33	1634.93	6	0.00
pmed38	5	900	40	40	1102.41	16	0.00
pmed39	10	900	74	74	32.70	20	0.00
pmed40	90	900	23	29	1394.52	0	26.09
AVG					**599.66s**	**9.32**	**4.20%**

Tabela A.3. Detaljni rezultati poređenja sa hibridnim algoritmom Lopez-Sancheza i sar. za manje instance

	P	*N*	*Hybrid Value*	*Time*	*Best Value*	*#Best*	*Gap (best-hybrid)/ hybrid*100*
pmed1_01	5	10	84	0.001	84	20	0.00
pmed1_02	5	20	120	0.004	120	20	0.00
pmed1_03	10	20	95	0.003	95	20	0.00
pmed1_04	5	30	126	0.007	126	20	0.00
pmed1_05	10	30	95	0.008	95	20	0.00
pmed1_06	5	40	144	0.492	144	20	0.00
pmed1_07	10	40	111	0.298	111	20	0.00
pmed1_08	20	40	89	0.007	89	20	0.00
pmed1_09	10	50	111	0.404	110	20	-0.90
pmed1_10	20	50	89	0.101	89	20	0.00
pmed2_01	5	10	128	0.011	121	20	-5.47

pmed2_02	5	20	147	0.005	147	20	0.00
pmed2_03	10	20	99	0.047	99	20	0.00
pmed2_04	5	30	169	0.013	169	20	0.00
pmed2_05	10	30	110	0.009	110	20	0.00
pmed2_06	5	40	164	0.015	164	20	0.00
pmed2_07	10	40	112	0.234	112	20	0.00
pmed2_08	20	40	96	0.010	96	20	0.00
pmed2_09	10	50	140	0.091	140	20	0.00
pmed2_10	20	50	99	0.037	99	20	0.00
pmed3_01	5	10	77	0.001	77	20	0.00
pmed3_02	5	20	145	0.005	145	20	0.00
pmed3_03	10	20	77	0.003	77	20	0.00
pmed3_04	5	30	157	0.012	157	20	0.00
pmed3_05	10	30	122	0.575	122	20	0.00
pmed3_06	5	40	157	0.030	157	20	0.00
pmed3_07	10	40	105	0.125	105	20	0.00
pmed3_08	20	40	77	0.023	77	20	0.00
pmed3_09	10	50	125	0.729	125	20	0.00
pmed3_10	20	50	87	0.056	87	20	0.00
pmed4_01	5	10	126	0.006	126	20	0.00
pmed4_02	5	20	139	0.004	139	20	0.00
pmed4_03	10	20	125	0.028	125	20	0.00
pmed4_04	5	30	173	0.041	173	20	0.00
pmed4_05	10	30	122	0.010	122	20	0.00
pmed4_06	5	40	175	0.014	175	20	0.00
pmed4_07	10	40	122	0.283	122	20	0.00
pmed4_08	20	40	85	0.018	85	20	0.00
pmed4_09	10	50	126	0.071	126	20	0.00
pmed4_10	20	50	91	0.177	91	20	0.00
AVG				**0.100s**		**20**	**-0.16%**

Tabela A.4. Detaljni rezultati poređenja sa hibridnim algoritmom Lopez-Sancheza i sar. za veće instance

	P	*N*	*Hybrid Value*	*Time*	*Worst Value*	*AVG Value*	*Best Value*	*#Best*	*Gap Worst vs. Hybrid*	*Gap AVG vs. Hybrid*	*Gap Best vs. Hybrid*
pmed1_01	10	60	112	0.22	112	112.00	112	20	0.00	0.00	0.00
pmed1_02	20	60	91	0.15	89	89.00	89	20	-2.20	-2.20	-2.20

pmed1_03	30	60	89	0.02	89	89.00	89	20	0.00	0.00	0.00
pmed1_04	10	70	119	0.52	119	119.00	119	20	0.00	0.00	0.00
pmed1_05	20	70	99	0.05	99	99.00	99	20	0.00	0.00	0.00
pmed1_06	30	70	73	0.10	73	73.00	73	20	0.00	0.00	0.00
pmed1_07	10	80	133	3.69	129	129.00	129	20	-3.01	-3.01	-3.01
pmed1_08	20	80	105	0.75	102	102.00	102	20	-2.86	-2.86	-2.86
pmed1_09	30	80	91	0.49	85	85.00	85	20	-6.59	-6.59	-6.59
pmed1_10	10	90	133	1.15	133	133.00	133	20	0.00	0.00	0.00
pmed1_11	20	90	108	7.46	107	107.00	107	20	-0.93	-0.93	-0.93
pmed1_12	30	90	91	1.06	87	87.00	87	20	-4.40	-4.40	-4.40
pmed1_13	50	90	70	0.15	70	70.00	70	20	0.00	0.00	0.00
pmed1_14	10	100	133	0.82	133	133.00	133	20	0.00	0.00	0.00
pmed1_15	20	100	108	1.10	108	108.00	108	20	0.00	0.00	0.00
pmed1_16	30	100	97	6.90	93	93.00	93	20	-4.12	-4.12	-4.12
pmed1_17	50	100	74	1.11	70	70.00	70	20	-5.41	-5.41	-5.41
pmed2_01	10	60	140	0.22	140	140.00	140	20	0.00	0.00	0.00
pmed2_02	20	60	99	0.11	99	99.00	99	20	0.00	0.00	0.00
pmed2_03	30	60	96	0.02	96	96.00	96	20	0.00	0.00	0.00
pmed2_04	10	70	138	0.23	138	138.00	138	20	0.00	0.00	0.00
pmed2_05	20	70	102	0.08	102	102.00	102	20	0.00	0.00	0.00
pmed2_06	30	70	96	0.04	96	96.00	96	20	0.00	0.00	0.00
pmed2_07	10	80	138	2.35	138	138.00	138	20	0.00	0.00	0.00
pmed2_08	20	80	109	0.41	109	109.00	109	20	0.00	0.00	0.00
pmed2_09	30	80	97	0.19	97	97.00	97	20	0.00	0.00	0.00
pmed2_10	10	90	140	7.26	138	138.00	138	20	-1.43	-1.43	-1.43
pmed2_11	20	90	109	11.54	109	108.75	108	5	0.00	-0.23	-0.92
pmed2_12	30	90	97	0.43	96	96.00	96	20	-1.03	-1.03	-1.03
pmed2_13	50	90	96	0.05	96	96.00	96	20	0.00	0.00	0.00
pmed2_14	10	100	135	1.27	135	135.00	135	20	0.00	0.00	0.00
pmed2_15	20	100	109	4.23	109	107.10	107	19	0.00	-1.74	-1.83
pmed2_16	30	100	96	0.44	96	96.00	96	20	0.00	0.00	0.00
pmed2_17	50	100	96	0.06	96	96.00	96	20	0.00	0.00	0.00
AVG				**1.61s**				**19.53**	**-0.94%**	**-1.00%**	**-1.02%**

Tabela A.5. Detaljni rezultati poređenja sa hibridnim algoritmom Lopez-Sancheza i sar. za veće instance

	P	*N*	*Hybrid Value*	*Time*	*Worst Value*	*AVG Value*	*Best Value*	*#Best*	*Gap Worst vs. Hybrid*	*Gap AVG vs. Hybrid*	*Gap Best vs. Hybrid*
pmed3_01	10	60	124	0.50	124	124.00	124	20	0.00	0.00	0.00
pmed3_02	20	60	97	0.07	97	97.00	97	20	0.00	0.00	0.00
pmed3_03	30	60	73	0.13	73	73.00	73	20	0.00	0.00	0.00
pmed3_04	10	70	121	1.15	121	121.00	121	20	0.00	0.00	0.00
pmed3_05	20	70	97	0.18	97	97.00	97	20	0.00	0.00	0.00
pmed3_06	30	70	82	0.65	82	82.00	82	20	0.00	0.00	0.00
pmed3_07	10	80	121	1.47	121	121.00	121	20	0.00	0.00	0.00
pmed3_08	20	80	93	0.29	93	93.00	93	20	0.00	0.00	0.00
pmed3_09	30	80	86	0.58	84	84.00	84	20	-2.33	-2.33	-2.33
pmed3_10	10	90	148	0.40	148	148.00	148	20	0.00	0.00	0.00
pmed3_11	20	90	105	3.36	105	105.00	105	20	0.00	0.00	0.00
pmed3_12	30	90	93	0.15	93	93.00	93	20	0.00	0.00	0.00
pmed3_13	50	90	93	0.04	93	93.00	93	20	0.00	0.00	0.00
pmed3_14	10	100	151	0.50	151	151.00	151	20	0.00	0.00	0.00
pmed3_15	20	100	113	10.83	114	110.45	109	11	0.88	-2.26	-3.54
pmed3_16	30	100	93	0.19	93	93.00	93	20	0.00	0.00	0.00
pmed3_17	50	100	93	0.06	93	93.00	93	20	0.00	0.00	0.00
pmed4_01	10	60	135	0.10	135	134.95	134	1	0.00	-0.04	-0.74
pmed4_02	20	60	93	0.04	93	93.00	93	20	0.00	0.00	0.00
pmed4_03	30	60	79	0.08	79	79.00	79	20	0.00	0.00	0.00
pmed4_04	10	70	146	3.08	146	146.00	146	20	0.00	0.00	0.00
pmed4_05	20	70	102	0.28	102	102.00	102	20	0.00	0.00	0.00
pmed4_06	30	70	85	0.06	85	85.00	85	20	0.00	0.00	0.00
pmed4_07	10	80	146	2.19	146	146.00	146	20	0.00	0.00	0.00
pmed4_08	20	80	114	26.77	114	114.00	114	20	0.00	0.00	0.00
pmed4_09	30	80	91	0.57	90	90.00	90	20	-1.10	-1.10	-1.10
pmed4_10	10	90	147	0.29	147	147.00	147	20	0.00	0.00	0.00
pmed4_11	20	90	112	8.23	116	112.20	112	19	3.57	0.18	0.00
pmed4_12	30	90	92	5.44	92	92.00	92	20	0.00	0.00	0.00
pmed4_13	50	90	82	0.09	82	82.00	82	20	0.00	0.00	0.00
pmed4_14	10	100	147	20.79	147	147.00	147	20	0.00	0.00	0.00
pmed4_15	20	100	119	1.16	118	118.00	118	20	-0.84	-0.84	-0.84
pmed4_16	30	100	96	1.49	96	96.00	96	20	0.00	0.00	0.00

pmed4_17	50	100	82	0.21	82	82.00	82	20	0.00	0.00	0.00
AVG				**2.69s**				**19.15**	**0.01%**	**-0.19%**	**-0.25%**

Tabela A.6. Detaljni rezultati poređenja sa hibridnim algoritmom Lopez-Sancheza i sar. za veće instance

