UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Aleksandar D. Đenić

REŠAVANJE DISKRETNIH LOKACIJSKIH PROBLEMA PRIMENOM METODE PROMENLJIVIH OKOLINA

doktorska disertacija

Beograd, 2018.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Aleksandar D. Đenić

VARIABLE NEIGHBORHOOD SEARCH FOR SOLVING DISCRETE LOCATION PROBLEMS

Doctoral Dissertation

Belgrade, 2018.

Mentor:

dr Miroslav Marić, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Vladimir Filipović, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

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

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

dr Nenad Mladenović, redovni profesor
Univerzitet u Beogradu, Fakultet organizacionih nauka

Datum odbrane: _____

Naslov disertacije: Rešavanje diskretnih lokacijskih problema primenom metode promenljivih okolina

Rezime: Predmet ovog rada je analiza i rešavanje dva diskretna lokacijska problema: problema određivanja lokacija autobuskih terminala (engl. Bus Terminal Location Problem - BTLP) i problema uspostavljanja centara za produženu negu pacijenata (engl. Long-term Care Facility Location Problem - LTCFLP). U radu je prikazana metoda promenljivih okolina (engl. Variable Neighborhood Search - VNS) za rešavanje BTLP i LTCFLP problema. VNS je metaheuristika vođena jednim rešenjem i zasnovana je na sistematičnoj pretrazi okolina rešenja prostora pretrage. Sastoji se iz dve faze, faze razmrdavanja i faze lokalne pretrage.

BTLP predstavlja diskretan lokacijski problem koji podrazumeva uspostavljanje velikih autobuskih terminala kako bi se omogućila što kvalitetnija usluga klijentima. Klijenti predstavljaju lokacije autobuskih i metro stanica javnog prevoza. Za rešavanje BTLP problema VNS metodom predstavljena je unapređena lokalna pretraga zasnovana na brzoj zameni okolina. Metoda je paralelizovana (PVNS) i postignuto je značajno vremensko ubrzanje metode u zavisnosti od broja jezgara procesora na kome se izvršava. Predložena PVNS metoda daje rešenja boljeg kvaliteta u odnosu na poznata rešenja BTLP problema iz literature. Algoritam je testiran i na većim instancama problema dobijenih modifikacijom biblioteke instanci za problem trgovačkog putnika i predstavljeni su rezultati metode na ovim instancama.

LTCFLP je nastao kao deo planiranja sistema zdravstvene zaštite u Južnoj Koreji. Klijenti predstavljaju lokacije na kojima se nalaze grupe pacijenata kojima je potrebna produžena nega, dok uspostavljeni centri predstavljaju lokacije na kojima će se izgraditi zdravstveni centri koji će pružati negu pacijentima. Unapred je zadato n lokacija na kojima mogu biti uspostavljeni centri. Potrebno je odabrati najviše K lokacija za uspostavljanje zdravstvenih centara tako da oni budu što ravnomernije opterećeni zahtevima klijenata. Za rešavanje LTCFLP problema VNS metodom predstavljena je struktura podataka zasnovana na brzoj zameni okolina uz pomoć koje je vremenska složenost jedne iteracije lokalne pretrage smanjena na $O(n \cdot \max(n, K^2))$ u odnosu na vremensku složenosti poznatu u literaturi $O(K^2 \cdot n^2)$. Smanjena vremenska složenost predstavljene lokalne pretrage dovela je do boljih rezultata zbog većeg broja iteracija VNS algoritma koje koje se mogu izvršiti u kraćem vremenskom periodu. Prikazani su rezultati predstavljenog algoritma koji nadmašaju poznate rezultate iz literature.

Ključne reči: lokacijski problemi, problem određivanja lokacija autobuskih terminala, problem uspostavljanja centara za produženu negu pacijenata, kombinatorna optimizacija, metaheuristike, paralelizacija, metoda promenljivih okolina

Naučna oblast: računarstvo

Uža naučna oblast: računarska inteligencija

Dissertation title: Variable Neighborhood Search for Solving Discrete Location Problems

Abstract:

This paper considers two discrete location problems: Bus Terminal Location Problem (BTLP) and Long-term Care Facility Location Problem (LTCFLP). Variable Neighborhood Search (VNS) method for solving BTLP and LTCFLP is presented in this paper. VNS is a single-solution based metaheuristic based on systematic change of neighborhoods while searching for optimal solution of the problem. It consists two main phases: shake phase and local search phase.

BTLP is a discrete location problem which considers locating bus terminals in order to provide the highest possible quality of public service to the clients. Clients are presented as public transportation stations, such as bus or metro stations. VNS algorithm is used for solving BTLP. This algorithm uses improved local search based on efficient neighborhood interchange. VNS is parallelized (PVNS) which leads to significant time improvement in function of the processor core count. Computational results show that proposed PVNS method improves existing results from the literature in terms of quality. Larger instances, based on instances from the Traveling Salesman Problem library, are presented and computational results for those instances are reported.

LTCFLP is created as a part of health care infrastructure planning in South Korea. Clients are considered as groups of patients with a need of long-term health care, while established facilities present locations where the centers that provide health care services should be built. Predefined are n locations where centers are to be established. This problem seeks at most K locations to establish health centers so they are to be equally loaded with clients demand. For solving LTCFLP, by using VNS algorithm, data structure based on fast interchange is presented. It reduces the time complexity of one iteration of local search algorithm to $O(n \cdot \max(n, K^2))$ comparing to the known time complexity from the literature $O(K^2 \cdot n^2)$. Reduced time complexity of the presented VNS leads to better quality solutions, due to larger number of VNS iterations that can be performed in less computational time. This paper presents computational results that outperform the best known results from the literature.

Keywords: Facility Location Problems, Bus Terminal Location Problem, Long-

term Care Facility Location Problem, Combinatorial Optimization, Metaheuristics, Parallelization, Variable Neighborhood Search

Research area: computer science

Research sub-area: Computational intelligence

Zahvalnica

Želelo bih da se zahvalim svom mentoru, prof. dr Miroslavu Mariću, na ukazanom poverenju, na svim savetima, nesebičnom zalaganju i velikoj podršci tokom realizacije ove disertacije. Takođe, hvala prof. dr Vladimiru Filipoviću na korisnim sugestijama tokom izrade ove disertacije. Veliku zahvalnost na lepoj saradnji, pomoći i savetima dugujem i prof. dr Filipu Mariću i prof. dr Zorici Stanimirović koji su značajno doprineli konačnom oblikovanju ovog rada. Posebnu zahvalnost na ukazanoj podršci i poverenju dugujem i prof. dr Nenadu Mladenoviću koji me je svojim radom i uspesima inspirisao i zainteresovao za metodu promenljivih okolina.

Zahvalnost i ljubav dugujem svojoj porodici i prijateljima koji su svojom podrškom, razumevanjem i strpljenjem umnogome doprineli ostvarenju mojih profesionalnih ciljeva.

Sadržaj

Glava 1

Uvod

1.1 Problem matematičke optimizacije

Neka su dati skup S, *funkcija cilja* $f : S \to \mathbb{R}$ i funkcije ograničenja $g_i : S \to \mathbb{R}, i = 1, 2, ..., m$. *Dopustiv skup* $X \subseteq S$ je zadat nizom ograničenja nad skupom S:

$$X = \{x \in S | g_i(x) \leq 0, i = 1, 2, ..., m\}.$$

Problem matematičke optimizacije je definisan kao pronalaženje

$$min\{f(x) | x \in X\}.$$

Problem matematičke optimizacije se često naziva i problem matematičkog programiranja ili samo problem optimizacije. Ukoliko je skup S konačan zadati problem je diskretan i reč je o *problemu kombinatorne optimizacije*. Ukoliko je $S = \mathbb{R}^n$ reč je o *problemu kontinualne (globalne) optimizacije*.

Elemente $x \in X$ nazivamo *rešenjima* problema matematičke optimizacije. Dopustiv skup X nazivamo prostorom rešenja ili prostorom pretrage datog problema. Rešenje $x^* \in X$ je *globalni optimum* ukoliko važi

$$\forall x \in X : f(x^*) \leq f(x).$$

Rešenje $x' \in X$ predstavlja *lokalni minimum* u nekoj svojoj okolini $N(x')$, $N(x') \subseteq X$, ukoliko važi

$$\forall x \in N(x') : f(x') \leq f(x).$$

Egzaktan algoritam za rešavanje problema matematičke optimizacije, ukoliko postoji, u konačnom broju koraka pronalazi optimalno rešenje x^* ili sa druge strane pokazuje da ne postoji dopustivo rešenje.

Problemi optimizacije koji imaju realnu primenu su često veoma složeni i teški za rešavanje. Egzaktne metode često ne mogu da reše ovakve probleme koristeći raspoložive vremenske i memorijske računarske resurse. Na primer, ako je za rešavanje problema klasičnom računaru potrebno više od 100 ili 1000 godina da ga reši, tada je korišćena metoda rešavanja praktično neupotrebljiva i potrebno je pronaći drugu metodu za rešavanje datog problema.

Kod hitnih intervencija, može postojati zahtev da se težak problem reši za svega nekoliko sati ili nekoliko minuta. U takvim situacijama se često koriste približne (heurističke) metode. Za razliku od egzaktnih algoritama heurističke metode skoro uvek pronađu rešenje, ali ne garantuju njegovu optimalnost. Tako se za dobijeno rešenje problema ne zna da li je ono optimalno, ali se pretpostavlja da je dovoljno dobro za realnu primenu zbog koje je problem nastao. Na primer, nakon neočekivane situacije na aerodromu, potrebno je brzo izračunati novi raspored rada šaltera za prijavu [19], kao i novi raspored postavljanja aviona na izlaze [32] tako da se ukupno čekanje aviona i putnika i operativna cena rada šaltera minimizuju. U praksi nisu značajna mala odstupanja ukupnog čekanja svih aviona i putnika od optimalnog čekanja, na primer za 20 ili 30 sekundi. Međutim, problem nastaje u slučaju značajnijeg odstupanja, na primer, za pola sata i više.

Jedna *klasa problema optimizacije* predstavlja skup problema koji imaju neke zajedničke osobine. U tom smislu može se govoriti o klasi problema rasporeda rada šaltera za prijavu na aerodromu, gde je potrebno odrediti optimalan odnos između operativnih troškova rada šaltera i ukupnog čekanja putnika na terminalu. Sa druge strane, jedna *instanca problema* predstavlja konkretan zadatak koji je potrebno rešiti. U prethodnom primeru instancu predstavljaju konkretan aerodrom, spiskovi letova, putnika i dostupnih šaltera i cene operativnih troškova rada šaltera. Instance problema koje pripadaju jednoj klasi problema karakteriše njihova veličina, odnosno broj (dimenzija) ulaznih podataka. Kaže se da je jedna instanca problema dimenzije n ukoliko se ona sastoji od n ulaznih podataka. Na primer, u ranije navedenom problemu dimenzija problema zavisi od ukupnog broja letova, putnika i dostupnih šaltera.

U daljem tekstu će se često, radi jednostavnosti, pod pojmom problema podrazumevati pojam klase problema ili jedne instance problema kada se iz ostatka teksta može jasno razumeti kontekst u kome se problem spominje.

1.2 Složenost problema matematičke optimizacije

U ovoj sekciji je predstavljena osnova teorije složenosti izračunavanja sa ciljem prikaza osnovnih pojmova korišćenih u disertaciji. Više detalja o teoriji složenosti izračunavanja se može naći u [7, 145].

Složenost jednog problema matematičke optimizacije podrazumeva količinu vremena i memorije potrebne da se taj problem reši na jednom računaru. Vreme rešavanja jednog problema zavisi od brzine računara na kome se problem rešava i meri se u sekundama, minutima ili nekoj sličnoj fizičkoj veličini. Ukoliko se umesto fizičkog računara koriste različiti apstraktni računarski sistemi (na primer, deterministička Tjuringova mašina [135, 39]), moguće je analizirati težinu problema nezavisno od specifičnog računara na kome problem rešava. Tada se vreme izvršavanja programa ne meri vremenskom fizičkom veličinom, već brojem koraka koji su potrebni da bi se rešila jedna instanca problema.

Teorija složenosti izračunavanja predstavlja osnovu teorije računarstva i aktivno se istražuje još od sredine šezdesetih godina prošlog veka [69, 25, 96, 7, 39]. Jedan od osnovnih zadataka teorije složenosti je proceniti broj koraka algoritma i količinu memorije potrebne za rešavanje određenog problema.

Notacija velikog O

Za date dve funkcije $f, g : \mathbb{N} \to \mathbb{N}$, važi $f(x) = O(g(x))$, ako postoje pozitivni celi brojevi c i x_0 takvi da za svako $x \geq x_0$ važi:

$$f(x) \leq cg(x).$$

Na primer, za funkcije $f(x) = 6x^4 - 2x^3 + 5$ i $g(x) = x^4$ važi $f(x) = O(g(x))$. Na osnovu definicije dovoljno je pokazati da postoje $c, x_0 > 0$ za koje važi da je $6x^4 - 2x^3 + 5 \leq cx^4$, za svako $x > x_0$. Neka je $x_0 = 1$. Tada važi:

$$\begin{aligned} 6x^4 - 2x^3 + 5 &\leq |6x^4 - 2x^3 + 5| \\ &\leq |6x^4| + |2x^3| + 5 \\ &= 6x^4 + 2x^3 + 5 && \text{jer je } x > 1 \\ &\leq 6x^4 + 2x^4 + 5x^4 && \text{jer je } x > 1 \\ &= 13x^4 \end{aligned}$$

Dakle za $c = 13$ je tvrdnja pokazana.

Vremenska složenost

Da bi se uporedile performanse dva različita algoritma neophodno je uporediti količinu potrebnih resursa tih algoritama za rešavanje problema različitih dimenzija. Najčešći pristup za poređenje predstavlja analiza rasta potrebne količine resursa sa povećanjem dimenzije razmatranog problema [96]. Za ovu vrstu analize se koristi notacija velikog O, gde se složenost izražava u funkciji dimenzije problema. Koriste se dva pristupa za procenu složenosti algoritma: analiza najgoreg slučaja i analiza prosečnog slučaja. Složenost najgoreg slučaja je zasnovana na slučaju za koji se algoritam najduže izvršava (ili koristi najviše memorije), dok je analiza prosečnog slučaja zasnovana na prosečnom vremenu izvršavanja algoritma. U nastavku teksta biće podrazumevana analiza najgoreg slučaja ukoliko nije navedeno drugačije. Tako je vremenska složenost linearne pretrage niza dimenzije n jednaka $O(n)$. Prosečna vremenska složenost sortiranja niza dimenzije n algoritmom *quick-sort* je jednaka $O(n \cdot log(n))$, dok je najgora vremenska složenost sortiranja niza dimenzije n algoritmom *quick-sort* jednaka $O(n^2)$.

Može se izdvojiti klasa algoritama čija je vremenska složenost $O(1)$ i oni se nazivaju algoritmima konstantne složenosti. Kod ove klase algoritama, vreme izvršavanja algoritma ne zavisi od dimenzije problema koji se rešava. Primeri algoritama konstante složenosti su algoritam za pristupanje određenom elementu niza ili algoritam za računanje vrednosti određenog izraza.

Klasu algoritama linearne vremenske složenosti čine algoritmi čija je vremenska složenost $O(n)$. Kod ove klase algoritama, vreme izvršavanja programa linearno raste sa porastom dimenzije problema. Primer algoritma linearne vremenske složenosti je pronalazak minimalne vrednosti u nizu.

Algoritmi polinomijalne vremenske složenosti čine klasu algoritama složenosti $O(n^k)$, gde je k konstanta koja zavisi od problema koja se rešava. Treba naglasiti da vrednost k ne sme da zavisi od dimenzije konkretne instance problema, kao i da su u praksi relevantni oni algoritmi kod kojih je vrednost k relativno mala. Primer algoritma vremenske složenosti $O(n^2)$ je *bubble-sort* algoritam za sortiranje nizova.