	P	*N*	*Hybrid Value*	*Time*	*Worst Value*	*AVG Value*	*Best Value*	*#Best*	*Gap Worst vs. Hybrid*	*Gap AVG vs. Hybrid*	*Gap Best vs. Hybrid*
pmed6_01	20	150	79	2.49	77	77.00	77	20	-2.53	-2.53	-2.53
pmed6_02	30	150	71	22.53	65	64.85	64	3	-8.45	-8.66	-9.86
pmed6_03	50	150	62	0.69	56	56.00	56	20	-9.68	-9.68	-9.68
pmed6_04	80	150	56	0.30	56	56.00	56	20	0.00	0.00	0.00
pmed6_05	20	200	79	25.02	77	76.80	76	4	-2.53	-2.78	-3.80
pmed6_06	30	200	72	37.29	66	66.00	66	20	-8.33	-8.33	-8.33
pmed6_07	50	200	68	81.82	49	49.00	49	20	-27.94	-27.94	-27.94
pmed6_08	80	200	54	1.41	49	49.00	49	20	-9.26	-9.26	-9.26
pmed7_01	20	150	69	5.34	67	67.00	67	20	-2.90	-2.90	-2.90
pmed7_02	30	150	62	1.14	59	59.00	59	20	-4.84	-4.84	-4.84
pmed7_03	50	150	59	0.24	59	59.00	59	20	0.00	0.00	0.00
pmed7_04	80	150	59	0.18	59	59.00	59	20	0.00	0.00	0.00
pmed7_05	20	200	73	8.84	68	68.00	68	20	-6.85	-6.85	-6.85
pmed7_06	30	200	68	28.24	60	60.00	60	20	-11.76	-11.76	-11.76
pmed7_07	50	200	63	6.23	50	50.00	50	20	-20.63	-20.63	-20.63
pmed7_08	80	200	52	1.72	46	46.00	46	20	-11.54	-11.54	-11.54
pmed8_01	20	150	74	35.59	73	72.05	70	4	-1.35	-2.64	-5.41
pmed8_02	30	150	61	1.68	58	58.00	58	20	-4.92	-4.92	-4.92
pmed8_03	50	150	58	0.27	58	58.00	58	20	0.00	0.00	0.00
pmed8_04	80	150	58	0.18	58	58.00	58	20	0.00	0.00	0.00
pmed8_05	20	200	84	18.91	81	81.00	81	20	-3.57	-3.57	-3.57
pmed8_06	30	200	77	26.19	69	69.00	69	20	-10.39	-10.39	-10.39
pmed8_07	50	200	68	0.52	68	68.00	68	20	0.00	0.00	0.00
pmed8_08	80	200	68	0.38	68	68.00	68	20	0.00	0.00	0.00
AVG				**12.80s**				**17.96**	**-6.14%**	**-6.22%**	**-6.43%**

Tabela A.7. Detaljni rezultati za test instance definisane nad grafovima sa različitim gustinama

	P	*N*	*Density*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
rnddnskreg1	5	500	50.10	7	7.00	7	3.54	20	0.00	0.00

rnddnskreg2	10	500	50.10	6	6.00	6	12.05	20	0.00	0.00
rnddnskreg3	50	500	50.10	5	5.00	5	7.69	20	0.00	0.00
rnddnskreg4	200	500	50.10	4	4.00	4	2.90	20	0.00	0.00
rnddnskreg5	5	500	60.12	6	6.00	6	3.68	20	0.00	0.00
rnddnskreg6	10	500	60.12	6	6.00	6	3.90	20	0.00	0.00
rnddnskreg7	50	500	60.12	4	4.00	4	14.03	20	0.00	0.00
rnddnskreg8	200	500	60.12	3	3.00	3	12.23	20	0.00	0.00
rnddnskreg9	5	500	80.16	5	5.00	5	11.45	20	0.00	0.00
rnddnskreg10	10	500	80.16	5	5.00	5	4.99	20	0.00	0.00
rnddnskreg11	50	500	80.16	4	4.00	4	2.69	20	0.00	0.00
rnddnskreg12	200	500	80.16	3	3.00	3	2.75	20	0.00	0.00
rnddnskreg13	5	600	50.08	6	6.00	6	23.05	20	0.00	0.00
rnddnskreg14	10	600	50.08	6	6.00	6	9.57	20	0.00	0.00
rnddnskreg15	50	600	50.08	4	4.90	5	38.60	2	25.00	22.50
rnddnskreg16	200	600	50.08	4	4.00	4	4.31	20	0.00	0.00
rnddnskreg17	5	600	60.10	6	6.00	6	6.54	20	0.00	0.00
rnddnskreg18	10	600	60.10	5	5.00	5	68.17	20	0.00	0.00
rnddnskreg19	50	600	60.10	4	4.00	4	15.61	20	0.00	0.00
rnddnskreg20	200	600	60.10	3	3.00	3	15.09	20	0.00	0.00
rnddnskreg21	5	600	80.13	5	5.00	5	6.55	20	0.00	0.00
rnddnskreg22	10	600	80.13	5	5.00	5	7.26	20	0.00	0.00
rnddnskreg23	50	600	80.13	4	4.00	4	4.71	20	0.00	0.00
rnddnskreg24	200	600	80.13	3	3.00	3	3.96	20	0.00	0.00
rnddnskreg25	5	800	50.06	6	6.00	6	15.08	20	0.00	0.00
rnddnskreg26	10	800	50.06	5	5.25	6	201.55	15	20.00	5.00
rnddnskreg27	50	800	50.06	4	4.00	4	118.06	20	0.00	0.00
rnddnskreg28	200	800	50.06	3	3.00	3	46.23	20	0.00	0.00
rnddnskreg29	5	800	60.08	5	5.00	5	35.65	20	0.00	0.00
rnddnskreg30	10	800	60.08	5	5.00	5	20.87	20	0.00	0.00
rnddnskreg31	50	800	60.08	4	4.00	4	19.53	20	0.00	0.00
rnddnskreg32	200	800	60.08	3	3.00	3	21.50	20	0.00	0.00
rnddnskreg33	5	800	80.10	5	5.00	5	15.45	20	0.00	0.00
rnddnskreg34	10	800	80.10	5	5.00	5	17.97	20	0.00	0.00

rnddnskreg35	50	800	80.10	4	4.00	4	10.25	20	0.00	0.00
rnddnskreg36	200	800	80.10	3	3.00	3	10.62	20	0.00	0.00
rnddnskreg37	5	1000	50.05	5	5.00	5	77.63	20	0.00	0.00
rnddnskreg38	10	1000	50.05	5	5.00	5	81.13	20	0.00	0.00
rnddnskreg39	50	1000	50.05	4	4.00	4	52.23	20	0.00	0.00
rnddnskreg40	200	1000	50.05	3	3.00	3	66.80	20	0.00	0.00
rnddnskreg41	5	1000	60.06	5	5.00	5	36.30	20	0.00	0.00
rnddnskreg42	10	1000	60.06	5	5.00	5	38.65	20	0.00	0.00
rnddnskreg43	50	1000	60.06	4	4.00	4	39.37	20	0.00	0.00
rnddnskreg44	200	1000	60.06	3	3.00	3	32.73	20	0.00	0.00
rnddnskreg45	5	1000	80.08	5	5.00	5	33.98	20	0.00	0.00
rnddnskreg46	10	1000	80.08	4	4.00	4	85.67	20	0.00	0.00
rnddnskreg47	50	1000	80.08	4	4.00	4	17.70	20	0.00	0.00
rnddnskreg48	200	1000	80.08	3	3.00	3	24.55	20	0.00	0.00
AVG			63.43%				**29.27s**	**19.52**	**0.94%**	**0.57%**

Tabela A.8. Detaljni rezultati za velike test instance

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
rndkreg1	5	1000	50000	14	14.00	14	26.24	20	0.00	0.00
rndkreg2	10	1000	50000	12	12.00	12	88.25	20	0.00	0.00
rndkreg3	20	1000	50000	11	11.45	12	210.22	11	9.09	4.09
rndkreg4	33	1000	50000	11	11.00	11	91.45	20	0.00	0.00
rndkreg5	50	1000	50000	10	10.65	11	177.69	7	10.00	6.50
rndkreg6	67	1000	50000	10	10.00	10	114.41	20	0.00	0.00
rndkreg7	100	1000	50000	9	9.00	9	117.98	20	0.00	0.00
rndkreg8	133	1000	50000	8	8.00	8	164.60	20	0.00	0.00
rndkreg9	150	1000	50000	8	8.00	8	310.48	20	0.00	0.00
rndkreg10	167	1000	50000	7	7.85	8	217.72	3	14.29	12.14
rndkreg11	200	1000	50000	7	7.30	8	520.05	14	14.29	4.29
rndkreg12	5	1500	112500	10	10.10	11	487.63	18	10.00	1.00
rndkreg13	10	1500	112500	10	10.00	10	114.73	20	0.00	0.00
rndkreg14	20	1500	112500	9	9.90	10	169.24	2	11.11	10.00
rndkreg15	33	1500	112500	9	9.00	9	391.06	20	0.00	0.00

rndkreg16	50	1500	112500	9	9.00	9	129.29	20	0.00	0.00
rndkreg17	67	1500	112500	8	8.40	9	323.43	12	12.50	5.00
rndkreg18	100	1500	112500	8	8.00	8	164.97	20	0.00	0.00
rndkreg19	133	1500	112500	7	7.00	7	436.41	20	0.00	0.00
rndkreg20	150	1500	112500	8	8.00	8	80.40	20	0.00	0.00
rndkreg21	167	1500	112500	7	7.00	7	335.15	20	0.00	0.00
rndkreg22	200	1500	112500	6	6.65	7	598.27	7	16.67	10.83
rndkreg23	5	2000	200000	10	10.00	10	219.93	20	0.00	0.00
rndkreg24	10	2000	200000	9	9.00	9	304.62	20	0.00	0.00
rndkreg25	20	2000	200000	8	8.50	9	581.56	10	12.50	6.25
rndkreg26	33	2000	200000	8	8.20	9	600.99	16	12.50	2.50
rndkreg27	50	2000	200000	8	8.00	8	324.95	20	0.00	0.00
rndkreg28	67	2000	200000	7	7.95	8	331.13	1	14.29	13.57
rndkreg29	100	2000	200000	7	7.00	7	352.57	20	0.00	0.00
rndkreg30	133	2000	200000	6	6.95	7	333.16	1	16.67	15.83
rndkreg31	150	2000	200000	6	6.70	7	614.78	6	16.67	11.67
rndkreg32	167	2000	200000	6	6.30	7	971.83	14	16.67	5.00
rndkreg33	200	2000	200000	6	6.00	6	889.99	20	0.00	0.00
rndkreg34	5	2500	312500	8	8.95	9	492.12	1	12.50	11.88
rndkreg35	10	2500	312500	8	8.00	8	615.86	20	0.00	0.00
rndkreg36	20	2500	312500	7	7.90	8	670.23	2	14.29	12.86
rndkreg37	33	2500	312500	7	7.90	8	713.28	2	14.29	12.86
rndkreg38	50	2500	312500	7	7.25	8	989.51	15	14.29	3.57
rndkreg39	67	2500	312500	7	7.00	7	724.95	20	0.00	0.00
rndkreg40	100	2500	312500	7	7.00	7	514.50	20	0.00	0.00
rndkreg41	133	2500	312500	6	6.00	6	850.72	20	0.00	0.00
rndkreg42	150	2500	312500	6	6.00	6	860.54	20	0.00	0.00
rndkreg43	167	2500	312500	6	6.00	6	732.00	20	0.00	0.00
rndkreg44	200	2500	312500	6	6.00	6	643.56	20	0.00	0.00
AVG							**422.78s**	**15.05**	**5.51%**	**3.41%**

Dodatak B.1

Dodatak B.1 sadrži tabele sa detaljnim rezultatima testiranja predloženog VNS algoritma za rešavanje problema *p*-drugog centra (pSCP).