U klasu algoritama eksponencijalne vremenske složenosti spadaju algoritmi čija je složenost jednaka $O(c^n)$, gde je c konstantna vrednost. Drugim rečima pri linearnom rastu dimenzije razmatranog problema, vreme izvršavanja raste eksponencijalno. Primer algoritma vremenske složenosti $O(c^n)$ je pronalazak varijacije dužine n nad c elemenata koja zadovoljava određeni uslov pretragom svim mogućih kombinacija.

Problem odlučivanja

U teoriji složenosti, *problem odlučivanja* predstavlja pitanje zadato u odabranom apstraktnom računarskom sistemu na koje se može odgovoriti sa *da* ili *ne*. Primer problema odlučivanja je: „za data dva broja x i y, da li je x deljivo sa y?". Odgovor može biti *da* ili *ne* u zavisnosti od vrednosti brojeva x i y.

Za svaki problem optimizacije se može kreirati odgovarajući problem odlučivanja uvođenjem granice g za vrednost funkcije cilja. Tada odgovarajući problem odlučivanja glasi: „da li postoji $x \in X$ takvo da je $f(x) \leq g$?".

Redukcija

Redukcija predstavlja mehanizam transformacije jednog problema u drugi gde se dobijeno rešenje drugog problema može iskoristiti za rešavanje prvog problema [123]. U matematici postoje mnogi primeri redukcije. Na primer, problem rešavanja sistema linearnih jednačina se može transformisati u problem pronalaska inverzne matrice.

Neka su A i B dva problema odlučivanja. Kažemo da se A može linearno redukovati na B (A je linearno svodljiv na B) ako na osnovu algoritma koji rešava problem B u vremenu $O(t(n))$ možemo konstruisati algoritam koji rešava problem A takođe u vremenu $O(t(n))$, za bilo koju funkciju $t(n)$. Ako važi da je problem A linearno svodljiv na problem B i ujedno važi da je problem B linearno svodljiv na problem A kažemo da su oni linearno ekvivalentni.

Problem A je polinomijalno svodljiv na problem B ukoliko možemo konstruisati algoritam za rešavanje problema A u polinomijalnom vremenu ne uzimajući u obzir vreme potrebno za rešavanje problema B. Drugim rečima algoritam za rešavanje problema A može koristiti, kada god je potrebno, algoritam koja rešava bilo koju instancu problema B, bez uzimanja u obzir cene za rešavanje problema B.

Klase složenosti

Klasa složenosti se može definisati kao skup problema koji se mogu rešiti u odgovarajućem apstraktom računarskom sistemu koristeći $O(f(n))$ resursa R, gde je f funkcija koja zavisi od dimenzije ulaznih podataka n [39]. Na primer klasa problema vremenske složenosti $O(n^2)$ predstavlja sve probleme za koje postoji algoritam vremenske složenosti $O(n^2)$ koji ih rešava.

Problem odlučivanja pripada *klasi složenosti* P ako se može rešiti algoritmom sa polinomijalnom vremenskom složenošću. Problemi iz klase P se mogu jednostavno rešiti na računaru. Primeri problema iz klase P su provera da li je broj prost, linearna pretraga i sortiranje niza.

Problem odlučivanja propada *klasi* NP ako se za jedno potencijalno rešenje može proveriti da li je zaista rešenje (da li je odgovor *da*) algoritmom sa polinomijalnom vremenskom složenošću. Kod problema iz klase NP se ne zahteva postojanje procedure za brzi pronalazak svih rešenja, već samo za brzu proveru da li je potencijalno rešenje zaista rešenje problema. Primer problema iz klase NP je pronalaženje podskupa datog skupa celih brojeva, takvog da je zbir elemenata podskupa jedak nuli. Na primer za skup $\{-1, -2, 3, 9, 8\}$ postoji podskup $\{-1, -2, 3\}$ gde je $(-1)+(-2)+3 = 0$. Jasno je da računanje zbira jednog podskupa vremenske složenosti $O(n)$.

Problem odlučivanja X je *NP-kompletan* ako pripada klasi NP i ako za svaki problem Y iz klase NP vazi da je Y polinomijalno svodljiv na X. Može se reći da NP-kompletni problemi predstavljaju jezgro najtežih problema iz klase NP. Interesantna karakteristika NP-kompletnih problema je da ukoliko je poznat skup problema koji su NP-kompletni i treba dokazati da je novi problem X takođe NP-kompletan, dovoljno je da odabrati jedan problem Y iz skupa i pokazati da je Y polinomijalno svodljiv na X. Jedan od prvih problema za koje je dokazano da je NP-kompletan je problem iskazne zadovoljivosti [25, 87]. Nakon ovog rezultata bilo je dosta jednostavnije pokazati da su mnogi drugi problemi NP-kompletni svođenjem na problem iskazne zadovoljivosti i na neki od problema za koje je u međuvremenu dokazano da su NP-kompletni.

Očigledno je da svi problemi klase P pripadaju klasi NP. Međutim još uvek ne postoji odgovor na pitanje da li svi problemi klase NP pripadaju klasi P, odnosno da li je $P = NP$. Kada bi se dokazalo da je $P = NP$, tada bi svi problemi iz klase NP, pa i svi NP-kompletni problemi bili rešivi algoritmom sa polinomijalnom vremenskom složenošću. Ovaj problem još uvek nije rešen i predstavlja jedan od najvećih problema savremene teorije računarstva. Trenutno većinsko uverenje je da je $NP \neq P$ zbog težine rešavanja NP-kompletnih problema.

Problem X je *NP-težak* ako i samo ako postoji NP-kompletan problem L koji je polinomijalno svodljiv na X. Za NP-teške probleme se kaže da su teški bar koliko i NP-kompletni problemi. Primer NP-teškog problema je problem trgovačkog putnika [70]. Postoje NP-teški problemi koji nisu NP-kompletni, na primer *halting*

problem - problem ispitivanja zaustavljanja programa.

1.3 Lokacijski problemi

Lokacijske odluke predstavljaju veoma važne elemente planiranja razvoja jedne organizacije. Posledice ovakvih odluka su dugoročne i imaju uticaj na brojna operativna i logistička rešenja. Visoka cena imovine i izgradnje postrojenja čini projekte izgradnje i realokacije postrojenja dugoročnim investicijama. Zbog toga donosioci odluka moraju da razmišljaju o ceni svake ovakve investicije kratkoročno, ali i u dužem vremenskom periodu, čak i kada se osnovni faktori okruženja promene. Pronalazak i rešavanje robusnih lokacijskih modela je težak zadatak, koji je neophodan za donosioce odluka kako bi mogli da planiraju dalji razvoj organizacije i poslovanja.

Savremena *teorija lokacijskih problema* uključuje istraživanja vezana za analizu lokacijskih problema, formulisanje matematičkih modela koji adekvatno opisuju ove probleme, kao i istraživanja vezana za algoritme za rešavanje lokacijskih problema. Brojni do sada predloženi modeli lokacijskih problema se koriste u procesu planiranja i upravljanja u privatnom i javnom sektoru, na primer prilikom određivanja lokacija za izgradnju industrijskih postrojenja, banaka, prodajnih objekata, bolnica, pošta, policijskih stanica, itd.

Koncept lokacijskih problema uveo je Alfred Veber 1909. godine u radu [140] (prevedeno na engleski jezik 1929. godine u [139]), u kome je razmatran problem određivanja optimalne lokacije jednog skladišta tako da se minimizuje ukupna udaljenost između skladišta i klijenata koji se iz njega snabdevaju. Jedan od prvih lokacijskih problema iz literature je i problem lociranja dve konkurentske radnje u jednoj ulici, koji je formulisao Hoteling 1929. godine u radu [72]. Dalja istraživanja ranih lokacijskih problema su nastavljena 40-ih i 50-ih godina prošlog veka u [76, 124, 130].

Intenzivnija istraživanja u oblasti lokacijskih problema su započeta 1950-ih godina, kada su nastale brojne studije u kojima su razmatrani problemi raspoređivanja centara [4, 75, 107, 109, 115] i ekonomski faktori vezani za lociranje centara proizvodnje [81, 108]. U tom periodu rad na teoriji lokacijskih problema se sastojao primarno od istraživanja različitih pojedinačnih problema i njihovih primena, a ne na jedinstvenoj teoriji. U to vreme počinju da se istražuju primene u određivanju lokacija vatrogasnih centara [137], deponija [141], fabričkih postrojenja [20], u telefonskoj [114] i železničkoj industriji [85].

Veće interesovanje za teorijske aspekte lokacijskih problema pojavilo se 1964. godine u radu [57] gde je razmatran problem lociranja jednog ili više centara u mreži sa ciljem minimizacije ukupne sume rastojanja ili maksimalnog rastojanja u mreži. Od tada je najveći deo istraživanja u ovoj oblasti deo teorije lokacijskih problema, a ne deo istraživanja pojedinačnih primena lokacijskih problema.

Teorija lokacijskih problema se odnosi na modelovanje, analizu i rešavanje klasa problema koji se mogu opisati kao lociranje postrojenja u određenom prostoru. Postoje četiri osnovne komponente koje opisuju lokacijske probleme:

- klijenti za koje se pretpostavlja da su unapred locirani,
- postrojenja (ili centri) koje je potrebno locirati,
- prostor u kome se nalaze klijenti i postrojenja i
- metrika udaljenosti između centara i klijenata.

Prostor može biti modelovan kao graf, mreža, stablo, geometrijska ravan, itd. Cilj je pronaći lokacije centara, nekada i alokacije klijenata ka centrima, tako da se optimizuje eksplicitno ili implicitno definisan cilj. Na primer moguće je minimizovati ukupan transportni trošak, prosečno vreme odgovora, cenu putovanja, maksimalnu dužinu puta, itd. Uopšteno, funkcija cilja se sastoji od uslova koji uključuju razdaljinu između centara i klijenata u datom prostoru.

Sa povećanjem brzine razvoja novih lokacijskih problema, pojavila se potreba za uvođenjem njihove sistematizovane podele. U literaturi postoje mnogi pregledni radovi koji razmatraju različite podele lokacijskih problema [45, 16, 84, 27, 119]. U nastavku su navedene neke od značajnih podela lokacijskih problema. Kada posmatramo prostor pretrage, on može biti konačan i beskonačan, preciznije razlikuju se diskretni [28] i kontinualni lokacijski problemi [17]. Metrika distance između objekata u prostoru može biti euklidska i neeuklidska. Ukoliko su centri koji se uspostavljaju uređeni u hijerarhiju, izdvajaju se hijerarhijski lokacijski problemi [88]. Lokacijske probleme možemo podeliti na one kod kojih je unapred poznat broj centara koje treba uspostaviti [102] i one gde taj broj nije unapred poznat [40]. Ukoliko nije poznat broj centara koje treba uspostaviti, uglavnom je zadata cena uspostavljanja svakog centra i potrebno je pronaći balans između broja uspostavljenih centara i kvaliteta usluge koji ti centri pružaju klijentima. Ukoliko se posmatra funkcija cilja lokacijski problemi se mogu podeliti na *min-sum* [102, 40, 128] i *min-max* probleme [113, 104]. Kod *min-sum* problema je cilj minimizovati

ukupnu distancu između centara i klijenata, dok je kod *min-max* problema cilj minimizovati najveću distancu između centra i klijenta. Uspostavljeni centri mogu imati zadate kapacitete, tj. ograničenja u vidu količine proizvedenog resursa potrebnog klijentima [89, 129, 1]. Klijenti uglavnom koriste usluge sebi najbliži uspostavljeni centar, ali mogu koristiti usluge uspostavljenog centra koji im najviše odgovara, što može biti definisano odgovarajućim prioritetima [93].

Posebno su interesantni lokacijski problemi kod kojih se pojavljuje neki vid nepouzdanosti ulaznih podataka [125]. U izvesnim situacijama svi parametri problema su deterministički i poznati. Ukoliko se nepouzdanost uzme u obzir, pojavljuju se određena slučajna ponašanja i izdvajaju se *problemi stohastičke optimizacije* i *problemi robusne optimizacije*. Problemi stohastičke optimizacije su nedeterministički, gde su parametri problema definisani nekom verovatnosnom raspodelom [83]. Problemi robusne optimizacije su deterministički i modeluju se tako što određene promenljive mogu menjati vrednost unutar unapred fiksiranog intervala [98, 11, 99]. U tim modelima se razmatra najgori mogući scenario i funkcija cilja ima *min-max* oblik.

U nastavku su predstavljeni neki od najznačajnijih lokacijskih problema: lokacijski problem bez ograničenja kapaciteta (engl. Uncapacitated Facility Location Problem - UFLP), problem p-medijane, problem p-centra i problem maksimalnog pokrivanja (engl. Maximal Covering Location Problem - MCLP).

Lokacijski problem bez ograničenja kapaciteta

Lokacijski problem bez ograničenja kapaciteta predstavlja problem kod kog je potrebno minimizovati ukupnu sumu transportnih troškova zajedno sa troškovima izgradnje centara [40]. UFLP spada u kategoriju *min-sum* problema.

Neka $I = \{1, 2, 3, ..., m\}$ predstavlja skup klijenata i neka $J = \{1, 2, 3, ..., n\}$ predstavlja skup potencijalnih lokacija za izgradnju centara. Tradicionalno se kaže da je centar j otvoren ili uspostavljen ukoliko je odlučeno da se na lokaciji j izgradi centar. Neka je nizom f_j zadata cena otvaranja centra za svaku lokaciju j i neka je matricom udaljenosti d_{ij} predstavljena udaljenost između svakog klijenta i i potencijalnog centra j. Neka su y_j celobrojne promenljive za koje važi

$$y_j = \begin{cases} 1, & \text{ukoliko je centar } j \text{ otvoren,} \\ 0, & \text{ukoliko je centar } j \text{ zatvoren.} \end{cases}$$

Neka su x_{ij} celobrojne promenljive za koje važi

$$x_{ij} = \begin{cases} 1, & \text{ukoliko je zahtev klijenta } i \text{ zadovoljen od strane centra } j, \\ 0, & \text{inače.} \end{cases}$$

Tada je UFLP formulisan kao:

$$min \sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} + \sum_{j \in J} f_j y_j \tag{1.1}$$

pri uslovima:

$$\sum_{j \in J} x_{ij} = 1 \qquad i \in I, \tag{1.2}$$

$$x_{ij} \leq y_j \qquad i \in I, j \in J, \tag{1.3}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad i \in I, j \in J. \tag{1.4}$$

Ograničenja (1.2) garantuju da će svaki klijent biti pridružen tačno jednom centru. Uslovi (1.3) obezbeđuju da se pridruživanje klijenata vrši samo ka uspostavljenim centrima. Ograničenja (1.4) ukazuju na binarnu prirodu promenljivih x_{ij} i y_j.

UFLP predstavlja uopštenje Veberovog problema i može se reći da predstavlja osnovni problem teorije lokacijskih problema. Poznate su različite varijante ovog problema koje se dobijaju dodavanjem i uklanjanjem nekih od ograničenja. Neke od poznatih varijanti ovog problema su:

- problem dobijen dodavanjem prioriteta koji određuju koji će uspostavljeni centar služiti kog klijenta [93, 92],
- problem dobijen uvođenjem hijerarhije centara [88, 91] i
- problem dobijen uvođenjem maksimalnog kapaciteta svakog centra [1].

Problem p-medijane

Problem p-medijane predstavlja lokacijski problem kod koga je potrebno uspostaviti tačno p centara sa ciljem minimizacije ukupne sume transportnih troškova

[54, 59, 102, 121]. Problem p-medijane spada u kategoriju *min-sum* problema. Problem p-medijane se može posmatrati i kao jedna od varijanti UFLP problema gde je unapred poznat ukupan broj centara koje treba uspostaviti p i gde ne postoji trošak uspostavljanja centara.

Koristeći ranije uvedenu notaciju u prethodnoj sekciji, problem p-medijane se može formulisati kao:

$$min \sum_{i \in I} \sum_{i \in J} d_{ij} x_{ij} \tag{1.5}$$

pri uslovima:

$$\sum_{j \in J} x_{ij} = 1 \qquad i \in I, \tag{1.6}$$

$$x_{ij} \leq y_j \qquad i \in I, j \in J, \tag{1.7}$$

$$\sum_{j \in J} y_j = p, \tag{1.8}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad i \in I, j \in J. \tag{1.9}$$

Može se videti da je u odnosu na formulaciju UFLP problema sklonjena cena uspostavljanja centra iz funkcije cilja, kao i da je dodat uslov za otvaranje tačno p centara (1.8).

Problem p-centra

Cilj *problema p-centra* je uspostaviti p centara tako da se minimizuje maksimalna distanca između klijenta i njemu pridruženog centra [113, 104]. Ovaj problem se može posmatrati kao problem p-medijane sa razlikom da se umesto minimizacije ukupnih transportnih troškova minimizuje maksimalna distanca između klijenta i uspostavljenog centra. Problem p-centra spada u kategoriju *min-max* problema.

Neka je sa z označena maksimalna distanca između klijenta i uspostavljenog centra. Koristeći ranije uvedenu notaciju, problem p-centra se može formulisati kao:

$$min(z) \tag{1.10}$$

pri uslovima:

$$\sum_{j \in J} x_{ij} = 1 \qquad i \in I, \tag{1.11}$$

$$x_{ij} \leq y_j \qquad i \in I, j \in J, \tag{1.12}$$

$$\sum_{j \in J} y_j = p, \tag{1.13}$$

$$z \geq \sum_{j \in J} d_{ij} x_{ij} \qquad i \in I, \tag{1.14}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad i \in I, j \in J. \tag{1.15}$$

Može se videti da je u odnosu na problem p-medijane promenjena funkcija cilja i da su dodati uslovi (1.14) koji definišu promenljivu z.

Problem maksimalnog pokrivanja

Problem maksimalnog pokrivanja predstavlja lokacijski problem kod koga je potrebno uspostaviti fiksan broj centara tako da se maksimizuje količina ispunjenih zahteva klijenata [24]. Problem maksimalnog pokrivanja je mnogo puta rešavan i postoje mnoge varijante ovog problema tako da se može govoriti i o klasi lokacijskih problema koje imaju karakteristike problema pokrivanja [37, 132, 36, 86].

Neka $I = \{1, 2, 3, ..., m\}$ predstavlja skup klijenata i neka $J = \{1, 2, 3, ..., n\}$ predstavlja skup potencijalnih lokacija za izgradnju centara. Svaki klijent ima određenu količinu zahteva koje je potrebno ispuniti h_i. Neka matrica pokrivanja a_{ij} pokazuje da li centar j može da ispuni zahteve klijenta i. Tačnije,

$$a_{ij} = \begin{cases} 1, & \text{ukoliko centar } j \text{ ispunjava zahteve klijenta } i, \\ 0, & \text{inače.} \end{cases}$$

Ova vrednost se najčešće definiše na osnovu udaljenosti klijenta i centra, tj. $a_{ij} = 1$ ako je razdaljina između klijenta i centra manja od unapred definisane maksimalne razdaljine, inače je $a_{ij} = 0$. Neka su y_j celobrojne promenljive za koje važi

$$y_j = \begin{cases} 1, & \text{ukoliko je centar } j \text{ otvoren,} \\ 0, & \text{ukoliko je centar } j \text{ zatvoren.} \end{cases}$$

Neka su x_i celobrojne promenljive za koje važi

$$x_i = \begin{cases} 1, & \text{ukoliko je klijent } i \text{ pokriven uslugom nekog od uspostavljenih centara,} \\ 0, & \text{inače.} \end{cases}$$

Tada se MCLP može formulisati kao:

$$max \sum_{i \in I} h_i x_i \tag{1.16}$$

pri uslovima:

$$x_i - \sum_{j \in J} a_{ij} y_j \leq 0 \qquad i \in I, \tag{1.17}$$

$$\sum_{j \in J} y_j = p, \tag{1.18}$$

$$x_i, y_j \in \{0, 1\} \qquad i \in I, j \in J. \tag{1.19}$$

Funkcija cilja 1.16 maksimizuje ukupnu količinu ispunjenih zahteva. Uslovi 1.17 obezbeđuju da klijent i ne može biti pokriven ukoliko ga bar jedan od uspostavljenih centara ne pokriva. Uslov 1.18 ograničava otvaranje tačno p centara. Ograničenja 1.19 ukazuju na binarnu prirodu promenljivih x_i i y_j.

U [42] analizirano je oko 160 radova zasnovanih na klasi problema pokrivanja korisnika i problemi su klasifikovani u smislu modela, načina rešavanja i primene. MCLP i njegove varijante su proučavani u mnogim radovima [24, 47, 23, 14].

1.4 Metaheuristike

Metaheuristike predstavljaju vrstu aproksimativnih algoritama u kojima se kombinuju osnovne heuristike u metode višeg nivoa u cilju efikasnog pretraživanja prostora rešenja i rešavanja problema optimizacije. Naziv metaheuristika je prvi put uveden 1986. godine u [50]. Naziv je izveden iz dve grčke reči: *metá*, što znači „iznad, na višem nivou" i *heuriskein*, što znači „pronalazak". Pre prihvatanja naziva metaheuristika u istraživanjima se najčešće koristio termin moderna heuristika [116].

Pojam metaheuristike je formalno definisan u više različitih istraživanja. Tako je 1996. godine u [110] metaheuristika definisana kao iterativni proces koji navodi pretragu kombinujući različite koncepte istraživanja prostora pretrage. U radu [131]

iz 1998. godine je metaheuristika definisana kao strategija visokog nivoa koja navodi osnovne heuristike specifične za problem koji se rešava, radi povećanja njihovih performansi, dok je 2012. godine u [138] metaheuristika definisana kao iterativni vodeći proces koji navodi i menja podređene heuristike kako bi efikasno pronalazio visoko kvalitetna rešenja.

Imajući u vidu različita istraživanja o metaheuristikama sa teorijskog i praktičnog aspekta, mogu se izdvojiti sledeća osnovna svojstva metaheuristika [15]:

- metaheuristike su metode koje navode proces pretrage,
- metaheuristike efikasno pretražuju prostor pretrage sa ciljom pronalaska optimalnog rešenja ili rešenja bliskog optimalnom rešenju,
- osnovne komponente metaheuristika variraju od jednostavne lokalne pretrage, do složenih procesa učenja,
- metaheuristike su aproksimativni i uglavnom nedeterministički algoritmi,
- metaheuristike koriste razne mehanizme da ne ostanu zarobljene u ograničenom delu prostora pretrage,
- osnovni koncepti metaheuristika su opisani na apstraktnom nivou,
- metaheuristike nisu posebno dizajnirane za jedan konkretan problem i
- metaheuristike mogu koristiti specifičnu heuristiku za rešavanje određenog problema, koju kontroliše opšta strategija višeg nivoa.

Za svaku metaheuristiku ključan je balans između diverzifikacije i intenzifikacije procesa pretrage. Diverzifikacija predstavlja istraživanje prostora pretrage, dok intenzifikacija predstavlja iskorišćavanje iskustva dobijenog u pretrazi. Drugim rečima mehanizam diverzifikacije je zadužen da brzo identifikuje regione prostora pretrage koji sadrže visoko kvalitetna rešenja, dok intenzifikacija predstavlja pretragu za što boljim rešenjem u okviru jednog regiona.

Postoje različite podele metaheuristika u zavisnosti od određenih karakteristika. Jedna podela metaheuristika ih deli na one koje su inspirisane prirodnim pojavama i na one koje to nisu. Tako, na primer, u algoritme inspirisane prirodnim pojavama spadaju genetski algoritam i algoritam inspirisan kolonijom mrava, dok u druge spadaju tabu pretraga i metoda promenljivih okolina. Iako intuitivna, ova podela nije previše značajna iz dva razloga. Prvo, postoje hibridni algoritmi koji spadaju

u obe ove klase istovremeno, na primer memetski algoritmi. Drugo, na osnovu ove podele, osim pripadnosti jednoj od ove dve klase, teško je zaključiti bilo koju drugu osobinu algoritma. Značajnija podela metaheuristika je na populacione i na one koje su vođene jednim rešenjem.

Metaheuristike vođene jednim rešenjem

Metaheuristike vođene jednim rešenjem se često nazivaju i metodama putanje ili metodama zasnovanim na lokalnoj pretrazi. U ove metaheuristike spadaju iterativna lokalna pretraga (engl. Iterated Local Search - ILS), tabu pretraga (engl. Tabu Search - TS), metoda promenljivih okolina (engl. Variable Neighborhood Search - VNS), metoda simuliranog kaljenja (engl. Simulated Annealing - SA), nasumično pohlepna adaptivna pretraga (engl. Greedy Randomized Adaptive Search Procedure - GRASP), itd. Ove metaheuristike se sastoje od mehanizma kreiranja početnog rešenja i mehanizma pomeranja iz tekućeg rešenja u naredno rešenje. Pomeranje zavisi od tekućeg rešenja, a često se koristi i iskustvo dobijeno u dosadašnjoj pretrazi. Na ovaj način se kreira putanja u prostoru pretrage, čije karakteristike pružaju informacije o ponašanju i efikasnosti same metaheuristike. Metaheuristike vođene jednim rešenjem najčešće sadrže neki oblik heuristike lokalne pretrage [78]. Lokalna pretraga se uglavnom sastoji od malih (lokalnih) pomeranja tekućeg rešenja sve dok se ne pronađe lokalni minimum ili dok se ne zadovolji neki drugi uslov. U nastavku je dat kratak pregled najznačajnijih metaheuristika vođenih jednim rešenjem, dok će metoda promenljivih okolina biti detaljno izložena u sekciji 2.

Iterativna lokalna pretrage

Iterativna lokalna pretraga predstavlja metaheuristiku zasnovanu na ponavljanju lokalne pretrage sa idejom povećavanja kvaliteta uzastopnih dobijenih rešenja. Početna rešenja mogu biti kreirana na slučajan način ili se za njihovo kreiranje mogu iskoristiti informacije dobijene u prethodnim iteracijama pretrage. Prvo se izvršava lokalna pretraga nad početnim rešenjem i dobija se odgovarajući lokalni minimum. Nakon toga se u svakoj iteraciji metode primenjuje pomeranje od lokalnog minimuma i izvršava se lokalna pretraga nad novodobijenim rešenjem. Novi lokalni minimum se pod određenim uslovima prihvata kao trenutno rešenje. Navedeni proces se ponavlja sve dok se ne ispuni uslov zaustavljanja. Ovakav pristup

prvi put je primenjen u radu [97], a zatim i generalizovan u [82, 131]. Osnovna šema ILS metode prikazana je algoritmom 1.

Algoritam 1: Iterativna lokalna pretraga

Ulaz: x
Izlaz: x_{best}
$x \leftarrow lokalnaPretraga(x)$;
$x_{best} \leftarrow x$;
while *uslov zaustavljanja nije ispunjen* **do**
  $x' \leftarrow pomeranje(x, istorijaPretrage)$;
  $x' \leftarrow lokalnaPretraga(x')$;
  if $f(x') < f(x_{best})$ **then**
    $x_{best} \leftarrow x'$;
  end
  $x \leftarrow prihvatanje(x, x', istorijaPretrage)$;
end

Operator pomeranja u ILS metodi uglavnom kreira blisko rešenje trenutnom rešenju, tako što se deo trenutnog rešenja zadržava, dok se drugi deo menja u zavisnosti od istorije, tj. iskustva dobijenog u toku izvršavanja algoritma. Kriterijum prihvatanja rešenja određuje da li novi lokalni minimum zadovoljava uslove da zameni trenutno rešenje. Za donošenje ove odluke se uglavnom koriste podaci dobijeni iz istorije pretrage.

Metoda simuliranog kaljenja

Metoda simuliranog kaljenja je dobila naziv prema svojoj analogiji sa fizičkim kaljenjem čvrstih tela. Ideja SA metode je da se iskoristi proces zagrevanja i sporog hlađenja čvrstih kristalnih tela, koja na kraju tog procesa dostižu najbolju molekularnu konfiguraciju kristalne rešetke. Na ovaj način tela prelaze u minimalna energetska stanja i veoma su otporna na fizičke defekte. Metoda SA je zasnovana na ovom procesu, za koji se može reći da predstavlja vezu između termodinamike i metode za rešavanje problema matematičke optimizacije. Osnovna šema metode simuliranog kaljenja prikazana je algoritmom 2.

Metoda SA koristi početno rešenje koje može biti dobijeno na slučajan način ili kreirano nekom heuristikom. U svakoj iteraciji ove metode, slučajno se generiše naredno rešenje koje se nalazi u okolini trenutnog. Ukoliko je naredno rešenje bolje od prethodnog ono se prihvata i pretraga se nastavlja od boljeg rešenja. Ukoliko

Algoritam 2: Metoda simuliranog kaljenja

Ulaz: x
Izlaz: x_{best}
$x_{best} \leftarrow x$;
$T \leftarrow početnaTemperatura()$;
while *uslov zaustavljanja nije ispunjen* **do**
  $k_{max} \leftarrow$ broj iteracija na temperaturi T;
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $x' = slučajnoRešenjeUOkolini(x)$;
    $\delta \leftarrow f(x') - f(x)$;
    if $\delta \leq 0$ **then**
      $x \leftarrow x'$;
    else
      $x \leftarrow x'$ sa verovatnoćom $p(\delta, T)$;
    end
    if $f(x) < f(x_{best})$ **then**
      $x_{best} \leftarrow x$;
    end
    $k \leftarrow k + 1$;
  end
  $T \leftarrow smanjiTemperaturu(T)$;
end

naredno rešenje nije bolje od prethodnog, odluka o njegovom prihvatanju se donosi na osnovu odgovarajuće verovatnoće prihvatanja lošijeg rešenja. Uloga mehanizma prihvatanja lošijih rešenja je izbegavanje lokalnih minimuma u potrazi za globalnim minimumom. Verovatnoća prihvatanja lošijeg rešenja zavisi od kvaliteta rešenja i trenutne temperature. Kako se temperatura vremenom smanjuje tako se i ova verovatnoća smanjuje i lošija rešenja se ređe prihvataju. Jedan od načina da se definiše ova verovatnoća je korišćenjem funkcije $e^{-\delta/T}$, gde je $\delta = f(x_{naredno}) - f(x_{prethodno})$, a T trenutna temperatura. Može se primetiti vrednost ove funkcije opada kada δ raste, kao i da opada sa smanjivanjem temperature. Na kraju svake iteracije SA metode se smanjuje trenutna temperatura. Način i brzina smanjivanja temperature zavise od korišćene šeme hlađenja. Neka je T_i vrednost temperature u i-toj iteraciji metode SA. Šema hlađenja mora da zadovoljava sledeće uslove:

- $\forall i \in \mathbb{N} : T_i > 0$,
- $\forall i \in \mathbb{N} : T_i \geq T_{i+1}$,

- $\lim_{i\to\infty} T_i = 0$.