Tabela B.1.1. Detaljni rezultati za pSCP-VNS algoritam testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1	5	100	200	268	268.00	268	0.17	20	0.00	0.00
pmed2	10	100	200	220	220.00	220	14.15	20	0.00	0.00
pmed3	10	100	200	208	208.00	208	1.72	20	0.00	0.00
pmed4	20	100	200	163	163.00	163	0.44	20	0.00	0.00
pmed5	33	100	200	110	110.00	110	0.03	20	0.00	0.00
pmed6	5	200	800	180	180.00	180	0.57	20	0.00	0.00
pmed7	10	200	800	143	143.10	144	11.18	18	0.70	0.07
pmed8	20	200	800	122	122.00	122	9.27	20	0.00	0.00
pmed9	40	200	800	85	85.00	85	0.54	20	0.00	0.00
pmed10	67	200	800	70	70.00	70	0.10	20	0.00	0.00
pmed11	5	300	1800	125	125.00	125	0.82	20	0.00	0.00
pmed12	10	300	1800	112	112.00	112	3.82	20	0.00	0.00
pmed13	30	300	1800	78	78.00	78	3.55	20	0.00	0.00
pmed14	60	300	1800	60	60.00	60	2.61	20	0.00	0.00
pmed15	100	300	1800	44	44.00	44	0.34	20	0.00	0.00
pmed16	5	400	3200	98	98.00	98	1.48	20	0.00	0.00
pmed17	10	400	3200	83	83.00	83	8.25	20	0.00	0.00
pmed18	40	400	3200	62	62.00	62	136.35	20	0.00	0.00
pmed19	80	400	3200	42	42.00	42	25.64	20	0.00	0.00
pmed20	133	400	3200	40	40.00	40	0.73	20	0.00	0.00
pmed21	5	500	5000	85	85.00	85	8.24	20	0.00	0.00
pmed22	10	500	5000	80	80.00	80	56.96	20	0.00	0.00
pmed23	50	500	5000	49	49.05	50	162.15	19	2.04	0.10
pmed24	100	500	5000	35	35.00	35	11.86	20	0.00	0.00
pmed25	167	500	5000	44	44.00	44	1.53	20	0.00	0.00
pmed26	5	600	7200	80	80.00	80	100.93	20	0.00	0.00
pmed27	10	600	7200	67	67.00	67	66.08	20	0.00	0.00

pmed28	60	600	7200	57	57.00	57	2.58	20	0.00	0.00
pmed29	120	600	7200	36	36.00	36	2.99	20	0.00	0.00
pmed30	200	600	7200	40	40.00	40	2.58	20	0.00	0.00
pmed31	5	700	9800	64	64.00	64	6.46	20	0.00	0.00
pmed32	10	700	9800	72	72.00	72	4.31	20	0.00	0.00
pmed33	70	700	9800	35	35.00	35	64.20	20	0.00	0.00
pmed34	140	700	9800	41	41.00	41	4.36	20	0.00	0.00
pmed35	5	800	12800	64	64.00	64	127.65	20	0.00	0.00
pmed36	10	800	12800	58	58.00	58	111.97	20	0.00	0.00
pmed37	80	800	12800	33	33.50	34	208.84	10	3.03	1.52
pmed38	5	900	16200	61	61.00	61	48.84	20	0.00	0.00
pmed39	10	900	16200	74	74.00	74	8.97	20	0.00	0.00
pmed40	90	900	16200	29	29.95	30	29.55	1	3.45	3.28
AVG							**31.32s**	**19.20**	**0.23%**	**0.12%**

Tabela B.1.2. Detaljni rezultati za VNS sa jednostavnom procedurom lokalne pretrage

	P	*N*	*Best-Known Value*	*Best-Found Value*	*Time*	*#Best-Known*	*Gap (best found-best known) /best known*100*
pmed1	5	100	268	268	2.09	20	0.00
pmed2	10	100	220	220	10.19	7	0.00
pmed3	10	100	208	208	17.58	19	0.00
pmed4	20	100	163	163	43.08	18	0.00
pmed5	33	100	110	110	7.79	20	0.00
pmed6	5	200	180	180	7.68	20	0.00
pmed7	10	200	143	143	69.55	14	0.00
pmed8	20	200	122	122	87.12	4	0.00
pmed9	40	200	85	86	145.83	0	1.18
pmed10	67	200	70	70	14.69	20	0.00
pmed11	5	300	125	125	16.71	20	0.00
pmed12	10	300	112	112	109.23	8	0.00
pmed13	30	300	78	81	237.97	0	3.85
pmed14	60	300	60	65	275.21	0	8.33
pmed15	100	300	44	47	280.48	0	6.82
pmed16	5	400	98	98	67.62	20	0.00

pmed17	10	400	83	83	173.91	10	0.00
pmed18	40	400	62	68	336.47	0	9.68
pmed19	80	400	42	51	350.41	0	21.43
pmed20	133	400	40	43	372.42	0	7.50
pmed21	5	500	85	85	86.36	18	0.00
pmed22	10	500	80	80	276.32	6	0.00
pmed23	50	500	49	57	441.19	0	16.33
pmed24	100	500	35	44	461.58	0	25.71
pmed25	167	500	44	44	422.11	9	0.00
pmed26	5	600	80	80	94.07	4	0.00
pmed27	10	600	67	68	281.74	0	1.49
pmed28	60	600	57	57	201.83	20	0.00
pmed29	120	600	36	42	564.00	0	16.67
pmed30	200	600	40	40	562.31	3	0.00
pmed31	5	700	64	64	60.24	20	0.00
pmed32	10	700	72	72	20.87	20	0.00
pmed33	70	700	35	44	593.64	0	25.71
pmed34	140	700	41	41	640.64	1	0.00
pmed35	5	800	64	64	240.73	4	0.00
pmed36	10	800	58	58	480.58	1	0.00
pmed37	80	800	33	45	739.75	0	36.36
pmed38	5	900	61	61	262.09	11	0.00
pmed39	10	900	74	74	18.67	20	0.00
pmed40	90	900	29	40	802.21	0	37.93
AVG					**246.92s**	**8.43**	**5.47%**

Tabela B.1.3 Detaljni rezultati VNS algoritma za prepoznavanje egzaktnih rešenja *OR-Library* test instanci

	Best-Known Value (1)	*Best-Found Value (2)*	*AVG Value (3)*	*Worst Value (4)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(4)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(3)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$	*Exact Value*
pmed1	268	268	270.50	275	0.02	3	2.61	0.93	0.00	
pmed2	220	220	223.55	227	0.11	4	3.18	1.61	0.00	
pmed3	208	208	215.30	227	0.09	3	9.13	3.51	0.00	
pmed4	163	163	171.55	185	0.07	1	13.50	5.25	0.00	

pmed5	110	110	111.00	116	0.02	16	5.45	0.91	0.00	
pmed6	180	180	181.60	186	0.23	10	3.33	0.89	0.00	
pmed7	143	143	145.30	149	0.51	3	4.20	1.61	0.00	
pmed8	122	122	123.50	125	0.48	2	2.46	1.23	0.00	
pmed9	85	85	87.30	91	0.20	3	7.06	2.71	0.00	
pmed10	70	70	70.00	70	0.10	20	0.00	0.00	0.00	✓
pmed11	125	125	125.20	126	0.52	16	0.80	0.16	0.00	
pmed12	112	112	114.05	119	1.20	5	6.25	1.83	0.00	
pmed13	78	78	79.75	85	1.00	3	8.97	2.24	0.00	
pmed14	60	60	62.30	66	0.63	1	10.00	3.83	0.00	✓
pmed15	44	44	44.00	44	0.36	20	0.00	0.00	0.00	✓
pmed16	98	98	98.20	99	1.61	16	1.02	0.20	0.00	
pmed17	83	83	83.80	86	4.26	9	3.61	0.96	0.00	
pmed18	62	63	64.40	66	2.70	0	6.45	3.87	1.61	
pmed19	42	43	44.50	47	1.51	0	11.90	5.95	2.38	
pmed20	40	40	40.00	40	0.81	20	0.00	0.00	0.00	✓
pmed21	85	85	85.30	86	5.00	14	1.18	0.35	0.00	
pmed22	80	80	81.80	84	20.38	3	5.00	2.25	0.00	
pmed23	49	50	50.75	53	6.17	0	8.16	3.57	2.04	
pmed24	35	35	36.50	37	2.37	1	5.71	4.29	0.00	
pmed25	44	44	44.00	44	1.48	20	0.00	0.00	0.00	✓
pmed26	80	80	80.90	82	20.52	5	2.50	1.13	0.00	
pmed27	67	67	67.75	69	20.04	7	2.99	1.12	0.00	
pmed28	57	57	57.00	57	2.86	20	0.00	0.00	0.00	✓
pmed29	36	36	36.00	36	3.02	20	0.00	0.00	0.00	✓
pmed30	40	40	40.00	40	3.06	20	0.00	0.00	0.00	✓
pmed31	64	64	64.00	64	7.32	20	0.00	0.00	0.00	
pmed32	72	72	72.00	72	4.58	20	0.00	0.00	0.00	✓
pmed33	35	35	35.70	37	16.87	8	5.71	2.00	0.00	
pmed34	41	41	41.00	41	4.50	20	0.00	0.00	0.00	✓
pmed35	64	64	64.70	65	66.86	6	1.56	1.09	0.00	
pmed36	58	58	58.60	59	50.63	8	1.72	1.03	0.00	
pmed37	33	33	34.35	35	24.95	2	6.06	4.09	0.00	✓

pmed38	61	61	61.20	62	32.37	16	1.64	0.33	0.00	
pmed39	74	74	74.00	74	9.82	20	0.00	0.00	0.00	✓
pmed40	29	30	30.00	30	18.08	0	3.45	3.45	3.45	
AVG					8.43s	9.63	3.64%	1.56%	0.24%	***Total: 12***

Tabela B.1.4. Detaljni rezultati za velike test instance

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
rndkreg1	5	1000	50000	24	24.00	24	39.59	20	0.00	0.00
rndkreg2	10	1000	50000	21	21.00	21	40.60	20	0.00	0.00
rndkreg3	20	1000	50000	19	19.00	19	160.91	20	0.00	0.00
rndkreg4	33	1000	50000	17	17.00	17	74.63	20	0.00	0.00
rndkreg5	50	1000	50000	16	16.00	16	25.96	20	0.00	0.00
rndkreg6	67	1000	50000	14	14.40	15	239.74	12	7.14	2.86
rndkreg7	100	1000	50000	12	12.00	12	48.70	20	0.00	0.00
rndkreg8	133	1000	50000	11	11.00	11	23.48	20	0.00	0.00
rndkreg9	150	1000	50000	10	10.50	11	210.27	10	10.00	5.00
rndkreg10	167	1000	50000	10	10.00	10	22.62	20	0.00	0.00
rndkreg11	200	1000	50000	9	9.00	9	24.81	20	0.00	0.00
rndkreg12	5	1500	112500	18	18.00	18	316.58	20	0.00	0.00
rndkreg13	10	1500	112500	16	16.85	17	228.66	3	6.25	5.31
rndkreg14	20	1500	112500	16	16.00	16	104.59	20	0.00	0.00
rndkreg15	33	1500	112500	14	14.00	14	191.94	20	0.00	0.00
rndkreg16	50	1500	112500	13	13.00	13	99.63	20	0.00	0.00
rndkreg17	67	1500	112500	12	12.00	12	116.56	20	0.00	0.00
rndkreg18	100	1500	112500	11	11.00	11	88.23	20	0.00	0.00
rndkreg19	133	1500	112500	10	10.00	10	87.35	20	0.00	0.00
rndkreg20	150	1500	112500	9	9.35	10	560.04	13	11.11	3.89
rndkreg21	167	1500	112500	9	9.00	9	146.23	20	0.00	0.00
rndkreg22	200	1500	112500	9	9.00	9	82.11	20	0.00	0.00
rndkreg23	5	2000	200000	17	17.00	17	386.43	20	0.00	0.00
rndkreg24	10	2000	200000	16	16.00	16	211.35	20	0.00	0.00
rndkreg25	20	2000	200000	14	14.00	14	205.01	20	0.00	0.00
rndkreg26	33	2000	200000	13	13.00	13	220.43	20	0.00	0.00

rndkreg27	50	2000	200000	12	12.00	12	196.28	20	0.00	0.00
rndkreg28	67	2000	200000	11	11.00	11	229.48	20	0.00	0.00
rndkreg29	100	2000	200000	10	10.00	10	222.36	20	0.00	0.00
rndkreg30	133	2000	200000	9	9.00	9	245.29	20	0.00	0.00
rndkreg31	150	2000	200000	9	9.00	9	222.65	20	0.00	0.00
rndkreg32	167	2000	200000	9	9.00	9	223.69	20	0.00	0.00
rndkreg33	200	2000	200000	8	8.00	8	222.17	20	0.00	0.00
rndkreg34	5	2500	312500	14	14.95	15	621.97	1	7.14	6.79
rndkreg35	10	2500	312500	14	14.00	14	436.78	20	0.00	0.00
rndkreg36	20	2500	312500	12	12.00	12	489.28	20	0.00	0.00
rndkreg37	33	2500	312500	12	12.00	12	406.06	20	0.00	0.00
rndkreg38	50	2500	312500	10	10.95	11	523.85	1	10.00	9.50
rndkreg39	67	2500	312500	10	10.00	10	481.56	20	0.00	0.00
rndkreg40	100	2500	312500	9	9.05	10	1227.70	19	11.11	0.56
rndkreg41	133	2500	312500	8	8.70	9	744.28	6	12.50	8.75
rndkreg42	150	2500	312500	8	8.00	8	533.15	20	0.00	0.00
rndkreg43	167	2500	312500	8	8.00	8	459.68	20	0.00	0.00
rndkreg44	200	2500	312500	8	8.00	8	418.84	20	0.00	0.00
AVG							**269.58s**	**17.84**	**1.71%**	**0.97%**