Tabu pretraga

Tabu pretragu je prvi predložio Fred Glover 1989. godine u [52]. Ideja TS je da obezbedi klasičnoj lokalnoj pretrazi mehanizam za izbegavanje lokalnih minimuma. U lokalnoj pretrazi trenutno rešenje se pomera ka susedu, ukoliko je sused bolji od trenutnog rešenja. U tabu pretrazi su dozvoljena pomeranja i ka lošijim rešenjima. Preciznije, ukoliko je trenutno rešenje lokalni minimum, dozvoljeno je pomeriti se ka najboljem susedu, iako je najbolji sused lošiji od trenutnog rešenja. Da bi se u ovom procesu izbegli ciklusi, bitno je da novo prihvaćeno rešenje nije bilo u bliskoj istoriji posećeno. Ovaj proces se može opisati i kao pamćenje puta kojim je algoritam došao do lokalnog minimuma i izbegavanje tog puta u daljoj pretrazi. Navedeni mehanizam se postiže korišćenjem kratkoročne memorije, odnosno tabu liste. Osnovna šema tabu pretrage prikazana je algoritmom 3.

Algoritam 3: Tabu pretraga

Ulaz: x
Izlaz: x_{best}
$x_{best} \leftarrow x$;
$TL \leftarrow$ prazna lista;
while *uslov zaustavljanja nije ispunjen* **do**
  $x = najboljiSusedKojiNijeTabu(x, TL)$;
  if $f(x) < f(x_{best})$ **then**
    $x_{best} \leftarrow x$;
  end
  $TL \leftarrow ažurirajTabuListu(TL, x)$;
end

Tabu lista je uglavnom niz konstantne dužine koji čuva prethodno posećena rešenja ili prethodno izvršena pomeranja. Ako tabu lista predstavlja prethodno izvršena pomeranja, tada svaki element liste sadrži niz atributa rešenja koji su se promenili i u narednim koracima algoritma se sprečavaju inverzna pomeranja. Ovim procesom je moguće odbaciti rešenje koje do sada nije bilo posećeno. Da bi se to sprečilo nekada je dozvoljeno prihvatiti tabu rešenje, ukoliko je zadovoljen aspiracijski uslov. Često korišćeni aspiracijski uslov je da je dozvoljeno tabu pomeranje ako ono vodi do boljeg rešenja do tada od najboljeg pronađenog rešenja. Drugi aspiracijski uslov može

biti dozvola tabu pomeranja ako ono vodi do boljeg rešenja u odnosu na aktivno rešenje u trenutku dodavanja tabu elementa u listu.

Metoda TS može sadržati i napredne mehanizme intenzifikacije i diverzifikacije. Intenzifikacija u TS je zasnovana na srednjoročnoj memoriji i čuvanju elitnog skupa najboljih rešenja dobijenih tokom pretrage. Ova elitna rešenja mogu da navode pretragu, tako što će pomeranja ka rešenjima sličnim njima imati veći prioritet. Diverzifikacija u TS je zasnovana na dugoročnoj memoriji u kojoj su sačuvane informacije o posećenim rešenjima tokom pretrage. Ideja je da se na ovaj način posećeni regioni izbegavaju i da se istražuju neposećeni delovi prostora pretrage.

Nasumično pohlepna adaptivna pretraga

Nasumično pohlepna adaptivna pretraga je prvi put predstavljena 1989. godine od strane Fea i Resendea u radu [43]. GRASP je metaheuristika vođena jednim rešenjem i sastoji se u ponavljanju dva osnovna koraka: pohlepno-adaptivne konstrukcije novog rešenja i lokalne pretrage. U konstruktivnoj fazi algoritma se iterativno gradi novo rešenje, odnosno parcijalnom rešenju se dodaje jedan po jedan element rešenja dok se rešenje u potpunosti ne konstruiše. Na primer ako je rešenje definisano nizom atributa $x = (x_1, x_2, ..., x_n)$, tada će atributi redom dobijati vrednosti, prvo x_1, zatim x_2, itd. sve dok se ne kreira celo rešenje. U svakom koraku ove iteracije računaju se sve moguće vrednosti koje atribut x_i može da dobije i one predstavljaju listu kandidata (engl. Candidate List). Iz liste kandidata se izdvaja podskup najboljih kandidata (engl. Restricted Candidate List - RCL). RCL zavisi od inkrementalnog troška (u smislu funkcije cilja) koji nastaje dodelom vrednosti atributu x_i. RCL može biti ograničen na fiksan broj najboljih elemenata ili može biti ograničen nekom vrednošću, na primer $c_{min} + \alpha(c_{max} - c_{min})$, gde su c_{min} i c_{max} redom najmanji i najveći inkrementalni trošak cele liste kandidata, a α unapred zadati parametar. Na kraju se iz RCL liste slučajno bira vrednost koju će dobiti atribut x_i. Može se reći da konstruktivna faza ima tri osobine:

- pohlepna - prilikom dodele vrednosti atributu x_i rešenja x u obzir se uzimaju samo one vrednosti koje će kreirati dobra parcijalna rešenja,
- nasumična - od vrednosti koje su uzete u obzir, nasumično se bira jedna i
- adaptivna - vrednost koju dobije odgovarajući atribut je vođena prethodno dodeljenim vrednostima drugih atributa.

Osnovna šema GRASP metode prikazana je algoritmom 4.

Algoritam 4: Nasumično pohlepna adaptivna pretraga

Izlaz: x_{best}
$x_{best} \leftarrow null$;
while *uslov zaustavljanja nije ispunjen* **do**
  $x \leftarrow$ prazan niz;
  while *nije završena konstrukcija rešenja* x **do**
    $RCL \leftarrow kreirajRCL(x)$;
    $e \leftarrow slučajnoOdaberi(RCL)$;
    $x_i \leftarrow e$;
  end
  $x \leftarrow lokalnaPretraga(x)$;
  if $f(x) < f(x_{best})$ **then**
    $x_{best} \leftarrow x$;
  end
end

Populacione metaheuristike

Za razliku od metaheuristika vođenih jednim rešenjem, *populacione metaheuristike* u svakoj iteraciji rade na skupu rešenja koji se naziva populacija. Ideja populacionih metaheuristika je da kroz uzastopne iteracije algoritma populacija napreduje i da se tako dobijaju rešenja koja su sve bliža globalnom optimumu. Performanse ovih metoda zavise isključivo od načina manipulacije populacijama. Populacione metaheuristike su uglavnom zasnovane na ponašanju i organizaciji skupa jedinki u prirodi (mravi, pčele, ptice,...) ili na nekim biološkim procesima (evolucija). Neke od najpoznatijih populacionih metaheuristika su genetski algoritmi (engl. Genetic Algorithms - GA), metoda rasute pretrage (engl. Scatter Search - SS), metoda zasnovana na koloniji mrava (engl. Ant Colony Optimization - ACO), metoda zasnovana na rojevima čestica (engl. Particle Swarm Optimization), metoda zasnovana na elektromagnetizmu (engl. Electromagnetism Metaheuristic), itd. U nastavku je dat kratak pregled najznačajnijih populacionih metaheuristika.

Evolutivni algoritmi

Evolutivni algoritmi predstavljaju jednu klasu populacionih metaheuristika [9, 127]. Prilikom pretrage prostora rešenja koriste mehanizme zasnovane na biološkoj

evoluciji jedne ili više jedinki populacije kao što su prirodna selekcija, ukrštanje, mutacija, prilagođenost, migracija, uklanjanje dela populacije, itd. Najpoznatiji evolutivni algoritmi su *genetski algoritmi* [79]. Kod genetskih algoritama, svaka jedinka u populaciji odgovara jednom rešenju u prostru pretrage. Osnovna ideja je da se, slično biološkom napretku jedne vrste kroz generacije, poboljšavaju i odgovarajuća rešenja problema u smislu vrednosti funkcije cilja. Osnovna šema genetskih algoritama prikazana je algoritmom 5. U svakoj iteraciji, genetski algoritam radi nad skupom jedinki koji predstavlja aktivnu populaciju. Iz populacije se operatorom selekcije biraju jedinke koje se ukrštaju i tako se dobijaju potomci, odnosno nove jedinke. Nakon procesa mutacije novih jedinki računa se njihova prilagođenost. Uloga operatora prilagođenosti je da oceni vrednost svake jedinke u poređenju sa drugima. Obično bolja vrednost funkcije cilja, označava bolju prilagođenost. Osim funkcije cilja na prilagođenost mogu da utiču i jedinstvenost genetskog materijala, standardna devijacija itd. Novodobijene jedinke čine novu generaciju jedinki i predstavljaju novu populaciju jedinki. U slučaju elitističke strategije u novu generaciju se uključuju i najbolje jedinke iz trenutne populacije.

Algoritam 5: Genetski algoritam

Izlaz: x_{best}
$pop \leftarrow inicijalnaPopulacijaJedinki()$;
while *uslov zaustavljanja nije ispunjen* **do**
  $roditelji \leftarrow selekcijaJedinki(pop)$;
  $potomci \leftarrow ukrštanje(roditelji)$;
  $potomci \leftarrow mutacija(potomci)$;
  $prilagodjenost \leftarrow izračunajPrilagodjenost(potomci)$;
  $pop \leftarrow novaGeneracija(pop, potomci, prilagodjenost)$;
end
$x_{best} \leftarrow$ najbolje rešenje dobijeno u pretrazi;

U novijim istraživanjima dosta su popularne i hibridne metaheuristike koje kombinuju dve ili više različitih metaheuristika za rešavanje jednog problema [126, 90, 129]. Kao posebna klasa ovih hibridnih metoda izdvojili su se memetski algoritmi koji predstavljaju hibrid evolutivnih algoritama i lokalne pretrage [91, 94].

Metoda rasute pretrage

Metoda rasute pretrage je prvi put predstavljena 1977. godine od strane Glovera u radu [51]. Osnovna ideja SS metode je kombinovanje rešenja referentnog skupa u

potrazi za novim rešenjima. Šema SS metode prikazana je algoritmom 6.

Algoritam 6: Metoda rasute pretrage

Izlaz: x_{best}
$pop \leftarrow inicijalnaPopulacijaJedinki()$;
$pop \leftarrow metodaPopravke(pop)$;
$rs \leftarrow kreirajReferentniSkup(pop)$;
while *uslov zaustavljanja nije ispunjen* **do**
  $ss \leftarrow kreirajPodskupove(rs)$;
  $novaRešenja \leftarrow$ prazan skup;
  foreach *podskup p iz ss* **do**
    $p' \leftarrow kombinacija(p)$;
    $p' \leftarrow metodaPopravke(p')$;
    $novaRešenja \leftarrow novaRešenja \cup p'$;
  end
  $rs \leftarrow ažuriraj(rs, novaRešenja)$;
end
$x_{best} \leftarrow najboljeRešenje(rs)$;

Na početku metode se kreira inicijalna populacija rešenja, koja treba da zadovolji zadate uslove raznovrsnosti i kvaliteta. Obično se za kreiranje početne populacije koristi neka nasumična ili pohlepna metoda. Kako bi se zadovoljio uslov kvaliteta, nad svim rešenjima početne populacije se primenjuje metoda popravke. To može biti lokalna pretraga ili neka složenija metaheuristika vođena jednim rešenjem. Iz populacije se izdvajaju reprezentativna rešenja koja kreiraju referentni skup. Prilikom odabira reprezentativnih rešenja bitno je da uslovi raznovrsnosti i kvaliteta ostanu ispunjeni. To se može postići tako što, na primer, polovina referentnog skupa sadrži najbolja rešenja, a druga polovina referentnog skupa najraznovrsnija rešenja početne populacije. Veličina referentnog skupa je uglavnom relativno mala i sadrži oko deset elemenata. Rešenja iz referentnog skupa se kombinuju između sebe i na taj način nastaju nova rešenja. Prvo je potrebno odabrati podskupove rešenja koji će se kombinovati. Ovi podskupovi uglavnom sadrže 2 elementa, a proces odabira podskupova može biti sličan operatoru selekcije GA. Nova kombinovana rešenja u opštem slučaju mogu nastati linearnim kombinacijama starih rešenja koristeći odgovarajuće težinske faktore. Operator kombinacije se može posmatrati kao uopštenje operatora ukrštanja kod GA. Nad novim rešenjima dobijenim kombinacijom se primenjuje metoda popravke. Na kraju se referentni skup ažurira koristeći novodobijena rešenja, pri čemu se vodi računa da uslovi kvaliteta i raznovrsnosti budu

očuvani. Ovaj proces se iterativno nastavlja sve dok se ne ispuni uslov zaustavljanja.

Optimizacija mravljim kolonijama

Optimizacija mravljim kolonijama je prvi put predstavljena 1992. godine od strane Doriga u doktorskoj disertaciji [35]. Osnovna ideja ACO metaheuristike je primena imitacije kooperativnog ponašanja mrava za rešavanje optimizacionih problema. Tokom kretanja mravi ostavljaju na tlu određenu količinu hemijske supstance poznate kao feromon. Uloga feromona je da navodi kretanje ostalih mrava tako što prate njegov trag. Preciznije, svaki mrav sa određenom verovatnoćom preferira praćenje debljeg sloja feromona na putu. Na osnovu ove jednostavne osobine jato mrava dobija sposobnost da reši složene zadatke, kao što su pronalazak najkraće putanje do hrane ili obilazak prepreke na putu. Osnovna šema ACO metode prikazana je algoritmom 7.

Algoritam 7: Algoritam optimizacije mravljim kolonijama

Izlaz: x_{best}
$trag \leftarrow početniTragFeromona()$;
while *uslov zaustavljanja nije ispunjen* **do**
    $rešenja \leftarrow kreirajRešenja(trag)$;
    $rešenja \leftarrow lokalnaPretraga(rešenja)$;
    $trag \leftarrow primeniIsparavanje(trag)$;
    $trag \leftarrow ostaviNoviTrag(trag, rešenja)$;
end
$x_{best} \leftarrow$ najbolje rešenje dobijeno u pretrazi;

Na početku algoritma sve promenljive koje predstavljaju tragove feromona uglavnom dobijaju fiksiranu početnu vrednost. Na osnovu tragova feromona se kreiraju rešenja. Kaže se da svaki mrav kreira po jedno rešenje, tako što iterativno gradi deo po deo rešenja. Preciznije, parcijalnom rešenju se dodaje jedan po jedan element rešenja dok se rešenje u potpunosti ne konstruiše. Za iterativno građenje rešenja se koristi nasumična pohlepna metoda koja zavisi od trenutne vrednosti tragova feromona. Na sva nova rešenja se primenjuje lokalna pretraga i nakon toga se ažuriraju tragovi feromona. Ažuriranje se sastoji iz dva koraka. U prvom koraku se primenjuje operator isparavanja, dok se u drugom koraku beleže tragovi koji zavise od novih rešenja. Ovaj proces se iterativno nastavlja sve dok se ne ispuni uslov zaustavljanja.

Testiranje performansi metaheuristika

U literaturi su predložene brojne metaheuristike korišćene za rešavanje različitih problema optimizacije. Kako bi se uporedile performanse različitih metaheuristika potrebno je na neki način uporediti njihovu efikasnost prilikom rešavanja problema. Veoma je teško primetiti klasičnu teorija složenosti u ovoj analizi zbog toga što su metaheuristike iterativni nedeterministički procesi. Zato je ustaljena praksa testiranje metaheuristika na većem broju instanci odgovarajućeg problema uz detaljnu analizu dobijenih rezultata.

Višestruko pokretanje algoritma

Kako su metaheuristike uglavnom nedeterministički algoritmi, više pokretanja jedne implementacije metode za rešavanje iste instance problema može voditi različitim rezultatima. Preciznije metaheuristike u svom radu koriste pseudo-slučajne brojeve i tok rešavanja instance problema zavisi od početnog semena za generisanje pseudo-slučajnih brojeva. Radi kvalitetnijeg testiranja algoritma, obično se za svaku instancu problema program izvršava više puta sa različitim vrednostima semena i dobija se veći broj rezultata. Kao osnovna mera kvaliteta algoritma se posmatra najbolji dobijeni rezultat iz više pokretanja i vreme za koje je najbolji rezultat dobijen. Osim vremena potrebnog za dobijanje najboljeg rešenja, za analizu su bitni i ukupno vreme rada algoritma, kao i prosečna vremena dobijena iz svih pokretanja algoritma.

Kada se posmatra vrednost funkcije cilja, najbolji dobijeni rezultat ne daje potpunu sliku kvaliteta algoritma. Za potpuniju sliku potrebno je analizirati i rezultate dobijene iz ostalih izvršavanja. Na osnovu svih dobijenih rezultata može se zaključiti koliko je algoritam stabilan, odnosno koliko su rezultati po vrednostima bliski jedan drugom. Takođe može se videti i koliko su rezultati blizu najboljem dobijenom rešenju. Neka je ukupan broj izvršavanja algoritma n. Ukupna ocena kvaliteta dobijenih rešenja izvodi se računanjem *prosečnog odstupanja u odnosu na najbolje rešenje* ($agap$) i *standardnom devijacijom prosečnog odstupanja* (σ).

Vrednosti $agap$ i σ se za svaku instancu računaju prema sledećim formulama:

- $agap = \frac{1}{n} \sum_{i=1}^{n} gap_i$,
- $\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (gap_i - agap)^2}$,

gde su n broj pokretanja algoritma sa različitim početnim semenima za generisanje pseudo-slučajnih brojeva i $gap_i = 100\frac{res_i - res}{res}$. Vrednost res_i predstavlja rešenje i-tog pokretanja algoritma, dok je res optimalno rešenje, ako je ono poznato, ili najbolje poznato rešenje, ukoliko optimalno rešenje nije poznato. Što su vrednosti $agap$ i σ manje kaže se da je algoritam kvalitetniji (stabilniji).

Kriterijumi zaustavljanja

Kako su metaheuristike uglavnom iterativni algoritmi, kriterijum zaustavljanja kontroliše kada se osnovni ciklus zaustavlja. Neki od kriterijuma zaustavljanja su:

- ukupan broj iteracija,
- ukupan broj sukcesivnih iteracija bez poboljšanja rešenja,
- ukupno vreme izvršavanja i
- ukupno proteklo vreme od poslednjeg poboljšanja rešenja.

Mogu se kombinovati dva ili više kriterijuma zaustavljanja, na primer ukupan broj sukcesivnih iteracija bez poboljšanja rešenja i ukupno vreme izvršavanja da bi se dobio balans između vremena izvršavanja i kvaliteta rešenja.

Paralelno izvršavanje metaheuristika

Paralelno računarstvo predstavlja simultano izvršavanje više procesa na više procesorskih jezgara izvršavajući jedan program [3]. Može se reći da paralelizam predstavlja dekompoziciju ukupnog računarskog opterećenja i njegovu distribuciju na različite procesore. Dekompozicija se može odnositi na algoritam metaheuristike, prostor pretrage ili na strukturu problema. Tehnike paralelnog programiranja se često primenjuju za poboljšanje efikasnosti rešavanja problema optimizacije posebno u slučaju rešavanja instanci problema velikih dimenzija [111, 91, 26]. U praksi su se izdvojile tri različite strategije za paralelizaciju metaheuristika [26].

1. *Paralelizacija niskog nivoa.* Ova strategija predstavlja ubrzavanje izračunavanja tako što se paralelno izvršava neki od računarsko zahtevnih zadataka. Uglavnom se implementira *master-slave* model. *Master* proces nadgleda rad *slave* procesa i preračunava globalni rezultat na osnovu dobijenog rezultata od

svakog *slave* procesa. Kod paralelizacije niskog nivoa, osnovni izvor paralelizacije je paralelno izvršavanje ciklusa, kao što su, na primer, paralelno istraživanje okoline i paralelno računanje funkcije cilja. Generalno nema mnogo mesta na kojima se paralelizacija ciklusa može primeniti, jer je većina ciklusa vremenski zavisna, odnosno, izračunavanje jednog koraka zahteva izračunavanje prethodnog koraka. Takođe, u nekim slučajevima, troškovi sinhronizacije pri paralelizaciji su veliki, što čini paralelno izvršavanje nepotrebnim, jer su dobici zanemarljivi. Paralelizacija niskog nivoa ima svojih prednosti, a osnovna prednost je u tome što je rezultat paralelnog izvršavanja algoritma često jednak rezultatu izvršavanja sekvencijalnog algoritma, pod uslovom da je korišćeno isto seme za generisanje slučajnih brojeva.

2. *Dekompozicija domena.* Prostor pretrage se uglavnom dekomponuje tako što se podeli niz atributa jednog rešenja na više manjih nizova. Tada se rešavanje problema na svakom od manjih nizova izvršava u okviru različitih procesa. Ovakva podela smanjuje prostor pretrage problema i moguće je da dobijeno optimalno rešenje novog problema nije ujedno i optimalno rešenje početnog problema. Zbog toga je praksa da se procedura pretrage ponavlja na različitim podelama kako bi se omogućilo istraživanje celog prostora pretrage. Uglavnom se koristi *master-slave* model, gde *master* proces kreira dekompoziciju, a *slave* procesi izvršavaju metaheuristiku na manjim problemima. *Master* proces prikuplja parcijalna rešenja i od njih konstruiše globalno rešenje.

3. *Višestruka pretraga.* Ova strategija predstavlja paralelizaciju dobijenu od više paralelnih istraživanja celog prostora pretrage. Paralelne pretrage mogu izvršavati iste ili različite heuristike, koje mogu započeti pretragu sa istim ili različitim rešenjima. Ove heuristike mogu razmenjivati informacije u toku pretrage i to se naziva kooperativna višestruka pretraga. U suprotnom, kada heuristike razmenjuju informacije samo na kraju da bi se pronašlo najbolje globalno rešenje, pretraga se naziva nezavisna višestruka pretraga. U kooperativnoj pretrazi se razlikuju sinhrona i asinhrona komunikacija. Kod sinhorne komunikacije u jednom trenutku (nakon određenog broja iteracija ili nakon vremenskog intervala ili u određenom stanju algoritma) se svi procesi zaustave radi deljenja informacije. Kod asinhrone komunikacije svaki proces sam odlučuje kada će pristupiti deljenim informacijama na osnovu kojih će doneti odluke vezane za svoje dalje izvršavanje.

U literaturi postoje brojne primene paralelnih metaheuristika za rešavanje lokacijskih problema. U radu [48] iz 2002. godine su za rešavanje problema p-medijane primenjene tri različite strategije paralelizacije metode promenljivih okolina: paralelizacija niskog nivoa, nezavisna višestruka pretraga i kooperativna višestruka pretraga sa sinhronom komunikacijom. Eksperimentalni rezultati su pokazali da su strategije višestruke pretrage imale bolje performanse u odnosu na paralelizaciju niskog nivoa. U radu [26] iz 2004. godine je predstavljena paralelizacija VNS metode kooperativnom pretragom sa asinhronom komunikacijom za rešavanje problema p-medijane. Paralelni algoritam je testiran koristeći vremensko ograničenje od t_{max}/np sekundi, gde t_{max} predstavlja konstantno vreme, a np broj korišćenih procesora. Eksperimentalni rezultati su pokazali da rezultati postaju lošiji sa povećanjem vrednosti np. U radu [91] iz 2014. godine predstavljena je kombinovana strategija za paralelizaciju memetskog algoritma za rešavanje lokacijskog problema snabdevača neograničenog kapaciteta u više nivoa. Evolutivni deo algoritma je paralelizovan strategijom niskog nivoa, dok je za paralelizaciju lokalne pretrage korišćena nezavisna višestruka pretraga.

Merenje performansi paralelnih algoritama

Za merenje performansi paralelnih algoritama mogu biti korišćene različite metrike, među kojima je najznačajnija metrika ubrzanja paralelne verzije algoritma u odnosu na sekvencijalnu. Preciznije, ukoliko je sa T_m označeno vreme izvršavanja algoritma koji koristi m procesora, ubrzanje se može definisati na sledeći način

$$s_m = \frac{T_1}{T_m}.$$

Osnovnu poteškoću pri ovakvoj definiciji ubrzanja predstavlja različita interpretacija pojmova T_1 i T_m [2, 12]. Naime, pojam T_1 se može posmatrati kao vreme izvršavanja najboljeg postojećeg sekvencijalnog algoritma za rešavanje datog problema. Zbog problema definisanja i pronalaska najboljeg postojećeg sekvencijalnog algoritma, ova mera se u praksi ne koristi. Druga moguća interpretacija pojma T_1 je vreme izvršavanja osnovne sekvencijalne verzije algoritma za koju se kreira paralelna verzija. Treća interpretacija pojma T_1 je vreme izvršavanja predložene paralelne verzije algoritma koja se izvršava na jednom procesoru.

Sa druge strane, za računanje vremena izvršavanja algoritma na m procesora, T_m, kritično je definisanje momenta u kome se zaustavlja merenje vremena. Mogu

se koristiti dva uslova: dostignut je potreban kvalitet rešenja i dostignuta je maksimalna količina izračunavanja. Maksimalna količina izračunavanja se može definisati brojem iteracija algoritma ili dozvoljenim vremenom za izvršavanje algoritma.

Važno je napomenuti da paralelne metaheuristike treba da daju rešenja veoma bliska sekvencijalnim metaheuristikama kako bi njihovo poređenje i merenje ubrzanja imalo smisla u bilo kojoj od navedenih interpretacija.

Glava 2

Metoda promenljivih okolina

U ovoj glavi izloženi su metoda promenljivih okolina, njeni osnovni teorijski koncepti kao i neke od varijanti ove metode koje su uspešno primenjene za rešavanje različitih problema matematičke optimizacije.

Metoda promenljivih okolina (engl. Variable Neighborhood Search - VNS) je metaheuristika vođena jednim rešenjem koja se koristi za rešavanje problema kombinatorne i globalne optimizacije. Osnovna ideja metode je sistematizovana zamena okolina u fazi pretrage za lokalnim minimumom i u fazi izlaska iz lokalnog minimuma u pretrazi za globalnim minimumom. Metoda VNS je predložena 1997. godine od strane Mladenovića i Hansena u radu [101]. Metoda promenljivih okolina je od tada primenjena za rešavanje mnogih klasa problema matematičke optimizacije [66].

2.1 Okoline prostora pretrage

Neka je dat problem matematičke optimizacije definisan skupom S, funkcijom cilja $f : S \to \mathbb{R}$ i dopustivim skupom $X \subseteq S$. *Strukturu okoline* N, ili kraće okolinu N, prostora pretrage X definišemo kao funkciju

$$N : X \to \mathcal{P}(X)$$

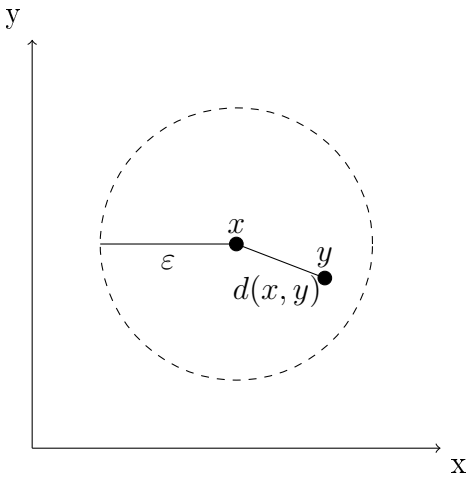
gde $\mathcal{P}(X)$ predstavlja partitivni skup skupa X. Okolinu možemo posmatrati kao preslikavanje koje jedno rešenje $x \in X$ preslikava u skup rešenja $N(x) \subseteq X$. Elemente skupa $N(x)$ nazivamo susedima rešenja x. U praksi skup $N(x)$ predstavlja rešenja koja su na neki način bliska rešenju x.

U matematičkoj optimizaciji skup S je često metrički prostor u kome je definisana metrika udaljenosti

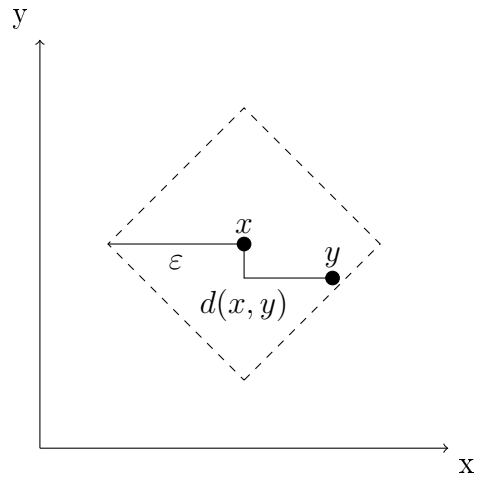
$$d : S \times S \to \mathbb{R}$$

koja definiše udaljenost svaka dva elementa skupa S. Tada se bliskost između dva rešenja može definisati na osnovu njihove udaljenosti. Na primer ε-okolina rešenja $x \in X$ predstavlja skup rešenja $\{y \in X | d(x, y) < \varepsilon\}$.

Neka je $X = \mathbb{R}^2$ i neka su $x, y \in X$. Može se reći da se y nalazi u ε-okolini tačke x ukoliko važi $d(x, y) < \varepsilon$. Kao metrika udaljenosti se može koristi na primer metrika euklidskog rastojanja $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ (slika 2.1) ili na primer metrika zasnovana na Menhetn rastojanju $d(x, y) = |x_1 - y_1| + |x_2 - y_2|$ (slika 2.2).



Slika 2.1: Okolina tačke x u $\mathbb{R}^2$ zasnovana na euklidskom rastojanju



Slika 2.2: Okolina tačke x u $\mathbb{R}^2$ zasnovana na Menhetn rastojanju

Prostor rešenja X može biti konačan diskretan skup, na primer $X = \{0, 1\}^n$, odnosno $X = \{(x_1, x_2, ..., x_n) | x_i \in \{0, 1\}, i = 1, 2, ..., n\}$. Najčešće korišćena metrika nad ovim prostorom je Hamingova metrika rastojanja [58]:

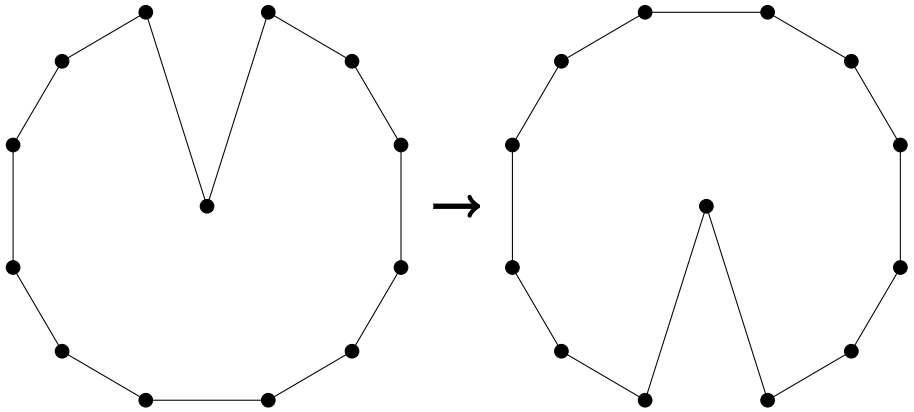
$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|.$$

Tada se mogu definisati ugnježdene okoline $N_1(x) = \{y \in X | d(x, y) \leq 1\}$, $N_2(x) = \{y \in X | d(x, y) \leq 2\}$, itd.