Tabela B.1.5. Detaljni rezultati za test instance definisane nad grafovima sa različitim gustinama

	P	*N*	*Density*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
rnddnskreg1	5	500	50.10	11	11.00	11	40.54	20	0.00	0.00
rnddnskreg2	10	500	50.10	10	10.00	10	3.79	20	0.00	0.00
rnddnskreg3	50	500	50.10	7	7.00	7	3.96	20	0.00	0.00
rnddnskreg4	200	500	50.10	4	4.00	4	4.00	20	0.00	0.00
rnddnskreg5	5	500	60.12	10	10.00	10	3.61	20	0.00	0.00
rnddnskreg6	10	500	60.12	9	9.00	9	5.34	20	0.00	0.00
rnddnskreg7	50	500	60.12	6	6.00	6	3.41	20	0.00	0.00
rnddnskreg8	200	500	60.12	3	3.00	3	5.03	20	0.00	0.00
rnddnskreg9	5	500	80.16	8	8.00	8	28.35	20	0.00	0.00
rnddnskreg10	10	500	80.16	8	8.00	8	1.78	20	0.00	0.00
rnddnskreg11	50	500	80.16	5	5.90	6	36.23	2	20.00	18.00

rnddnskreg12	200	500	80.16	3	3.00	3	2.15	20	0.00	0.00
rnddnskreg13	5	600	50.08	10	10.00	10	8.98	20	0.00	0.00
rnddnskreg14	10	600	50.08	9	9.40	10	146.03	12	11.11	4.44
rnddnskreg15	50	600	50.08	6	6.05	7	108.84	19	16.67	0.83
rnddnskreg16	200	600	50.08	4	4.00	4	5.23	20	0.00	0.00
rnddnskreg17	5	600	60.10	10	10.00	10	4.61	20	0.00	0.00
rnddnskreg18	10	600	60.10	8	8.00	8	9.68	20	0.00	0.00
rnddnskreg19	50	600	60.10	6	6.00	6	5.84	20	0.00	0.00
rnddnskreg20	200	600	60.10	4	4.00	4	5.65	20	0.00	0.00
rnddnskreg21	5	600	80.13	8	8.00	8	7.14	20	0.00	0.00
rnddnskreg22	10	600	80.13	7	7.75	8	79.87	5	14.29	10.71
rnddnskreg23	50	600	80.13	5	5.20	6	109.07	16	20.00	4.00
rnddnskreg24	200	600	80.13	3	3.00	3	8.01	20	0.00	0.00
rnddnskreg25	5	800	50.06	9	9.30	10	229.69	14	11.11	3.33
rnddnskreg26	10	800	50.06	8	8.00	8	124.15	20	0.00	0.00
rnddnskreg27	50	800	50.06	6	6.00	6	18.14	20	0.00	0.00
rnddnskreg28	200	800	50.06	4	4.00	4	16.23	20	0.00	0.00
rnddnskreg29	5	800	60.08	8	8.00	8	104.69	20	0.00	0.00
rnddnskreg30	10	800	60.08	8	8.00	8	15.45	20	0.00	0.00
rnddnskreg31	50	800	60.08	6	6.00	6	15.05	20	0.00	0.00
rnddnskreg32	200	800	60.08	4	4.00	4	13.76	20	0.00	0.00
rnddnskreg33	5	800	80.10	8	8.00	8	14.17	20	0.00	0.00
rnddnskreg34	10	800	80.10	7	7.00	7	30.23	20	0.00	0.00
rnddnskreg35	50	800	80.10	5	5.00	5	55.29	20	0.00	0.00
rnddnskreg36	200	800	80.10	4	4.00	4	15.91	20	0.00	0.00
rnddnskreg37	5	1000	50.05	8	8.25	9	314.63	15	12.50	3.13
rnddnskreg38	10	1000	50.05	8	8.00	8	37.44	20	0.00	0.00
rnddnskreg39	50	1000	50.05	6	6.00	6	32.43	20	0.00	0.00
rnddnskreg40	200	1000	50.05	4	4.00	4	36.88	20	0.00	0.00
rnddnskreg41	5	1000	60.06	8	8.00	8	38.71	20	0.00	0.00
rnddnskreg42	10	1000	60.06	8	8.00	8	27.15	20	0.00	0.00
rnddnskreg43	50	1000	60.06	6	6.00	6	15.08	20	0.00	0.00
rnddnskreg44	200	1000	60.06	4	4.00	4	30.11	20	0.00	0.00

rnddnskreg45	5	1000	80.08	7	7.85	8	92.01	3	14.29	12.14
rnddnskreg46	10	1000	80.08	6	6.00	6	72.49	20	0.00	0.00
rnddnskreg47	50	1000	80.08	5	5.00	5	56.53	20	0.00	0.00
rnddnskreg48	200	1000	80.08	4	4.00	4	24.22	20	0.00	0.00
AVG			63.43%				**43.07s**	**18.46**	**2.50%**	**1.18%**

Dodatak B.2

Da bi se uporedila rešenja i problemi *p*-centra (pCP), *p*-sledećeg centra (pNCP) i *p*-medijana (pMP) sa problemom *p*-drugog centra (pSCP), u Dodatku B.2 predstavljeni su rezultati testiranja VNS algoritama za pomenute probleme. Takođe, za svako dobijeno rešenje izračunata je vrednost funkcije cilja za pSCP i određeno odstupanje od najboljeg poznatog rešenja, tj. rešenja dobijenog VNS algoritmom za pSCP. Problem *p*-centra je optimizacioni problem identifikacije lokacija *p* od potencijalnih *n* centara na takav način da se minimizuje najveće rastojanje između korisnika i njemu dodeljenog, tj. najbližeg centra. Problem *p*-sledećeg centra je definisan kao lociranje *p* od mogućih *n* centara u cilju minimizacije maksimalne sume rastojanja korisnika do najbližeg centra i rastojanja tog centra do njemu najbližeg centra. Problem *p*-medijana je problem identifikacije *p* od potencijalnih *n* lokacija za postavljanje centara na takav način da se minimizuje suma rastojanja između svih korisnika i njima najbližih centara. Problem *p*-drugog centra predstavlja identifikuju *p* od potencijalnih *n* lokacija centara u cilju minimizacije maksimalne sume rastojanja između korisnika i njemu najbližeg i drugog-najbližeg centra.

Za potrebe poređenja preuzeti su rezultati VNS algoritama iz radova "Mladenovic, N., Labbe, M., Hansen, P.: *Solving the p-center problem with tabu search and variable neighborhood search*", "Ristic, D., Mladenovic, N., Todosijevic, R., Urosevic, D.: *Filtered variable neighborhood search method for the p-next center problem*" i "Hansen, P., Mladenovic, N.: *Variable neighborhood search for the p-median*". Sva testiranja su izvršena nad *OR-Library* test primerima na istoj hardverskoj konfiguraciji (*Intel Core i7-8700K (3.7GHz) CPU* sa *32GB RAM*-a), tako da su rezultati uporedivi. Dobijeni rezultati su predstavljeni u narednim tabelama.

Tabele B.2.1 - B.2.3 prikazuju rešenja za svaku izvornu instancu *OR-Library* test biblioteke za *p*-centar, *p*-sledeći centar i *p*-medijan probleme u poređenju sa problemom *p*-drugog centra. Prva kolona tabele sadrži naziv instance, naredne tri kolone predstavljaju vrednost *p* (broj centara), *n* (broj korisnika) i "*Best-Known pSCP Value*", tj. vrednost najboljeg rešenja pronađenog algoritmom za pSCP. U koloni "*pCP/pNCP/pMP Value*" prikazana je vrednost dobijenog rešenja algoritma za konkretan problem, a u "*pSCP Value*" vrednost dobijenog rešenja primenjenog na *p*-drugi centar problem. Kolona "*Time*" (ili *time-to-target*) prikazuje vreme u sekundama koje je bilo potrebno da se prvi put pronađe najbolje rešenje. Konačno, u poslednjoj koloni, dato je procentualno odstupanje pSCP vrednosti pronađenog rešenja u odnosu na vrednost najboljeg poznatog pSCP rešenja. Procenat odstupanja (*eng. percentage gap*) se računa kao $\frac{pSCP\ value - Best\ known\ pSCP\ value}{Best\ known\ pSCP\ value} * 100$.

Tabela B.2.1. Rezultati poređenja sa VNS algoritmom za *p*-centar problem

	P	*N*	*Best-Known pSCP Value (1)*	*pCP Value*	*pSCP Value (2)*	*Time*	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	5	100	268	127	271	0.02	1.12
pmed2	10	100	220	98	263	0.45	19.55
pmed3	10	100	208	93	248	0.23	19.23
pmed4	20	100	163	74	220	0.12	34.97

pmed5	33	100	110	48	147	0.02	33.64
pmed6	5	200	180	84	182	0.32	1.11
pmed7	10	200	143	64	154	0.24	7.69
pmed8	20	200	122	55	153	0.92	25.41
pmed9	40	200	85	37	119	0.18	40.00
pmed10	67	200	70	20	77	0.15	10.00
pmed11	5	300	125	59	138	0.67	10.40
pmed12	10	300	112	51	128	1.03	14.29
pmed13	30	300	78	36	99	9.65	26.92
pmed14	60	300	60	26	85	1.02	41.67
pmed15	100	300	44	18	55	0.39	25.00
pmed16	5	400	98	47	102	0.72	4.08
pmed17	10	400	83	39	105	1.87	26.51
pmed18	40	400	62	28	75	28.60	20.97
pmed19	80	400	42	18	49	45.85	16.67
pmed20	133	400	40	13	44	4.86	10.00
pmed21	5	500	85	40	94	1.76	10.59
pmed22	10	500	80	38	94	153.49	17.50
pmed23	50	500	49	22	58	71.28	18.37
pmed24	100	500	35	15	47	6.28	34.29
pmed25	167	500	44	11	48	2.70	9.09
pmed26	5	600	80	38	86	3.47	7.50
pmed27	10	600	67	32	70	5.53	4.48
pmed28	60	600	57	18	61	117.47	7.02
pmed29	120	600	36	13	47	5.55	30.56
pmed30	200	600	40	9	40	33.63	0.00
pmed31	5	700	64	30	68	4.73	6.25
pmed32	10	700	72	29	78	140.07	8.33
pmed33	70	700	35	16	42	20.90	20.00
pmed34	140	700	41	11	49	15.37	19.51
pmed35	5	800	64	30	73	12.67	14.06
pmed36	10	800	58	27	61	239.01	5.17
pmed37	80	800	33	15	43	160.43	30.30

pmed38	5	900	61	29	69	10.40	13.11
pmed39	10	900	74	23	86	71.29	16.22
pmed40	90	900	29	14	37	16.66	27.59
AVG						**29.75s**	**17.23%**

Rezultati algoritma za problem *p*-centra dati u Tabeli B.2.1, potvrđuju jedinstvenost problema *p*-drugog centra. Primećuje se da se identifikovana pSCP vrednost gotovo svih rešenja razlikuje od najbolje poznate vrednosti. Samo u 1 od 40 slučajeva (pmed30) je došlo do poklapanja ovih vrednosti. Prosečno procentualno odstupanje vrednosti pSCP rešenja od najboljeg poznatog rešenja, dato u poslednjoj koloni tabele, je značajno, 17.23%. Što se tiče vremena potrebnog da se pronađe najbolje rešenje, algoritmi su približno jednaki. Predloženom algoritmu za rešavanje pSCP-a je bilo potrebno neznatno više vremena, 31.32 sekunde u poređenju sa 29.75 sekundi koliko je bilo potrebno algoritmu za pCP.