Često je potrebno da broj jedinica u rešenju $x \in \{0, 1\}^n$ bude fiksan broj k. Tada je prostor pretrage jednak $X = \{(x_1, x_2, ..., x_n) | \sum_{i=1}^{n} x_i = k\}$. Kao metrika

udaljenosti se obično koristi minimalni broj razmena da bi se od rešenja x dobilo rešenje y. Jedna razmena podrazumeva zamenu jedne nule sa jednom jedinicom i zamenu jedne jedinice sa jednom nulom u rešenju x.

Posebno su interesantne strukture okolina se pojavljuju kod problema trgovačkog putnika (engl. Traveling Salesman Problem - TSP). Poznati su skup lokacija $\{l_1, l_2, ..., l_n\}$ i razdaljine između svake dve lokacije. Potrebno je odrediti najkraći put kojim trgovački putnik treba da se kreće tako da obiđe sve lokacije i da se vrati na početnu lokaciju od koje je krenuo. Prostor rešenja predstavlja skup svih permutacija brojeva $1, 2, ..., n$ bez ponavljanja. Jedna permutacija predstavlja redosled lokacija u kom trgovački putnik treba da ih obiđe. Za rešavanje TSP problema obično se koriste okoline koje se dobijaju operatorom realokacije (slika 2.3), 2-opt i 3-opt operatorima (slika 2.4 i 2.5) kao i operatorom dvostrukog mosta (slika 2.6) [105].



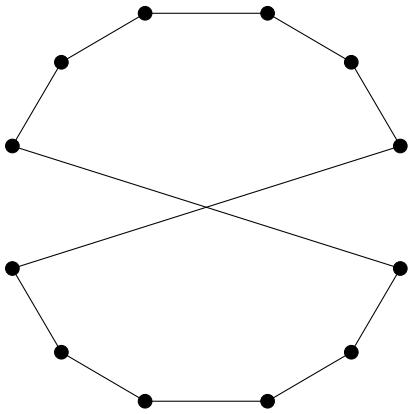
Slika 2.3: TSP - Operator realokacije

2.2 Osnovni koncepti metode promenljivih okolina

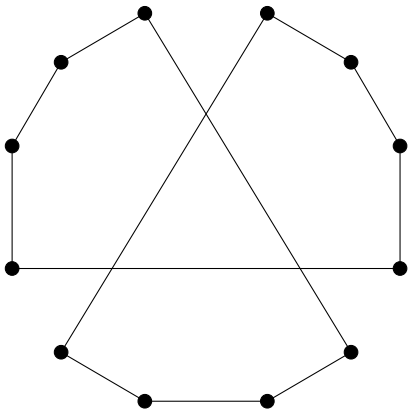
Neka su dati skup S, funkcija cilja $f : S \to \mathbb{R}$ i dopustiv skup $X \subseteq S$. Niz unapred odabranih struktura okolina dopustivog skupa X se označava sa $N_k, k = 1, 2, ..., k_{max}$. Obično su uzastopne okoline N_k ugnježđene i mogu biti izvedene iz jedne ili više metrika definisanih nad skupom S.

Metoda VNS je zasnovana na tri jednostavne činjenice:

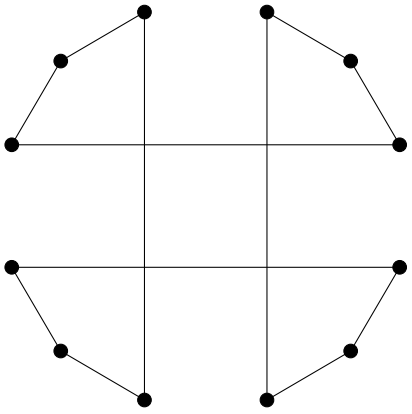
- lokalni minimum u jednoj okolini ne mora biti lokalni minimum u drugoj okolini,



Slika 2.4: TSP - 2-opt operator



Slika 2.5: TSP - 3-opt operator



Slika 2.6: TSP - operator dvostrukog mosta

- globalni minimum je lokalni minimum u svim okolinama,
- u mnogim problemima lokalni minimumi, u jednoj ili više okolina, su relativno blizu.

Poslednja, empirijska činjenica, podrazumeva da lokalni minimum često pruža neke informacije o globalnom minimumu. Pretpostavimo da se jedno rešenje $x \in X$ može zapisati kao niz atributa $x = (x_1, x_2, ..., x_n)$. Tada se za dva različita lokalna minimuma pretpostavlja da imaju značajan broj atributa sa istim vrednostima, pri čemu se unapred ne zna koji su to atributi. Iz tog razloga, neophodno je sprovesti sistematičnu pretragu okolina oko lokalnog minimuma, sve dok se ne pronađe bolje rešenje, odnosno novi bolji lokalni minimum.

Postoje različite determinističke, stohastičke i kombinovane metode za rešavanje problema matematičke optimizacije zasnovane na pretrazi okolina rešenja prostora

pretrage.

2.3 Funkcije zamena okolina, razmrdavanje i zadrži bolje rešenje

Jedna od funkcija koja se koristi u metodi promenljivih okolina je *zamenaOkolina*. Ona poredi trenutno rešenje x sa novim rešenjem x' koje se nalazi u njegovoj k-toj okolini. Ukoliko je postignuto poboljšanje, odnosno ukoliko je rešenje x' bolje od rešenja x, trenutno rešenje se ažurira i vrednost k se postavlja na inicijalnu vrednost 1, u suprotnom se posmatra sledeća okolina. Funkcija *zamenaOkolina* prikazana je algoritmom 8.

Algoritam 8: Zamena okolina

Ulaz: x, x', k
Izlaz: x, k
if $f(x') < f(x)$ **then**
  $x \leftarrow x'$;
  $k \leftarrow 1$;
else
  $k \leftarrow k + 1$;
end

Sledeća važna funkcija naziva se *razmrdavanje* i njena uloga je da slučajno generiše novo rešenje x' u k-toj okolini rešenja x. Ova funkcija je predstavljena algoritmom 9.

Algoritam 9: Razmrdavanje

Ulaz: x, k
Izlaz: x'
$x' \leftarrow y, y \in N_k(x)$;

Funkcija *zadržiBoljeRešenje* zadržava bolje od dva rešenja x ili x' (algoritam 10).

Algoritam 10: Zadrži bolje rešenje

Ulaz: x, x'
Izlaz: x
if $f(x') < f(x)$ **then**
  $x \leftarrow x'$;
end

2.4 Metoda promenljivog spusta

Metoda promenljivog spusta (engl. Variable Neighborhood Descent - VND) je metoda zasnovana na determinističkoj promeni okolina. Funkcija VND je predstavljena algoritmom 11. Ideja pretrage je da se prvo posmatraju poboljšanja u N_1 okolini. Ukoliko postoji poboljšanje primenjuje se zamena okolina i pretraga se nastavlja od novog boljeg rešenja. Kada je dostignut lokalni minimum u N_1 okolini, posmatraju se poboljšanja u narednoj okolini N_2. Kada je dostignut lokalni minimum u N_2, posmatraju se dalja poboljšanja u N_3 i tako dalje. Nakon svake uspešne zamene okolina, odnosno nakon svakog pronađenog poboljšanja, pretraga se vraća nazad na okolinu N_1.

Algoritam 11: Metoda promenljivog spusta

Ulaz: x, k_{max}
Izlaz: x
$k \leftarrow 1$;
while $k \leq k_{max}$ **do**
  $x' \leftarrow argmin_{y \in N_k(x)} f(y)$;
  $(x, k) \leftarrow zamenaOkolina(x, x', k)$;
end

Većina heuristika lokalne pretrage koriste mali broj okolina za unapređenje trenutnog rešenja. Kod VND metode se može primetiti da je konačno rešenje lokalni minimum u odnosu na svih k_{max} okolina, i zbog toga se sa povećanjem broja okolina povećava šansa za pronalazak globalnog minimuma. Osim sekvencijalnog poretka okolina u VND algoritmu, može se primeniti i strategija ugnježdenih okolina. Na primer, jedna od ugnježdenih strategija se može realizovati primenom VND metode sa $k_{max} = 2$ na svako rešenje koje se nalazi u trećoj okolini tekućeg rešenja ($x' \in N_3(x)$). Ovaj pristup primenjen je uspešno u [18, 64].

2.5 Redukovana metoda promenljivih okolina

Redukovana metoda promenljivih okolina (engl. Reduced Variable Neighborhood Search - RVNS) zasnovana je na slučajnoj pretrazi u odabranoj okolini $N_k(x)$ bez primene lokalne pretrage ili VND algoritma. Novodobijeno rešenje se poredi sa trenutnim, i ukoliko je ostvareno poboljšanje, vrši se zamena. Za potrebe algoritma potrebno je definisati uslov zaustavljanja. RVNS je predstavljen algoritmom 12.

Algoritam 12: Redukovana metoda promenljivih okolina

Ulaz: x, k_{max}
Izlaz: x
while *uslov zaustavljanja nije ispunjen* **do**
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $x' \leftarrow razmrdavanje(x, k)$;
    $(x, k) \leftarrow zamenaOkolina(x, x', k)$;
  end
end

RVNS se najčešće primenjuje za rešavanje instanci veoma velikih dimenzija gde lokalna pretraga, odnosno detaljna pretraga okolina, nije dovoljno efikasna. Takođe, RVNS algoritam može da se iskoristi za brzo dobijanje početnog rešenja nekog drugog algoritma. Vrednost k_{max} kod RVNS algoritma obično je mali broj između 2 i 5. RVNS je srodan sa Monte Karlo metodom, ali je sistematičniji, jer za razliku od Monte Karlo metode, RVNS navodi pretragu na osnovu definisanih parametara.

2.6 Osnovna metoda promenljivih okolina

Osnovna metoda promenljivih okolina (engl. Basic Variable Neighborhood Search) predstavlja kombinaciju determinističke i stohastičke promene okolina [101]. Deterministički deo predstavlja heuristiku lokalne pretrage. Kod lokalne pretrage potrebno je pronaći bolje rešenje x' u okolini rešenja x. Ukoliko takvo rešenje postoji trenutno rešenje se ažurira i pretraga se nastavlja od novog rešenja, u suprotnom je dostignut lokalni minimum. Obično se od svih poboljšanja pronađenih u okolini bira najbolje poboljšanje što predstavlja strategiju najboljeg poboljšanja. Kako strategija najboljeg poboljšanja može biti vremenski zahtevna, alternativa je primena strategije prvog poboljšanja, kod koje se pronalaskom prvog boljeg rešenja

u okolini trenutno rešenje ažurira i pretraga nastavlja od novog trenutnog rešenja. Lokalna pretraga koja koristi strategije najboljeg (funkcija *najboljePoboljšanje*) i prvog poboljšanja (funkcija *prvoPoboljšanje*) prikazana je redom algoritmima 13 i 14.

Algoritam 13: Lokalna pretraga sa strategijom najboljeg poboljšanja

Ulaz: x
Izlaz: x
while *true* **do**
  $x' \leftarrow argmin_{y \in N_k(x)} f(y)$;
  if $f(x') < f(x)$ **then**
    $x \leftarrow x'$;
  else
    break;
  end
end

Algoritam 14: Lokalna pretraga sa strategijom prvog poboljšanja

Ulaz: x
Izlaz: x
while *true* **do**
  $x' \leftarrow$ prvo pronađeno $y, y \in N_k(x)$ i $f(y) < f(x)$;
  if x' *postoji* **then**
    $x \leftarrow x'$;
  else
    break;
  end
end

U daljem tekstu će se osnovna metoda promenljivih okolina zbog jednostavnosti uglavnom nazivati samo metodom promenljivih okolina. Metoda promenljivih okolina predstavljena je algoritmom 15.

2.7 Opšta metoda promenljivih okolina

Opšta metoda promenljivih okolina (engl. General Variable Neighborhood Search - GVNS) nastala je iz osnovne VNS metode zamenom lokalne pretrage VND metodom. Osnovna šema GVNS metode prikazana je algoritmom 16. Maksimalna

Algoritam 15: Osnovna metoda promenljivih okolina

Ulaz: x, k_{max}
Izlaz: x
while *uslov zaustavljanja nije ispunjen* **do**
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $x' \leftarrow razmrdavanje(x, k)$;
    $x'' \leftarrow najboljePoboljšanje(x')$;
    $(x, k) \leftarrow zamenaOkolina(x, x'', k)$;
  end
end

okolina VND metode obeležena je sa l_{max}. GVNS pristup je uspešno primenjen za rešavanje mnogih problema matematičke optimizacije [73, 100, 103, 106].

Algoritam 16: Opšta metoda promenljivih okolina

Ulaz: x, k_{max}, l_{max}
Izlaz: x
while *uslov zaustavljanja nije ispunjen* **do**
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $x' \leftarrow razmrdavanje(x, k)$;
    $x'' \leftarrow VND(x', l_{max})$;
    $(x, k) \leftarrow zamenaOkolina(x, x'', k)$;
  end
end

2.8 Adaptivna metoda promenljivih okolina

Adaptivna metoda promenljivih okolina (engl. Skewed Variable Neighborhood Search - SVNS) je varijanta VNS metode koja omogućava pretraživanje prostora pretrage daleko od trenutnog rešenja. Naime nekada trenutno najbolje rešenje predstavlja lokalni minimum u većem regionu prostora pretrage, i tada je neophodno pomeriti se daleko u potrazi za poboljšanjem. Jedan od pristupa za rešavanja ovog problema je višestruko pokretanje (engl. multistart) VNS algoritma sa različitim slučajno generisanim početnim rešenjima, u nadi da će neke od pretraga izbeći kritični region. Problem u ovom pristupu je što pretraga zavisi od slučajno generisanih

početnih rešenja. Kao rešenje ovog problema predložen je SVNS koji omogućava pretraživanje prostora rešenja daleko od trenutnog rešenja na sistematizovan način. Struktura SVNS metode je predstavljena algoritmom 17.

Algoritam 17: Adaptivna metoda promenljivih okolina

Ulaz: x, k_{max}, α
Izlaz: x
$x_{best} \leftarrow x$;
while *uslov zaustavljanja nije ispunjen* **do**
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $x' \leftarrow razmrdavanje(x, k)$;
    $x'' \leftarrow prvoPoboljšanje(x')$;
    $(x, k) \leftarrow zamenaOkolinaSVNS(x, x'', k, \alpha)$;
  end
  $x_{best} \leftarrow zadrziBoljeRešenje(x_{best}, x)$;
  $x \leftarrow x_{best}$;
end

Metoda SVNS umesto klasične funkcije zamene okolina koristi specifičnu funkciju *zamenaOkolinaSVNS* (algoritam 18). Funkcija *zamenaOkolinaSVNS* predstavlja pomeranje rešenja u slučaju da su ispunjeni odgovarajući uslovi. Ona koristi funkciju $\rho(x, x')$ za merenje udaljenosti između dva rešenja koja može biti jednaka metrici udaljenosti korišćenoj za definisanje strukture okolina N_k. Parametar α se bira tako da omogući pomeranja u okoline koje su daleko od x i u slučajevima kada je rešenje x' lošije od rešenja x. Što je vrednost α manja, to je šansa za pomeranje ka lošijem rešenju manja, tj. vrednost $f(x')$ treba da bude bliža vrednosti $f(x)$, gde funkcija f predstavlja zadatu funkciju cilja. Adekvatna vrednost parametra α se može dobiti testiranjem, ali i implementacijom nekog od modela učenja u algoritam.

Algoritam 18: Zamena okolina kod SVNS metode

Ulaz: x, x', k, α
Izlaz: x, k
if $f(x') - \alpha\rho(x, x') < f(x)$ **then**
  $x \leftarrow x'$;
  $k \leftarrow 1$;
else
  $k \leftarrow k + 1$;
end

2.9 Metoda promenljivih okolina sa dekompozicijom

Metoda promenljivih okolina sa dekompozicijom (engl. Variable Neighborhood Decomposition Search - VNDS) predstavlja proširenje osnovne metode promenljivih okolina. VNDS je metoda na dva nivoa, zasnovana na dekompoziciji problema koji se rešava [68]. Primenjuje se kod veoma teških problema gde osnovna VNS metoda ne može efikasno da pretraži prostor pretrage i konvergira ka globalnom optimumu. VNDS je predstavljen algoritmom 19. Za potrebe algoritma uveden je još jedan uslov zaustavljanja, koji se koristi za zaustavljanje dekomponovanih problema. Tačnije, parametar t_d predstavlja dozvoljeno vreme izvršavanja osnovne VNS metode pri rešavanju dekomponovanih problema.

Algoritam 19: Metoda promenljivih okolina sa dekompozicijom

Ulaz: x, k_{max}, t_d
Izlaz: x
while *uslov zaustavljanja nije ispunjen* **do**
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $x' \leftarrow razmrdavanje(x, k)$;
    $y \leftarrow x' \backslash x$;
    $y' \leftarrow osnovnaMetodaPromenljivihOkolina(y, k, t_d)$;
    $x'' \leftarrow (x' \backslash y) \cup y'$;
    $x''' \leftarrow prvoPoboljšanje(x'')$;
    $(x, k) \leftarrow zamenaOkolina(x, x''', k)$;
  end
end

Neka je rešenje x predstavljeno određenim skupom atributa i neka je y skup od k atributa rešenja x' koji se ne nalaze u rešenju x ($y = x' \backslash x$). U VNDS metodi, potrebno je pronaći lokalni optimum y' u odnosu na prostor atributa od kojih se sastoji y. Neka je sa x'' označeno odgovarajuće rešenje u celom prostoru atributa, odnosno rešenje koje se dobija zamenom atributa y sa y' u rešenju x' ($x'' = (x' \backslash y) \cup y'$). Nakon zamene se pronalazi lokalni optimum x''' u celom prostoru S koristeći x'' kao početno rešenje.

2.10 Pregled primena VNS metode za rešavanje lokacijskih problema

Metoda promenljivih okolina je primenjena za rešavanje mnogih klasa problema matematičke optimizacije: lokacijskih problema, problema klasterovanja, problema planiranja, problema usmeravanja vozila, problema dizajna mreže, problema udruživanja, problema u biologiji, geometriji, telekomunikacijama [5, 6, 22, 63, 80, 144, 67, 133], itd. U nastavku ove sekcije biće detaljnije predstavljene primene VNS metode za rešavanje različitih varijanti problema p-medijane, p-centra i hab lokacijskih problema.

Prva primena VNS metode za rešavanje lokacijskih problema prikazana je u radu [59] publikovanom 1997. godine u kome je razmatran problem p-medijane. Tada je prvi put primenjena metoda brze zamene okolina sa ciljem poboljšanja efikasnosti pretrage. U radu [48] iz 2002. godine autori su predstavili više različitih strategija paralelizacije VNS metode prilikom rešavanja problema p-medijane. Paralelizacija VNS metode kooperativnom višestrukom pretragom za rešavanje p-medijane je predstavljena 2004. godine u radu [26]. Osnovna ideja ovog pristupa je paralelno izvršavanje više nezavisnih VNS metoda, koristeći asinhronu komunikaciju za razmenu podataka o najboljem pronađenom rešenju. U radu [44] iz 2008. godine predstavljena je VNS metoda za rešavanje problema p-medijane sa kapacitetima. Predložena heuristika na osnovu jednostavno izračunatih donjih granica odlučuje da li je potrebno računati odgovarajuća pomeranja u okolini. Eksperimentalni rezultati su pokazali da je ovaj pristup pronašao sva poznata najbolja rešenja ovog problema, dok je za jednu instancu najbolje poznato rešenje poboljšano. Metoda promenljivih okolina je primenjena i za rešavanje problema klasterovanja, tako što je rešavan problem p-medijane, gde uspostavljeni centri i njima pridruženi klijenti predstavljaju dobijene klastere. Predloženi algoritam je rešavao veoma velike instance problema efikasno, bez potrebe za smanjivanjem količine podataka ili izdvajanjem uzoraka. Eksperimentalni rezultati su pokazali da je VNS nadmašio poznate heuristike iz literature, CLARA i CLARANS.

U radu [104] publikovanom 2003. godine je predstavljene su VNS metoda i tabu pretraga za rešavanje problema p-centra. Korišćena je struktura okolina zasnovana na zameni uspostavljenog i neuspostavljenog centra. Pokazano je da je ova struktura okolina efikasnija od one korišćene za rešavanje problema p-medijane. Prikazani su detaljni eksperimentalni rezultati i upoređene su performanse VNS metode, tabu

pretrage i drugih heuristika za rešavanje problema p-centra. U radu [30] iz 2014. godine prikazana je VNS metoda za rešavanje problema $(p|q)$-centra. Hibrid VNS metode i egzaktne metode za rešavanje problema p-centra i $(p|q)$-centra velikih dimenzija je predstavljen u radu [74] iz 2016. godine. Hibridna metoda je testirana na instancama sa do 71 hiljadom lokacija i vrednošću p koja varira između 5 i 100. Ovaj metod je pokazao zadovoljavajuće rezultate prilikom rešavanja instanci obe klase problema.

Metoda GVNS za rešavanje p-hab lokacijskog problema sa jednostrukom alokacijom bez ograničenja kapaciteta je predstavljena 2010. godine u radu [73]. U metodi su korišćene tri različite strukture okoline, kao i strukture podataka koje obezbeđuju efikasno računanje funkcije cilja. Predložena je ugnježdena strategija u okviru metode promenljivog spusta. Eksperimentalni rezultati su pokazali da je GVNS nadmašio najbolje poznate heuristike za rešavanje ovog problema. Osim toga neki od najboljih poznatih rezultata su poboljšani. U radu [89] iz 2013. godine predstavljena je osnovna VNS metoda za rešavanje hab lokacijskog problema sa jednostrukom alokacijom i sa ograničenjem kapaciteta. Ovaj problem se sastoji iz dva podproblema: odabir habova u mreži i pronalazak optimalne alokacije čvorova ka uspostavljenim habovima. Metoda VNS je korišćena za rešavanje prvog podproblema, dok je za rešavanje drugog podproblema korišćen CPLEX rešavač. Predstavljeni algoritam je na svim testiranim instancama dostigao poznata optimalna rešenja za kratko vreme izvršavanja.

Metodom VNS su rešavani i drugi lokacijski problemi: lokacijski problem bez ograničenja kapaciteta [60], problem maksimalnog pokrivanja korisnika sa fazi poluprečnikom pokrivenosti [29], p-hub lokacijski problem sa r alokacija bez ograničenja kapaciteta [134], itd. U literaturi je predloženo više hibridnih heuristika za rešavanje lokacijskih problema koje uključuju VNS metodu [56, 136, 112, 71]. Osim toga VNS metoda je uspešno primenjena i za rešavanje robusnih lokacijskih problema [98].

Glava 3

Problem određivanja lokacija autobuskih terminala

Problem određivanja lokacija autobuskih terminala (engl. Bus Terminal Location Problem - BTLP) je lokacijski problem, prvi put predstavljen u radu [49] 2011. godine. Kako je formulisan kao kombinacija dva NP-teška problema, problema p-medijane i problema maksimalnog pokrivanja korisnika, pretpostavlja se da je i BTLP NP-težak. Klijenti su predstavljeni skupom stanica (stajališta) javnog prevoza, kao što su autobuske ili metro stanice. Poznat je broj korisnika svake stanice, na primer broj ljudi koji u toku jednog dana koristi usluge te stanice, kao i udaljenost između svake stanice i potencijalnog terminala. Potencijalni terminal (centar), može biti postojeća stanica ili lokacija za izgradnju novog terminala. Poznat je dostupan prostor (poluprečnik) oko svakog potencijalnog centra. Uspostavljen centar može da opslužuje samo klijente u svom dostupnom prostoru. Sa ciljem što boljeg iskorišćenja usluga javnog servisa, putnici svake stanice koriste usluge najbližeg uspostavljenog terminala, pod uslovom da se nalaze u njegovom dostupnom prostoru, odnosno da je razdaljina od klijenta do centra manja od zadatog poluprečnika. Potrebno je odabrati unapred definisani broj centara iz datog skupa potencijalnih centara, tako da se maksimizuje ukupan kvalitet usluge javnog servisa. Rezultati iz ove glave su prikazani u radu [34].

3.1 Kombinatorna formulacija problema

Neka je $J = \{j_1, j_2, ..., j_n\}$ skup klijenata (autobuske ili metro stanice jednog grada) i neka je $I = \{i_1, i_2, ..., i_m\}$ skup potencijalnih centara, gde I ne mora biti

podskup skupa J. Neka je sa $C = [c_{ij}]$ označena matrica rastojanja između klijenta $i \in I$ i potencijalnog centra $j \in J$. Poluprečnik dostupnog prostora oko svakog potencijalnog centra je konstantan i označen sa r, dok p predstavlja tačan broj centara koje treba uspostaviti. Neka je sa $D = \{d_1, d_2..., d_n\}$ označen broj korisnika usluga svakog klijenta, na primer broj ljudi koji u toku jednog dana koriste usluge odgovarajuće autobuske ili metro stanice. Preciznije, d_k je maksimalan broj korisnika usluga klijenta $j_k \in J$. Sa J_i^* je označen podskup skupa J, koji sadrži sve lokacije koje mogu biti uslužene od strane centra $i \in I$. Preciznije, $J_i^* = \{j \in J : c_{ij} \leq r\}$ i $J^* = \bigcup_{i \in S} J_i^*$, gde $S \subset I$ predstavlja uspostavljene centre.

Potrebno je odabrati podskup S skupa I, $|S| = p$, tako da ukupan kvalitet usluge javnog servisa bude maksimizovan. Ukupan kvalitet usluge se definiše kao:

$$F(S) = \sum_{j \in J^*} d_j \times f(\min_{i \in S}\{c_{ij}\}),$$

gde f predstavlja opadajuću funkciju. Može se primetiti da sa povećanjem rastojanja između klijenta i centra vrednost funkcije cilja opada.

U tabeli 3.1 prikazano je 20 lokacija klijenata određenih x i y koordinatama, kao i broj korisnika usluga svakog klijenta. U tabeli 3.2 prikazano je 8 potencijalnih lokacija za uspostavljanje centara određenih x i y koordinatama. Neka je distanca c_{ij} između klijenata i i potencijalnog centra j definisana euklidskim rastojanjem između dve odgovarajuće tačke. Neka je poluprečnik dostupnog prostora oko svakog potencijalnog centra ima vrednost $r = 1$, a opadajuća funkcija jednaka $f(x) = e^{-x}$. Neka je potrebno uspostaviti 5 centara, tj. $p = 5$. Tada je optimalno rešenje BTLP problema uspostaviti centre i_1, i_3, i_4, i_7 i i_8 i funkcija cilja ima vrednost 985.09. Grafički prikaz ovog rešenja je dat na slici 3.1. Crvenom bojom su prikazani uspostavljeni centri, dok su plavom bojom prikazani neuspostavljeni centri. Može se zapaziti da svi klijenti osim klijenta j_5 koriste usluge nekog od uspostavljenih terminala, kao i da su centri odabrani tako da im dodeljeni klijenti budu relativno blizu, jer je po definiciji problema funkcija cilja bolja ukoliko su klijenti bliži dodeljenim centrima.

3.2 Pregled relevantne literature za BTLP

Za BTLP je pokazano značajno interesovanje u različitim istraživanjima. BTLP je u [49] predstavljen kao p-lokacijski problem snabdevača sa neograničenim kapacitetima (p-UFLP) i ograničenim rastojanjima. Kako je p-UFLP NP-težak, u [49] su

Tabela 3.1: Lista klijenata

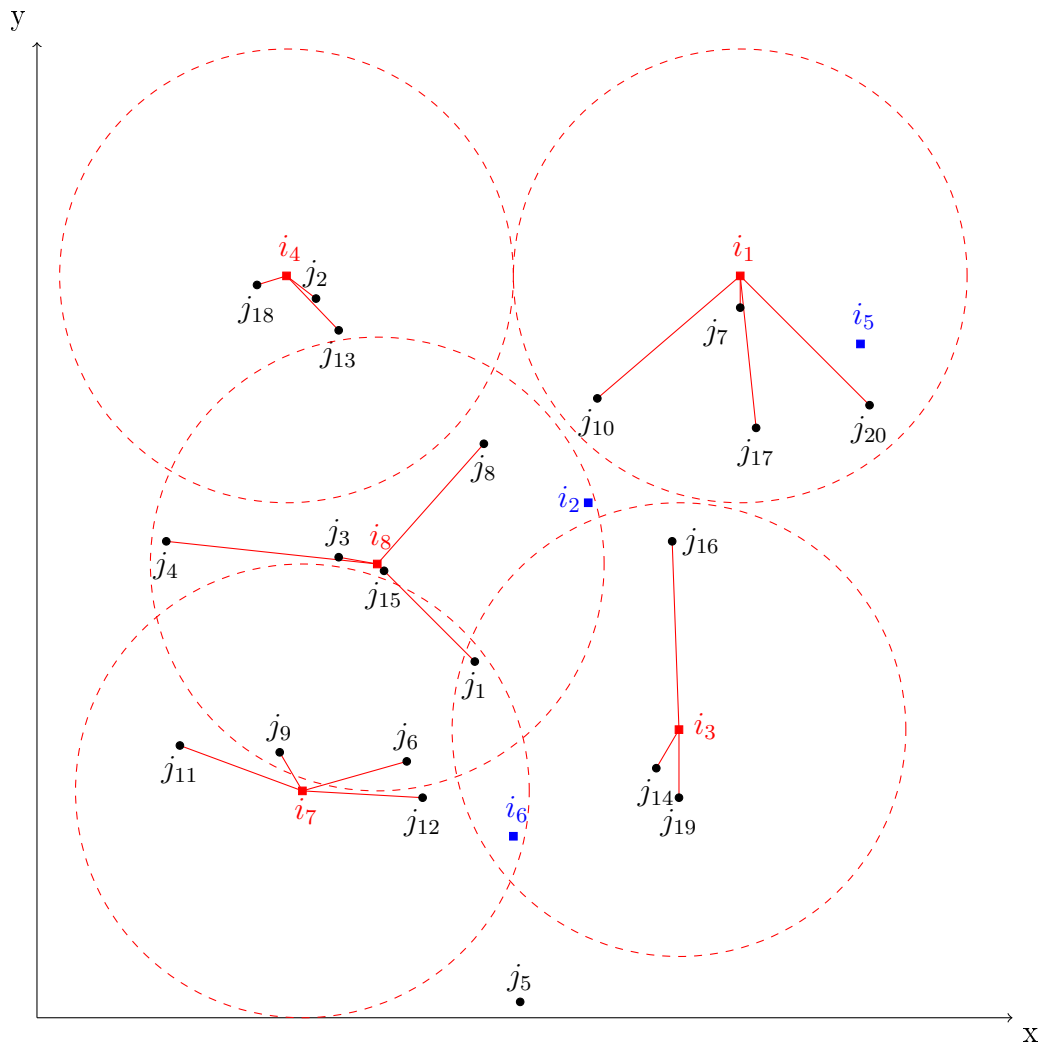
klijent	x koordinata	y koordinata	broj korisnika usluga klijenta
j_1	1.43	1.57	73
j_2	0.73	3.17	96
j_3	0.83	2.03	62
j_4	0.07	2.10	64
j_5	1.63	0.07	90
j_6	1.13	1.13	73
j_7	2.60	3.13	53
j_8	1.47	2.53	81
j_9	0.57	1.17	100
j_{10}	1.97	2.73	70
j_{11}	0.13	1.20	93
j_{12}	1.20	0.97	97
j_{13}	0.83	3.03	78
j_{14}	2.23	1.10	67
j_{15}	1.03	1.97	78
j_{16}	2.30	2.10	57
j_{17}	2.67	2.60	95
j_{18}	0.47	3.23	100
j_{19}	2.33	0.97	73
j_{20}	3.17	2.70	74

Tabela 3.2: Lista potencijalnih centara

centar	x koordinata	y koordinata
i_1	2.60	3.27
i_2	1.93	2.27
i_3	2.33	1.27
i_4	0.60	3.27
i_5	3.13	2.97
i_6	1.60	0.80
i_7	0.67	1.00
i_8	1.00	2.00

predloženi novi evolutivni algoritam i memetski algoritam za rešavanje BTLP. Svakom centru je pridružena vrednost nazvana potencijal funkcije cilja (engl. Potential of Object Function - POF) i na osnovu te vrednosti dizajniran je novi operator mutacije. Kako bi ubrzali memetski algoritam, autori su računali procenu varijacije funkcije cilja zasnovanu na vrednosti POF u okviru lokalne pretrage. Dodatno, autori su predložili metodu simuliranog kaljenja sa više početaka za rešavanje BTLP.