Tabela B.2.2. Rezultati poređenja sa VNS algoritmom za *p*-sledeći centar problem

	P	*N*	*Best-Known pSCP Value (1)*	*pNCP Value*	*pSCP Value (2)*	*Time*	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	5	100	268	166	304	0.50	13.43
pmed2	10	100	220	135	257	100.46	16.82
pmed3	10	100	208	151	294	87.24	41.35
pmed4	20	100	163	118	211	177.84	29.45
pmed5	33	100	110	85	164	38.63	49.09
pmed6	5	200	180	107	207	146.09	15.00
pmed7	10	200	143	84	162	390.52	13.29
pmed8	20	200	122	84	160	332.01	31.15
pmed9	40	200	85	71	135	261.44	58.82
pmed10	67	200	70	70	102	9.74	45.71
pmed11	5	300	125	70	137	168.67	9.60
pmed12	10	300	112	72	139	544.26	24.11
pmed13	30	300	78	52	101	658.49	29.49
pmed14	60	300	60	60	103	515.14	71.67
pmed15	100	300	44	44	80	811.39	81.82
pmed16	5	400	98	55	109	106.52	11.22
pmed17	10	400	83	47	92	791.43	10.84
pmed18	40	400	62	50	98	1101.69	58.06
pmed19	80	400	42	40	71	1113.12	69.05
pmed20	133	400	40	40	76	1079.11	90.00
pmed21	5	500	85	48	90	377.36	5.88

pmed22	10	500	80	52	96	849.43	20.00
pmed23	50	500	49	42	78	1371.95	59.18
pmed24	100	500	35	35	64	1196.55	82.86
pmed25	167	500	44	44	82	242.45	86.36
pmed26	5	600	80	47	93	905.53	16.25
pmed27	10	600	67	40	79	946.39	17.91
pmed28	60	600	57	57	76	20.40	33.33
pmed29	120	600	36	36	71	1461.31	97.22
pmed30	200	600	40	40	68	108.86	70.00
pmed31	5	700	64	35	69	1057.44	7.81
pmed32	10	700	72	72	84	15.29	16.67
pmed33	70	700	35	33	61	1611.42	74.29
pmed34	140	700	41	41	64	72.14	56.10
pmed35	5	800	64	36	70	927.56	9.38
pmed36	10	800	58	42	74	223.52	27.59
pmed37	80	800	33	33	57	1634.93	72.73
pmed38	5	900	61	40	73	1102.41	19.67
pmed39	10	900	74	74	74	32.70	0.00
pmed40	90	900	29	29	50	1394.52	72.41
AVG						**599.66s**	**40.39%**

Tabela B.2.2 prikazuje rezultate algoritma za problem *p*-sledećeg centra, kao i poređenje pSCP vrednosti dobijenih rešenja sa najboljim poznatim vrednostima za problem *p*-drugog centra. U koloni "*Gap*" se vidi da su, sem za pmed39, ove vrednosti različite za sve ostale *OR-Library* instance. U proseku, razlika je 40.39%. Sa druge strane, algoritam za problem *p*-drugog centra je znatno brži u pogledu vremena pronalaska najboljeg rešenja. Prosečno, nad kompletnim test skupom, vreme pNCP algoritma je 599.66 sekundi, dok je pSCP algoritmu potrebno 31.32 sekunde.

Tabela B.2.3. Rezultati poređenja sa VNS algoritmom za *p*-medijan problem

	P	*N*	*Best-Known pSCP Value (1)*	*pMP Value*	*pSCP Value (2)*	*Time*	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	5	100	268	5819	271	0.02	1.12
pmed2	10	100	220	4093	308	0.03	40.00
pmed3	10	100	208	4250	388	0.05	86.54
pmed4	20	100	163	3034	220	0.17	34.97
pmed5	33	100	110	1355	134	0.06	21.82
pmed6	5	200	180	7824	210	0.12	16.67

pmed7	10	200	143	5631	163	1.90	13.99
pmed8	20	200	122	4445	219	1.65	79.51
pmed9	40	200	85	2734	127	5.07	49.41
pmed10	67	200	70	1255	77	1.09	10.00
pmed11	5	300	125	7696	146	0.40	16.80
pmed12	10	300	112	6634	191	0.47	70.54
pmed13	30	300	78	4374	116	7.03	48.72
pmed14	60	300	60	2968	89	13.13	48.33
pmed15	100	300	44	1729	55	98.02	25.00
pmed16	5	400	98	8162	107	1.05	9.18
pmed17	10	400	83	6999	102	2.66	22.89
pmed18	40	400	62	4809	114	74.99	83.87
pmed19	80	400	42	2845	65	122.41	54.76
pmed20	133	400	40	1789	48	51.27	20.00
pmed21	5	500	85	9138	96	1.64	12.94
pmed22	10	500	80	8579	139	14.82	73.75
pmed23	50	500	49	4619	76	340.85	55.10
pmed24	100	500	35	2961	58	279.36	65.71
pmed25	167	500	44	1828	48	246.11	9.09
pmed26	5	600	80	9917	97	20.87	21.25
pmed27	10	600	67	8307	102	121.17	52.24
pmed28	60	600	57	4498	69	380.00	21.05
pmed29	120	600	36	3033	53	487.84	47.22
pmed30	200	600	40	1989	40	380.08	0.00
pmed31	5	700	64	10086	78	33.99	21.88
pmed32	10	700	72	9297	176	12.95	144.44
pmed33	70	700	35	4700	67	602.43	91.43
pmed34	140	700	41	3013	45	437.49	9.76
pmed35	5	800	64	10400	75	32.85	17.19
pmed36	10	800	58	9934	106	249.97	82.76
pmed37	80	800	33	5057	59	571.88	78.79
pmed38	5	900	61	11060	96	9.93	57.38
pmed39	10	900	74	9423	173	22.52	133.78

pmed40	90	900	29	5128	52	594.08	79.31
AVG						**130.56s**	**45.73%**

Poređenje sa algoritmom za *p*-medijan problem je predstavljeno Tabelom B.2.3. Ponovo, pSCP vrednosti dobijenih rešenja se razlikuju od najboljih poznatih pSCP vrednosti u svim primerima, sem za pmed30. Prosečna razlika nad kompletnim test skupom je 45.73%. Sa druge strane, vreme do pronalaska najboljeg rešenja je u proseku oko četiri puta veće kod algoritma za pMP, 130.56 sekundi.

Rezultati testiranja dati u prethodnim tabelama potvrđuju jedinstvenost *p*-centar, *p*-sledeći centar, *p*-medijan i *p*-drugi centar problema. Očigledno je da se rešenja ovih problema međusobno razlikuju. Sa druge strane, sem za pNCP, prosečna vremena pronalazaka najboljih rešenja su srazmerno jednaka. Samo u slučaju algoritma za pNCP prosečno vreme je znatno veće.

Dodatak C

Dodatak C sadrži detaljne rezultate dobijene tokom testiranja predloženog VNS algoritma za probleme *p*-*α*-sledećih (pαNCP) i *p*-*α*-najbližih (pαCCP) centara.

Tabela C.1. Detaljni rezultati za pαNCP algoritam ($\alpha = p$, $t_{max} = \alpha n$) testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1	5	100	200	214	214.00	214	0.78	20.00	0.00	0.00
pmed2	10	100	200	251	251.00	251	8.79	20.00	0.00	0.00
pmed3	10	100	200	313	313.00	313	18.07	20.00	0.00	0.00
pmed4	20	100	200	575	575.00	575	575.62	20.00	0.00	0.00
pmed5	33	100	200	642	664.70	679	881.63	2.00	5.76	3.54
pmed6	5	200	800	130	130.00	130	6.30	20.00	0.00	0.00
pmed7	10	200	800	139	139.00	139	193.98	20.00	0.00	0.00
pmed8	20	200	800	257	259.85	286	1839.93	5.00	11.28	1.11
pmed9	40	200	800	423	456.95	510	3643.33	1.00	20.57	8.03
pmed10	67	200	800	570	608.80	681	9579.85	1.00	19.47	6.81
pmed11	5	300	1800	83	83.00	83	6.78	20.00	0.00	0.00
pmed12	10	300	1800	131	133.05	134	626.55	1.00	2.29	1.56
pmed13	30	300	1800	218	234.40	315	4060.08	2.00	44.50	7.52
pmed14	60	300	1800	456	503.80	587	12739.08	1.00	28.73	10.48
pmed15	100	300	1800	811	858.80	949	24820.11	1.00	17.02	5.89
pmed16	5	400	3200	67	67.35	68	488.92	13.00	1.49	0.52
pmed17	10	400	3200	83	84.85	93	1171.25	15.00	12.05	2.23
pmed18	40	400	3200	227	272.45	395	9385.67	1.00	74.01	20.02
pmed19	80	400	3200	477	547.55	641	23558.33	1.00	34.38	14.79
pmed20	133	400	3200	1031	1148.10	1295	47275.12	1.00	25.61	11.36
pmed21	5	500	5000	59	59.00	59	159.00	20.00	0.00	0.00
pmed22	10	500	5000	88	89.75	95	867.54	15.00	7.95	1.99
pmed23	50	500	5000	246	273.20	303	18638.96	1.00	23.17	11.06
pmed24	100	500	5000	568	647.95	765	45016.45	1.00	34.68	14.08
pmed25	167	500	5000	1046	1222.00	1354	97020.31	1.00	29.45	16.83
pmed26	5	600	7200	55	55.50	56	102.61	10.00	1.82	0.91
pmed27	10	600	7200	67	67.90	70	1127.50	14.00	4.48	1.34

pmed28	60	600	7200	246	292.70	361	26754.36	1.00	46.75	18.98
pmed29	120	600	7200	608	747.50	874	77169.14	1.00	43.75	22.94
pmed30	200	600	7200	1364	1557.85	1779	179308.10	1.00	30.43	14.21
pmed31	5	700	9800	45	45.00	45	759.51	20.00	0.00	0.00
pmed32	10	700	9800	94	95.25	99	1878.64	13.00	5.32	1.33
pmed33	70	700	9800	274	330.90	422	39902.22	1.00	54.01	20.77
pmed34	140	700	9800	733	883.35	1070	122855.70	1.00	45.98	20.51
pmed35	5	800	12800	43	43.90	57	880.48	17.00	32.56	2.09
pmed36	10	800	12800	69	69.80	71	2168.30	11.00	2.90	1.16
pmed37	80	800	12800	346	416.30	490	57421.27	1.00	41.62	20.32
pmed38	5	900	16200	46	46.00	46	32.13	20.00	0.00	0.00
pmed39	10	900	16200	94	94.10	96	904.36	19.00	2.13	0.11
pmed40	90	900	16200	375	448.70	572	83746.41	1.00	52.53	19.65
AVG							**22439.83s**	**8.85**	**18.92%**	**7.05%**

Tabela C.2. Detaljni rezultati za pαCCP algoritam ($\alpha = p$, $t_{max} = \alpha n$) testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1	5	100	200	834	834.00	834	0.56	20	0.00	0.00
pmed2	10	100	200	1618	1618.00	1618	30.37	20	0.00	0.00
pmed3	10	100	200	1983	1983.00	1983	12.73	20	0.00	0.00
pmed4	20	100	200	3645	3645.00	3645	220.09	20	0.00	0.00
pmed5	33	100	200	5459	5460.90	5463	990.49	5	0.07	0.03
pmed6	5	200	800	542	542.00	542	1.97	20	0.00	0.00
pmed7	10	200	800	981	981.00	981	42.41	20	0.00	0.00
pmed8	20	200	800	2323	2323.45	2326	727.00	17	0.13	0.02
pmed9	40	200	800	4616	4620.05	4628	3927.02	3	0.26	0.09
pmed10	67	200	800	6841	6841.00	6841	84.80	20	0.00	0.00
pmed11	5	300	1800	355	355.00	355	3.78	20	0.00	0.00
pmed12	10	300	1800	858	858.00	858	744.76	20	0.00	0.00
pmed13	30	300	1800	2396	2398.15	2403	6193.63	4	0.29	0.09
pmed14	60	300	1800	5938	5938.10	5940	5013.34	19	0.03	0.00
pmed15	100	300	1800	7881	7891.95	7909	15206.19	2	0.36	0.14
pmed16	5	400	3200	284	284.00	284	26.88	20	0.00	0.00

pmed17	10	400	3200	552	554.20	563	174.55	16	1.99	0.40
pmed18	40	400	3200	2881	2883.15	2887	10317.44	3	0.21	0.07
pmed19	80	400	3200	4638	4657.05	4683	21736.74	1	0.97	0.41
pmed20	133	400	3200	8964	8964.80	8968	31451.40	9	0.04	0.01
pmed21	5	500	5000	250	250.00	250	219.40	20	0.00	0.00
pmed22	10	500	5000	610	610.50	613	1008.63	12	0.49	0.08
pmed23	50	500	5000	2656	2670.65	2692	17957.98	1	1.36	0.55
pmed24	100	500	5000	5835	5843.80	5855	39613.49	1	0.34	0.15
pmed25	167	500	5000	10128	10136.00	10154	66651.67	2	0.26	0.08
pmed26	5	600	7200	235	235.00	235	41.17	20	0.00	0.00
pmed27	10	600	7200	474	479.30	488	2111.27	8	2.95	1.12
pmed28	60	600	7200	4027	4027.00	4027	2748.69	20	0.00	0.00
pmed29	120	600	7200	6081	6085.10	6092	55257.29	2	0.18	0.07
pmed30	200	600	7200	12735	12735.00	12735	70673.27	20	0.00	0.00
pmed31	5	700	9800	185	185.00	185	31.02	20	0.00	0.00
pmed32	10	700	9800	697	697.00	697	16.10	20	0.00	0.00
pmed33	70	700	9800	2961	2973.85	2982	38922.15	1	0.71	0.43
pmed34	140	700	9800	8503	8503.00	8503	50381.25	20	0.00	0.00
pmed35	5	800	12800	186	186.00	186	21.02	20	0.00	0.00
pmed36	10	800	12800	448	448.00	448	640.61	20	0.00	0.00
pmed37	80	800	12800	3642	3642.00	3642	14760.79	20	0.00	0.00
pmed38	5	900	16200	212	215.25	217	1434.04	7	2.36	1.53
pmed39	10	900	16200	722	722.00	722	32.66	20	0.00	0.00
pmed40	90	900	16200	3342	3344.75	3350	62194.15	2	0.24	0.08
AVG							**13040.57s**	**13.38**	**0.33%**	**0.13%**

Tabela C.3. Detaljni rezultati za pαNCP algoritam ($\alpha = 2$, $t_{max} = 2n$) testiran nad *OR-Library* test instancama

	Best-Known Value (1)	*Best-Found Value (2)*	*AVG Value (3)*	*Worst Value (4)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(4)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(3)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	166	166	166.00	166	0.18	20	0.00	0.00	0.00
pmed2	135	135	135.00	135	6.06	20	0.00	0.00	0.00
pmed3	151	151	151.00	151	10.13	20	0.00	0.00	0.00
pmed4	118	119	119.00	119	26.17	0	0.85	0.85	0.85

pmed5	85	85	85.00	85	2.54	20	0.00	0.00	0.00
pmed6	107	107	107.00	107	10.03	20	0.00	0.00	0.00
pmed7	84	84	84.10	85	132.07	18	1.19	0.12	0.00
pmed8	81	81	82.55	84	195.76	2	3.70	1.91	0.00
pmed9	71	71	71.00	71	70.48	20	0.00	0.00	0.00
pmed10	70	70	70.00	70	1.65	20	0.00	0.00	0.00
pmed11	70	70	70.00	70	4.09	20	0.00	0.00	0.00
pmed12	72	72	72.00	72	9.92	20	0.00	0.00	0.00
pmed13	47	48	51.10	54	404.03	0	14.89	8.72	2.13
pmed14	60	60	60.00	60	115.04	20	0.00	0.00	0.00
pmed15	44	44	44.35	51	177.19	19	15.91	0.80	0.00
pmed16	54	54	54.05	55	125.72	19	1.85	0.09	0.00
pmed17	46	46	47.00	48	263.28	1	4.35	2.17	0.00
pmed18	50	50	50.00	50	222.19	20	0.00	0.00	0.00
pmed19	32	36	39.25	43	623.40	0	34.38	22.66	12.50
pmed20	40	40	40.10	42	359.01	19	5.00	0.25	0.00
pmed21	48	48	48.05	49	69.51	19	2.08	0.10	0.00
pmed22	49	49	49.60	51	309.60	10	4.08	1.22	0.00
pmed23	32	37	39.30	45	700.35	0	40.63	22.81	15.63
pmed24	33	33	35.00	38	605.55	4	15.15	6.06	0.00
pmed25	44	44	44.00	44	27.40	20	0.00	0.00	0.00
pmed26	47	47	47.00	47	101.91	20	0.00	0.00	0.00
pmed27	38	39	39.30	40	344.50	0	5.26	3.42	2.63
pmed28	57	57	57.00	57	9.10	20	0.00	0.00	0.00
pmed29	36	36	36.00	36	384.16	20	0.00	0.00	0.00
pmed30	40	40	40.00	40	25.18	20	0.00	0.00	0.00
pmed31	35	35	35.00	35	51.93	20	0.00	0.00	0.00
pmed32	72	72	72.00	72	5.22	20	0.00	0.00	0.00
pmed33	22	31	32.60	35	814.66	0	59.09	48.18	40.91
pmed34	41	41	41.00	41	26.31	20	0.00	0.00	0.00
pmed35	36	36	36.00	36	65.23	20	0.00	0.00	0.00
pmed36	42	42	42.00	42	25.11	20	0.00	0.00	0.00
pmed37	33	33	33.40	37	678.37	16	12.12	1.21	0.00

pmed38	40	40	40.00	40	19.70	20	0.00	0.00	0.00
pmed39	74	74	74.00	74	11.42	20	0.00	0.00	0.00
pmed40	23	27	30.10	33	967.70	0	43.48	30.87	17.39
AVG					**200.05s**	**14.68**	**6.60%**	**3.79%**	**2.30%**

Tabela C.4. Detaljni rezultati za pαNCP algoritam ($\alpha = 2$, $t_{max} = 5n$) testiran nad *OR-Library* test instancama

	Best-Known Value (1)	*Best-Found Value (2)*	*AVG Value (3)*	*Worst Value (4)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(4)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(3)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	166	166	166.00	166	4.27	20	0.00	0.00	0.00
pmed2	135	135	135.00	135	34.07	20	0.00	0.00	0.00
pmed3	151	151	151.60	157	122.68	18	3.97	0.40	0.00
pmed4	118	118	118.50	122	176.66	16	3.39	0.42	0.00
pmed5	85	85	85.00	85	1.08	20	0.00	0.00	0.00
pmed6	107	107	107.00	107	117.66	20	0.00	0.00	0.00
pmed7	84	84	85.55	88	321.86	4	4.76	1.85	0.00
pmed8	81	81	82.55	85	187.27	3	4.94	1.91	0.00
pmed9	71	71	71.00	71	141.54	20	0.00	0.00	0.00
pmed10	70	70	70.00	70	0.47	20	0.00	0.00	0.00
pmed11	70	70	70.00	70	10.24	20	0.00	0.00	0.00
pmed12	72	72	72.00	72	17.88	20	0.00	0.00	0.00
pmed13	47	47	48.05	51	782.73	7	8.51	2.23	0.00
pmed14	60	60	60.00	60	9.07	20	0.00	0.00	0.00
pmed15	44	44	44.00	44	14.53	20	0.00	0.00	0.00
pmed16	54	54	54.70	55	188.06	6	1.85	1.30	0.00
pmed17	46	47	47.65	49	291.46	0	6.52	3.59	2.17
pmed18	50	50	50.00	50	15.01	20	0.00	0.00	0.00
pmed19	32	32	33.40	36	449.91	13	12.50	4.38	0.00
pmed20	40	40	40.00	40	38.95	20	0.00	0.00	0.00
pmed21	48	48	48.70	51	136.79	9	6.25	1.46	0.00
pmed22	49	49	50.45	52	594.38	5	6.12	2.96	0.00
pmed23	32	33	34.75	38	760.58	0	18.75	8.59	3.13
pmed24	33	33	33.00	33	73.06	20	0.00	0.00	0.00
pmed25	44	44	44.00	44	5.72	20	0.00	0.00	0.00

pmed26	47	47	47.30	48	298.07	14	2.13	0.64	0.00
pmed27	38	38	39.15	40	869.88	1	5.26	3.03	0.00
pmed28	57	57	57.00	57	3.02	20	0.00	0.00	0.00
pmed29	36	36	36.00	36	14.66	20	0.00	0.00	0.00
pmed30	40	40	40.00	40	5.73	20	0.00	0.00	0.00
pmed31	35	35	35.30	38	380.93	18	8.57	0.86	0.00
pmed32	72	72	72.00	72	4.57	20	0.00	0.00	0.00
pmed33	22	25	26.10	28	1690.01	0	27.27	18.64	13.64
pmed34	41	41	41.00	41	5.14	20	0.00	0.00	0.00
pmed35	36	36	36.05	37	303.66	19	2.78	0.14	0.00
pmed36	42	42	42.00	42	8.18	20	0.00	0.00	0.00
pmed37	33	33	33.00	33	33.19	20	0.00	0.00	0.00
pmed38	40	40	40.00	40	10.46	20	0.00	0.00	0.00
pmed39	74	74	74.00	74	9.81	20	0.00	0.00	0.00
pmed40	23	23	23.35	25	2025.31	14	8.70	1.52	0.00
AVG					**253.96s**	**15.18**	**3.31%**	**1.35%**	**0.47%**

Tabela C.5. Detaljni rezultati za pαCCP algoritam ($\alpha = 2$, $t_{max} = 2n$) testiran nad *OR-Library* test instancama

	Best-Known Value (1)	*Best-Found Value (2)*	*AVG Value (3)*	*Worst Value (4)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(4)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(3)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	268	268	268.00	268	0.64	20	0.00	0.00	0.00
pmed2	220	220	220.20	221	17.03	16	0.45	0.09	0.00
pmed3	208	208	208.00	208	9.36	20	0.00	0.00	0.00
pmed4	163	163	163.00	163	6.12	20	0.00	0.00	0.00
pmed5	110	110	110.00	110	0.33	20	0.00	0.00	0.00
pmed6	180	180	180.00	180	1.01	20	0.00	0.00	0.00
pmed7	143	143	143.20	144	40.17	16	0.70	0.14	0.00
pmed8	122	122	122.00	122	68.61	20	0.00	0.00	0.00
pmed9	85	85	85.00	85	20.73	20	0.00	0.00	0.00
pmed10	70	70	70.00	70	0.43	20	0.00	0.00	0.00
pmed11	125	125	125.00	125	1.12	20	0.00	0.00	0.00
pmed12	112	112	112.00	112	21.90	20	0.00	0.00	0.00
pmed13	78	78	78.00	78	73.09	20	0.00	0.00	0.00
pmed14	60	60	60.00	60	137.51	20	0.00	0.00	0.00

pmed15	44	44	44.00	44	8.41	20	0.00	0.00	0.00
pmed16	98	98	98.00	98	4.11	20	0.00	0.00	0.00
pmed17	83	83	83.00	83	58.34	20	0.00	0.00	0.00
pmed18	62	63	63.60	64	217.95	0	3.23	2.58	1.61
pmed19	42	42	42.75	43	275.33	5	2.38	1.79	0.00
pmed20	40	40	40.00	40	8.86	20	0.00	0.00	0.00
pmed21	85	85	85.00	85	24.63	20	0.00	0.00	0.00
pmed22	80	80	80.05	81	202.06	19	1.25	0.06	0.00
pmed23	49	50	50.05	51	458.16	0	4.08	2.14	2.04
pmed24	35	35	35.70	36	422.82	6	2.86	2.00	0.00
pmed25	44	44	44.00	44	4.86	20	0.00	0.00	0.00
pmed26	80	80	80.00	80	350.84	20	0.00	0.00	0.00
pmed27	67	67	67.10	68	283.32	18	1.49	0.15	0.00
pmed28	57	57	57.00	57	5.45	20	0.00	0.00	0.00
pmed29	36	36	36.00	36	25.06	20	0.00	0.00	0.00
pmed30	40	40	40.00	40	7.27	20	0.00	0.00	0.00
pmed31	64	64	64.00	64	11.05	20	0.00	0.00	0.00
pmed32	72	72	72.00	72	4.91	20	0.00	0.00	0.00
pmed33	35	35	36.05	37	553.88	1	5.71	3.00	0.00
pmed34	41	41	41.00	41	11.73	20	0.00	0.00	0.00
pmed35	64	64	64.00	64	224.52	20	0.00	0.00	0.00
pmed36	58	58	58.15	59	371.30	17	1.72	0.26	0.00
pmed37	33	35	35.15	36	498.33	0	9.09	6.52	6.06
pmed38	61	61	61.00	61	119.94	20	0.00	0.00	0.00
pmed39	74	74	74.00	74	9.85	20	0.00	0.00	0.00
pmed40	29	30	30.95	31	421.08	0	6.90	6.72	3.45
AVG					**124.55s**	**16.45**	**1.00%**	**0.64%**	**0.33%**

Dodatak D

Dodatak D sadrži detaljne rezultate dobijene tokom testiranja predloženih VNS algoritama za *p*-sledeći medijan (pNMP) i *p*-*α*-sledeći medijan (pαNMP) probleme.

Tabela D.1. Detaljni rezultati za pNMP algoritam testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1	5	100	200	8032	8032.00	8032	2.42	20	0.00	0.00
pmed2	10	100	200	5948	5948.00	5948	11.77	20	0.00	0.00
pmed3	10	100	200	6465	6465.00	6465	11.32	20	0.00	0.00
pmed4	20	100	200	5222	5222.00	5222	11.91	20	0.00	0.00
pmed5	33	100	200	2964	2964.00	2964	11.09	20	0.00	0.00
pmed6	5	200	800	10473	10523.40	10995	47.34	7	4.98	0.48
pmed7	10	200	800	7425	7425.00	7425	120.90	20	0.00	0.00
pmed8	20	200	800	6559	6559.00	6559	465.77	20	0.00	0.00
pmed9	40	200	800	4691	4691.00	4691	312.43	20	0.00	0.00
pmed10	67	200	800	2997	2997.45	3000	112.02	17	0.10	0.02
pmed11	5	300	1800	9404	9404.00	9404	95.50	20	0.00	0.00
pmed12	10	300	1800	8555	8555.00	8555	320.19	20	0.00	0.00
pmed13	30	300	1800	6232	6232.00	6232	1263.01	20	0.00	0.00
pmed14	60	300	1800	5153	5154.95	5179	8381.73	13	0.50	0.04
pmed15	100	300	1800	3913	3913.95	3918	1805.80	15	0.13	0.02
pmed16	5	400	3200	10240	10240.00	10240	159.16	20	0.00	0.00
pmed17	10	400	3200	9044	9050.70	9178	734.54	19	1.48	0.07
pmed18	40	400	3200	6907	6918.50	6925	7601.10	1	0.26	0.17
pmed19	80	400	3200	4953	4956.95	4979	13123.56	8	0.52	0.08
pmed20	133	400	3200	4216	4217.20	4222	6510.65	15	0.14	0.03
pmed21	5	500	5000	11597	11597.00	11597	380.50	20	0.00	0.00
pmed22	10	500	5000	10657	10657.15	10660	1923.62	19	0.03	0.00
pmed23	50	500	5000	6710	6711.10	6712	6802.13	9	0.03	0.02
pmed24	100	500	5000	5104	5106.60	5113	11995.51	6	0.18	0.05
pmed25	167	500	5000	4324	4327.30	4334	21293.53	5	0.23	0.08
pmed26	5	600	7200	12130	12167.25	12279	533.23	15	1.23	0.31
pmed27	10	600	7200	10000	10001.65	10011	2115.68	17	0.11	0.02

pmed28	60	600	7200	6461	6466.70	6484	21741.65	4	0.36	0.09
pmed29	120	600	7200	5271	5274.95	5287	40481.98	5	0.30	0.07
pmed30	200	600	7200	4616	4616.00	4616	17574.59	20	0.00	0.00
pmed31	5	700	9800	12430	12430.00	12430	538.64	20	0.00	0.00
pmed32	10	700	9800	11676	11699.70	11786	2480.92	15	0.94	0.20
pmed33	70	700	9800	6959	6965.15	6993	34167.47	8	0.49	0.09
pmed34	140	700	9800	5290	5291.40	5297	65529.24	10	0.13	0.03
pmed35	5	800	12800	12982	13027.20	13267	1044.86	16	2.20	0.35
pmed36	10	800	12800	12287	12294.75	12329	3298.49	8	0.34	0.06
pmed37	80	800	12800	7487	7525.35	7569	46176.89	1	1.10	0.51
pmed38	5	900	16200	13651	13673.60	13764	1862.14	16	0.83	0.17
pmed39	10	900	16200	11671	11681.00	11816	4675.71	10	1.24	0.09
pmed40	90	900	16200	7520	7544.70	7649	62438.79	2	1.72	0.33
AVG							**9703.94s**	**14.03**	**0.49%**	**0.08%**

Tabela D.2. Detaljni rezultati za pαNMP algoritam ($\alpha = p$) testiran nad *OR-Library* test instancama

	P	*N*	*M*	*Best Value*	*AVG Value*	*Worst Value*	*Time*	*#Best*	*Gap (worst-best) /best*100*	*Gap (avg-best) /best*100*
pmed1	5	100	200	12910	13831.50	17536	2.06	13	35.83	7.14
pmed2	10	100	200	16635	18037.10	21348	80.36	14	28.33	8.43
pmed3	10	100	200	19386	19386.00	19386	104.74	20	0.00	0.00
pmed4	20	100	200	48378	52389.95	60020	320.37	4	24.06	8.29
pmed5	33	100	200	54774	55819.55	59532	1506.42	5	8.69	1.91
pmed6	5	200	800	14841	15111.15	20115	260.42	16	35.54	1.82
pmed7	10	200	800	16851	17134.50	17796	122.44	14	5.61	1.68
pmed8	20	200	800	37847	38912.35	42901	2012.83	9	13.35	2.81
pmed9	40	200	800	72049	79957.95	90674	4479.46	1	25.85	10.98
pmed10	67	200	800	100699	108994.70	134882	9635.65	1	33.95	8.24
pmed11	5	300	1800	12341	12341.00	12341	38.25	20	0.00	0.00
pmed12	10	300	1800	22525	22714.60	24061	1131.96	16	6.82	0.84
pmed13	30	300	1800	51937	54573.95	62345	5584.04	1	20.04	5.08
pmed14	60	300	1800	127514	149051.20	199856	13144.32	1	56.73	16.89
pmed15	100	300	1800	217144	242453.95	268962	25336.91	1	23.86	11.66
pmed16	5	400	3200	14328	14353.50	14413	432.45	14	0.59	0.18

pmed17	10	400	3200	20934	21446.80	23273	2003.99	4	11.17	2.45
pmed18	40	400	3200	84941	94885.60	109137	12265.23	1	28.49	11.71
pmed19	80	400	3200	190032	215983.40	244426	25450.39	1	28.62	13.66
pmed20	133	400	3200	377130	415835.55	475243	47285.32	1	26.02	10.26
pmed21	5	500	5000	15699	15730.90	16337	488.25	19	4.06	0.20
pmed22	10	500	5000	23377	23761.50	27229	1396.04	18	16.48	1.64
pmed23	50	500	5000	105569	136280.05	162735	18449.07	1	54.15	29.09
pmed24	100	500	5000	269848	310842.60	365236	45203.74	1	35.35	15.19
pmed25	167	500	5000	484161	536702.90	589284	104915.06	1	21.71	10.85
pmed26	5	600	7200	15313	15313.00	15313	336.52	20	0.00	0.00
pmed27	10	600	7200	19640	21026.80	21793	2882.72	5	10.96	7.06
pmed28	60	600	7200	133111	167999.45	242403	26800.72	1	82.11	26.21
pmed29	120	600	7200	334530	395481.60	476721	89343.46	1	42.50	18.22
pmed30	200	600	7200	701150	874546.20	1043990	199227.10	1	48.90	24.73
pmed31	5	700	9800	16543	16802.20	17953	589.98	13	8.52	1.57
pmed32	10	700	9800	24593	26111.30	33335	3258.24	4	35.55	6.17
pmed33	70	700	9800	198790	236720.15	295402	43475.22	1	48.60	19.08
pmed34	140	700	9800	433562	524149.20	641473	164218.65	1	47.95	20.89
pmed35	5	800	12800	18323	18599.60	19190	1096.57	6	4.73	1.51
pmed36	10	800	12800	24270	25180.90	27004	4234.06	10	11.26	3.75
pmed37	80	800	12800	286917	330158.65	374121	63514.03	1	30.39	15.07
pmed38	5	900	16200	17787	17989.10	18622	2194.45	7	4.69	1.14
pmed39	10	900	16200	24748	26689.20	29235	5323.34	1	18.13	7.84
pmed40	90	900	16200	293361	388867.20	473646	94568.69	1	61.45	32.56
AVG							**25567.84s**	**6.75**	**25.03%**	**9.17%**

Tabela D.3. Detaljni rezultati za pαNMP algoritam ($\alpha = 2$) testiran nad *OR-Library* test instancama

	Best-Known Value (1)	*Best-Found Value (2)*	*AVG Value (3)*	*Worst Value (4)*	*Time*	*#Best-Known*	*Gap* $\left[\frac{(4)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(3)-(1)}{(1)} * 100\right]$	*Gap* $\left[\frac{(2)-(1)}{(1)} * 100\right]$
pmed1	8032	8032	8032.00	8032	6.31	20	0.00	0.00	0.00
pmed2	5948	5948	5948.00	5948	30.16	20	0.00	0.00	0.00
pmed3	6465	6465	6465.00	6465	24.12	20	0.00	0.00	0.00
pmed4	5222	5222	5222.00	5222	89.44	20	0.00	0.00	0.00

pmed5	2964	2964	2964.00	2964	123.44	20	0.00	0.00	0.00
pmed6	10473	10473	10473.30	10474	174.89	14	0.01	0.00	0.00
pmed7	7425	7425	7425.00	7425	251.17	20	0.00	0.00	0.00
pmed8	6559	6559	6566.00	6603	2018.17	16	0.67	0.11	0.00
pmed9	4691	4691	4713.45	4759	5318.38	3	1.45	0.48	0.00
pmed10	2997	2997	3005.10	3020	8379.04	5	0.77	0.27	0.00
pmed11	9404	9404	9404.00	9404	58.60	20	0.00	0.00	0.00
pmed12	8555	8555	8555.00	8555	629.68	20	0.00	0.00	0.00
pmed13	6232	6232	6297.00	6395	5629.00	3	2.62	1.04	0.00
pmed14	5153	5155	5213.50	5278	8810.38	0	2.43	1.17	0.04
pmed15	3913	3926	3949.60	3987	13935.34	0	1.89	0.94	0.33
pmed16	10240	10240	10240.00	10240	53.86	20	0.00	0.00	0.00
pmed17	9044	9044	9055.35	9178	1739.51	16	1.48	0.13	0.00
pmed18	6907	6968	7014.80	7077	10229.49	0	2.46	1.56	0.88
pmed19	4953	4987	5039.15	5094	18940.14	0	2.85	1.74	0.69
pmed20	4216	4245	4287.00	4330	30018.15	0	2.70	1.68	0.69
pmed21	11597	11597	11604.10	11739	517.26	19	1.22	0.06	0.00
pmed22	10657	10657	10664.15	10748	2907.54	14	0.85	0.07	0.00
pmed23	6710	6718	6825.40	6895	15160.12	0	2.76	1.72	0.12
pmed24	5104	5180	5224.20	5275	25195.46	0	3.35	2.36	1.49
pmed25	4324	4371	4421.75	4459	13441.82	0	3.12	2.26	1.09
pmed26	12130	12130	12197.05	12279	442.31	11	1.23	0.55	0.00
pmed27	10000	10000	10027.50	10117	3344.95	8	1.17	0.28	0.00
pmed28	6461	6559	6647.05	6742	22910.04	0	4.35	2.88	1.52
pmed29	5271	5343	5409.25	5499	19337.99	0	4.33	2.62	1.37
pmed30	4616	4631	4684.65	4735	39873.49	0	2.58	1.49	0.32
pmed31	12430	12430	12430.00	12430	840.88	20	0.00	0.00	0.00
pmed32	11676	11676	11715.60	11788	4028.30	11	0.96	0.34	0.00
pmed33	6959	7058	7165.75	7273	29198.20	0	4.51	2.97	1.42
pmed34	5290	5390	5434.30	5546	39504.77	0	4.84	2.73	1.89
pmed35	12982	12982	13043.55	13398	762.51	12	3.20	0.47	0.00
pmed36	12287	12287	12323.95	12434	4762.13	4	1.20	0.30	0.00
pmed37	7487	7668	7767.25	7882	20533.98	0	5.28	3.74	2.42

pmed38	13651	13651	13686.35	13775	1546.94	14	0.91	0.26	0.00
pmed39	11671	11671	11702.90	11772	4666.58	5	0.87	0.27	0.00
pmed40	7520	7671	7768.40	7872	34542.05	0	4.68	3.30	2.01
AVG					**9749.41s**	**8.88**	**1.77%**	**0.94%**	**0.41%**

Literatura

[1] Albareda-Sambola, M., Hinojosa, Y., Marín, A., Puerto, J., 2015. When centers can fail: a close second opportunity. Computers and Operations Research, 62, 145–156.

[2] Beasley, J.E., 1990. OR-Library: distributing test problems by electronic mail. Journal of the operational research society, 41(11), 1069–1072.

[3] Brimberg, J., Mladenovic, N., Todosijevic, R., Urosevic, D., 2017. Less is more: solving the max-mean diversity problem with variable neighborhood search. Information Sciences, 382, 179-200.

[4] Calik, H., Labbé, M., Yaman, H., 2019. p-Center Problems. Location Science, 51-65.

[5] Calik, H., Tansel, B.C., 2013. Double bound method for solving the p-center location problem. Computers and Operations Research, 40, 2991–2999.

[6] Costa, L.R., Aloise, D., Mladenovic, N., 2017. Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. Information Sciences, 415, 247-253.

[7] Daskin, M.S., 2013. Network and discrete location: models, algorithms, and applications, 2nd edn. Wiley, Hoboken.

[8] Davidovic, T., Ramljak, D., Selmic, M., Teodorovic, D., 2011. Bee colony optimization for the p-center problem. Computers and Operations Research, 38, 1367–1376 .

[9] Elloumi, S., Labbé, M., Pochet, Y., 2004. A new formulation and resolution method for the p-center problem. INFORMS Journal on Computing, 16, 84–94.

[10] Feo, T.A., Resende, M.G.C., 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, 8(2), 67–71.

[11] Garfinkel, R., Neebe, A., Rao, M., 1977. The m-center problem: minimax facility location. Management Science, 23(10), 1133-1142.

[12] Goncalves-E-Silva, K., Aloise, D., Xavier-De-Souza, S., Mladenovic, N., 2018, Less is more: Simplified Nelder-Mead method for large unconstrained optimization. Yugoslav Journal of Operations Research, 28(2), 153-169.

[13] Hakimi, S.L., 1964 Optimum locations of switching centers and the absolute centers and medians of a graph. Operations Research, 12(3), 450–459.

[14] Hakimi, S.L., 1965. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Operations Research, 13(3), 462-475.

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

[16] Hochbaum, D.S., Shmoys, D.B., 1985. A best possible heuristic for the k-center problem. Mathematics of Operations Research, 10(2), 180-184.

[17] Hsu, W.L., Nemhauser, G.L., 1979. Easy and hard bottleneck location problems. Discrete Applied Mathematics, 1(3), 209–215.

[18] Ilhan, T., Pınar, M.C., 2001. An efficient exact algorithm for the vertex p-center problem. Technical report, Department of Industrial Engineering, Bilkent University.

[19] Kariv, O., Hakimi, S.L., 1979. An algorithmic approach to network location problems. I: The p-Centers. SIAM Journal on Applied Mathematics, 37(3), 513-538.

[20] Kariv, O., Hakimi, S.L., 1979. An algorithmic approach to network location problems. II: The p-Medians. SIAM Journal on Applied Mathematics, 37(3), 539-560.

[21] López-Sánchez, A.D., Sánchez-Oro, J., Hernández-Díaz, A.G., 2019. GRASP and VNS for solving the p-next center problem. Computers and Operations Research, 104, 295-303.

[22] Martínez-Merino, L.I., Albareda-Sambola, M., Rodríguez-Chia, A.M., 2017. The probabilistic p-center problem: planning service for potential customers. European Journal of Operational Research, 262(2), 509–520.

[23] Mihelic, J., Robic, B. 2005. Solving the k-center problem efficiently with a dominating set algorithm. Journal of Computing and Information Technology, 13(3), 225-23.

[24] Mikic, M., Todosijevic, R., Urosevic, D., 2019. Less is more: General variable neighborhood search for the capacitated modular hub location problem. Computers and Operations Research, 110, 101-115.
[25] Minieka, E., 1970. The m-center problem. Society for Industrial and Applied Mathematics, 12(1), 138–139.
[26] Mladenovic, N., Brimberg, J., Hansen, P., Moreno-Perez, J.A., 2007. The p-median problem: A survey of metaheuristic approaches. European Journal of Operational Research, 179(3), 927-939.
[27] Mladenovic, N., Hansen, P., 1997. Variable neighborhood search. Computers and Operations Research, 24(11), 1097–1100.
[28] Mladenovic, N., Labbé, M., Hansen, P., 2003. Solving the p-center problem with tabu search and variable neighborhood search. Networks 42(1), 48–64.
[29] Mladenovic, N., Todosijevic, R., Urosevic, D., 2016. Less is more: basic variable neighborhood search for minimum differential dispersion problem. Information Sciences, 326, 160-171.
[30] Pullan, W., 2008. A memetic genetic algorithm for the vertex p-center problem. Evol Comput, 16(3), 417–436.
[31] Ristic, D., Mladenovic, N., Todosijevic, R., Urosevic, D., 2021. Filtered variable neighborhood search method for the p-next center problem. International Journal for Traffic and Transport Engineering, 11(2), 294 - 309.
[32] Whitaker, R.A., 1983. A Fast Algorithm For The Greedy Interchange For Large-Scale Clustering And Median Location Problems. INFOR: Information Systems and Operational Research, 21(2), 95-108.

Biografija autora

Dalibor V. Ristić je rođen 1981. godine u Nišu. Osnovnu školu je završio u Svrljigu kao đak generacije, a gimnaziju prirodno-matematičkog usmerenja u Nišu kao nosilac Vukove diplome. Tokom osnovne i srednje škole je bio učesnik i dobitnik nagrada na mnogim matematičkim takmičenjima u organizaciji Ministarstva prosvete Republike Srbije. Na Elektronskom fakultetu Univerziteta u Nišu, smer za računarsku tehniku i informatiku, diplomirao je 2007. godine. Doktorske studije na Računarskom fakultetu u Beogradu upisao je 2013. godine.

Počev od 2008. godine obavljao je poslove programera i softverskog inženjera u Vojsci Srbije i više softverskih kompanija u Beogradu i Novom Sadu. U zvanju asistenta na Računarskom fakultetu držao je vežbe iz sledećih kurseva: Uvod u programiranje, Objektno orijentisani dizajn i metodologija, Funkcionalno programiranje i Dizajn i analiza algoritama.

Универзитет УНИОН у Београду

Изјава о ауторству

Факултет	Рачунарски факултет
Име и презиме аутора	Далибор Ристић
Број индекса	Д01/2013

Изјављујем

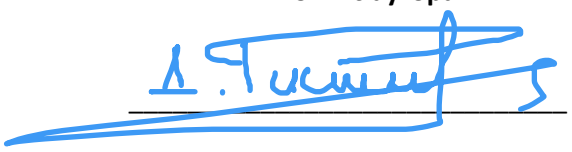
да је докторска дисертација, односно, докторски уметнички пројекат под насловом:

Имплементација метода променљивих околина за решавање p-следећи проблема над скупом чворова графа

- резултат мог сопственог истраживачког / уметничког рада;
- да рад у целини ни у деловима није био предложен за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени;
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 30.11.2022.



Универзитет УНИОН у Београду

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора	Далибор Ристић
Број индекса	Д01/2013
Студијски програм	Алгоритми, комбинаторика, оптимизација
Наслов рада	**Имплементација метода променљивих околина за решавање p-следећи проблема над скупом чворова графа**
Ментор	др Драган Урошевић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањивања у дигитални репозиторијум Универзитета ”Унион” у Београду.

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука/доктора уметности, као што су: име и презиме, година и место рођења и датум одбране рада.

Ови подаци могу се објавити у дигиталном репозиторијуму Универзитета ”Унион” у Београду и у Националном репозиторијуму дисертација у Србији – НАРДУС.

Потпис аутора

У Београду, 30.11.2022.