U pregledu transportnih problema u radu [31], BTLP je svrstan u grupu pro-



Slika 3.1: Grafički prikaz rešenja BTLP problema

blema kopnenog transporta. Osim toga, BTLP je opisan u radu [143] u kome su autori razmatrali hab lokacijski problem u dizajnu urbanih javnih tranzitnih mreža, koji uključuje dve faze: uspostavljanje habova i optimizaciju tranzita između uspostavljenih habova. Predloženi dvofazni optimizacioni pristup testiran je na realnim podacima urbane oblasti grada Dalian u Kini. Takođe, u radu [8] predstavljena su dva modela za problem uspostavljanja autobuskih terminala sa fazi-parametrima (FBTLP). Razlika u odnosu na klasičan BTLP se ogleda u činjenici da je maksimalni broj korisnika jedne stanice predstavljen fazi brojem. U drugoj formulaciji, razmotrena je dodatna pretpostavka fazi dostupnog prostora oko potencijalnog centra. Autori su predstavili efikasan genetski algoritam za rešavanje oba FBTLP

problema.

BTLP poseduje karakteristike nekih od najviše istraživanih lokacijskih problema. Ako je u kombinatornoj formulaciji BTLP problema opisanoj u sekciji 3.1 funkcija f definisana sa $f(x) = -x$, broj korisnika svih stanica jednak 1 (tj. $d_j = 1, j = 1, 2, \ldots, n$) i poluprečnik r dovoljno veliki broj (npr. $r = \max_{i=1,\ldots,m, j=1,\ldots,n}(c_{ij})$), formulacija BTLP postaje ekvivalentna formulaciji problema p-medijane. Sa druge strane, ako je funkcija f konstantna $f(x) = 1$, formulacija BTLP je ekvivalentna formulaciji problema MCLP.

U [122] razmatran je problem p-medijane sa dodatnim ograničenjem pokrivenosti (engl. P-median Problem With an Additional Coverage Constraint - CONPMP). BTLP je generalizacija CONPMP problema, budući da se CONPMP formulacija može dobiti iz BTLP formulacije postavljanjem $f(x) = -x$.

3.3 Metoda promenljivih okolina za BTLP

U daljem tekstu opisana je paralelna metoda promenljivih okolina (PVNS) za rešavanje BTLP problema. U predstavljenoj PVNS implementaciji, rešenje je definisano nizom uspostavljenih centara. Na osnovu niza uspostavljenih centara, lako se može odrediti koji klijent je povezan sa kojim centrom. Za definisanje strukture okoline korišćena je metrika rastojanja između dva rešenja zasnovana na broju zamena uspostavljenih i neuspostavljenih centara. Jedna zamena predstavlja uspostavljanje jednog trenutno neuspostavljenog centra i zatvaranje jednog uspostavljenog centra. Rešenje res_1 se nalazi u k-toj okolini rešenja res_2, ako je moguće konstruisati res_2 iz res_1 primenom tačno k' zamena, pri čemu je $k' \leq k$.

Struktura predstavljenog algoritma

Primenjena je kombinovana metoda redukovane i osnovne metode promenljivih okolina. Rešenje redukovane metode promenljivih okolina iskorišćeno je kao početno rešenje osnovne metode promenljivih okolina. Početno rešenje redukovane metode slučajno je generisano.

Unapređena lokalna pretraga

U pogledu procesorskog vremena, faza lokalne pretrage je najzahtevniji deo VNS algoritma. Mnoga unapređenja lokalne pretrage zasnovane na zameni (engl. Swap

Based Local Search) korišćena su za rešavanje različitih lokacijskih problema [142, 59, 118]. Metoda brze zamene okolina (engl. Swap Based Fast Interchange) prvi put je predstavljena 1983. godine u [142], dok je prvi put primenjena za rešavanje problema p-medijane u radu [59] iz 1997. godine. Osnovna ideja u [59] je efikasan pronalazak najboljeg centra za zatvaranje u slučaju da je poznat centar koji se uspostavlja.

U nastavku će sa t_{in} biti obeležen centar koji se uspostavlja, i sa t_{out} centar koji je najbolje zatvoriti u slučaju uspostavljanja centra t_{in}. Pritom t_{fc} označava prvi najbliži uspostavljeni centar klijenta l, dok t_{sc} označava drugi najbliži uspostavljeni centar klijenta l.

Neka promenljiva *dobit* označava poboljšanje vrednosti funkcije cilja. Niz *gubitak* predstavlja pogoršanje vrednosti funkcije cilja, takvo da je *gubitak*[i] pogoršanje u slučaju zatvaranja centra i.

Neka je nova funkcija f_r, zasnovana na opadajućoj funkciji f i dostupnom prostoru oko potencijalnog centra, definisana kao:

$$f_r(x) = \begin{cases} f(x), & \text{ukoliko je } x \leq r, \\ 0, & \text{inače.} \end{cases}$$

U nastavku je prikazana analiza uticaja otvaranja centra t_{in} i zatvaranja nekog od uspostavljenih centara na klijenta l. Tri situacije se razlikuju:

- $c_{t_{in},l} < c_{t_{fc},l} \leq c_{t_{sc},l}$: kada je centar t_{in} bliži klijentu l od najbližeg uspostavljenog centra t_{fc}, l prelazi sa t_{fc} na t_{in} i *dobit* dobija vrednost:

$$dobit = dobit + (f_r(c_{t_{in},l}) - f_r(c_{t_{fc},l}))d_l,$$

- $c_{t_{fc},l} \leq c_{t_{in},l} < c_{t_{sc},l}$: kada je prvi najbliži uspostavljeni centar t_{fc} bliži klijentu l od centra t_{in}, i t_{in} je bliži klijentu l od drugog najbližeg uspostavljenog terminala t_{sc}, u slučaju zatvaranja centra t_{fc} l prelazi sa t_{fc} na t_{in} i *gubitak*[t_{fc}] dobija vrednost:

$$gubitak[t_{fc}] = gubitak[t_{fc}] + (f_r(c_{t_{in},l}) - f_r(c_{t_{fc},l}))d_l,$$

- $c_{t_{fc},l} \leq c_{t_{sc},l} \leq c_{t_{in},l}$: kada je drugi najbliži uspostavljeni centar t_{sc} klijentu l bliži klijentu l od centra t_{in}, l prelazi sa t_{fc} na t_{sc} u slučaju zatvaranja centra t_{fc} i *gubitak*[t_{fc}] dobija vrednost:

$$gubitak[t_{fc}] = gubitak[t_{fc}] + (f_r(c_{t_{sc},l}) - f_r(c_{t_{fc},l}))d_l.$$

Računanje prvog i drugog najbližeg uspostavljenog centra svakom klijentu neophodno je pre svake iteracije lokalne pretrage.

Smanjena veličina okoline

Uspostavljanje jednog centra t_{in} u problemu BTLP ima uticaj samo na one klijente koje se nalaze u $J^*_{t_{in}}$, dok kod problema p-medijane ima potencijalni uticaj na sve lokacije iz J. Zbog toga je dovoljno preračunati kako lokacije iz $J^*_{t_{in}}$ utiču na funkciju cilja u slučaju uspostavljanja t_{in}. Za svaku lokaciju $l \notin J^*_{t_{in}}$, l ostaje povezano za svoj prvi najbliži centar t_{fc}, ako t_{fc} ostaje uspostavljen, dok se inače l prebacuje na svoj drugi najbliži uspostavljeni centar t_{sc}. Dodatno, kada se računaju nizovi prvi i drugi najbliži, niz *gubitak* se inicijalizuje sa pretpostavkom da se u slučaju zatvaranja prvog najbližeg centra lokacije l, l pomera na drugi najbliži uspostavljeni centar. Ako se pomeranje na drugi najbliži centar ne desi, izvršava se korekcija: dodata vrednost se oduzima od odgovarajućeg elementa niza *gubitak*.

Vremenska složenost predstavljene smanjene veličine okoline je funkcija zavisna od poluprečnika r. Preciznije, koristeći smanjenu veličinu okoline, vremenska složenost pronalaska najbolje zamene je $O(\sum_{t_{in} \in I \setminus S} |J^*_{t_{in}}|)$, dok je kompleksnost bez smanjene veličine okoline jednaka $O(\sum_{t_{in} \in I \setminus S} |J|)$. Ukoliko je poluprečnik r dovoljno veliki smanjena i originalna veličina okoline su iste veličine. Međutim, to nije slučaj sa realnim instancama kod kojih postoji značajno poboljšanje performansi algoritma.

Paralelna verzija VNS algoritma za BTLP

U ovom radu je korišćena paralelizacija niskog nivoa VNS algoritma. Dobijena su ista rešenja nezavisno od broja procesora na kojima se izvršava testiranje. Za merenje performansi je korišćeno ubrzanje u odnosu na osnovu sekvencijalnu verziju algoritma, dok je za uslov zaustavljanja paralelnog algoritma korišćena dostignuta količina izvršavanja, što znači da je izvršen isti broj iteracija algoritma u sekvencijalnoj i paralelnoj pretrazi.

Sa ciljem očuvanja fundamentalne jednostavnosti VNS koncepta, nije paralelizovan svaki deo korišćenog algoritma, već su paralelizovani samo određeni vremenski najzahtevniji delovi algoritma. Za identifikaciju računarski najzahtevnijih delova algoritama korišćen je alat za merenje performansi procesora koji je ugrađen u razvojno okruženje Microsoft Visual Studio Ultimate 2013.

Posmatrani su statistički uzorci izvršavanja algoritma na određenim instancama problema. Prikupljanje uzoraka predstavlja statistički metod merenja performansi algoritma koji pokazuje koje su funkcije vremenski najzahtevnije u programu [55]. Metod prikupljanja uzoraka sakuplja informacije o funkcijama koje se izvršavaju u specifičnim vremenskim intervalima. Kada se merenje performansi završi kao rezultat se dobijaju inkluzivni i ekskluzivni uzorci svih implementiranih funkcija u aplikaciji. Inkluzivni uzorci pokazuju koliko je vremena program izvršavao naredbe određene funkcije uključujući i sve funkcije koje su pozvane iz te funkcije. Ekskluzivni uzorci pokazuju koliko je vremena program izvršavao naredbe samo jedne određene funkcije, isključujući vreme izvršavanja drugih funkcija koje su pozvane iz te funkcije.

Inkluzivni uzorci dobijeni iz alata za merenje performansi procesora pokazali su da je u neparalelizovanoj verziji algoritma, više od 90% vremena izvršavanja aplikacije potrošeno na dve funkcije koje pripadaju fazi lokalne pretrage:

- funkcija koja računa prvi i drugi najbliži uspostavljeni centar za svakog klijenta $j \in J$ i
- funkcija koja računa koji je centar najbolje zatvoriti u slučaju da se uspostavlja centar t_{in}.

Dobijeni rezultati predstavljali su motivaciju za paralelizaciju samo ove dve funkcije. Paralelna verzija lokalne pretrage se sastoji od ponavljanja naredna dva koraka u svakoj iteraciji sve dok ima poboljšanja. Prvo, u svakoj iteraciji, nizovi koji predstavljaju prvog i drugog najbližeg uspostavljenog centra svakom klijentu ($prviNajbliži$ i $drugiNajbliži$) se računaju i niz $gubitak$ se inicijalizuje. Nakon toga, u paralelnoj petlji, za svaki neuspostavljeni centar t_{in} koji je moguće uspostaviti se računa koji centar t_{out} je najbolje zatvoriti, koristeći već izračunate nizove $prviNajbliži$, $drugiNajbliži$ i $gubitak$. Takođe računa se i razlika vrednosti funkcije cilja Δ. Na kraju, ukoliko najbolja pronađena razlika funkcije cilja Δ_{max} predstavlja poboljšanje ($\Delta_{max} > 0$, tj. vrednost funkcije cilja se povećala), odgovarajući centar $best_{t_{in}}$ se uspostavlja i $best_{t_{out}}$ se zatvara. Opisana paralelna implementacija lokalne pretrage predstavljena je algoritmom 20.

Funkcija za računanje nizova $prviNajbliži$ i $drugiNajbliži$, u kojoj se inicijalizuje i niz $gubitak$ takođe je paralelizovana. Prvo se nizovi $prviNajbliži$ i $drugiNajbliži$ inicijalizuju tako da se sve vrednosti u njima postave na vrednost $null$. Dalje se u paralelnom ciklusu nizovi $prviNajbliži$ i $drugiNajbliži$, za svaki uspostavljeni cen-

Algoritam 20: Paralelna lokalna pretraga za rešavanje BTLP problema

```
Ulaz: rešenje
Izlaz: rešenje
popravka ← true;
while popravka do
    popravka ← false;
    (prviNajbliži, drugiNajbliži, gubitak) ←
        izračunajNajbližeCentreInicijalizujGubitak(rešenje);
    Δ_best ← 0;
    parallel foreach neuspostavljeni centar t_in do
        (t_out, Δ) ←
            najpogodnijiCentarZaZatvaranje
                (t_in, prviNajbliži, drugiNajbliži, gubitak);
        begin lock Δ_best:
            if Δ > Δ_best then
                Δ_best ← Δ;
                best_tout ← t_out;
                best_tin ← t_in;
            end
        end
    end
    if Δ_best > 0 then
        popravka ← true;
        rešenje ← zamena(rešenje, best_tin, best_tout);
    end
end
```

tar t i za svakog klijenta l koji se nalazi u dostupnom prostoru oko centra t, ažuriraju ukoliko je klijent l bliži centru t u odnosu na $prviNajbliži[l]$ i $drugiNajbliži[l]$. Niz gubitak se najpre inicijalizuje tako što svi elementi niza dobiju vrednost 0, a zatim se modifikuje u paralelnom ciklusu sa pretpostavkom da u slučaju zatvaranja prvog najbližeg centra od klijenta l, klijent l prelazi na svoj drugi najbliži centar. Opisana procedura predstavljena je algoritmom 21.

Računanje najboljeg centra koji je potrebno zatvoriti u slučaju da se centar t_{in} uspostavlja realizovano je funkcijom *najpogodnijiCentarZaZatvaranje* opisanom u sekciji *Unapređena lokalna pretraga* na strani 46.

Funkcija *najpogodnijiCentarZaZatvaranje* predstavljena je algoritmom 22.

Algoritam 21: Procedura koja pronalazi najbliže centre i inicijalizuje gubitak

Ulaz: *rešenje*
Izlaz: $prviNajbliži, drugiNajbliži, gubitak$
parallel for $i = 1{:}n$ **do**
  $prviNajbliži[i] \leftarrow null$;
  $drugiNajbliži[i] \leftarrow null$;
end
parallel foreach *uspostavljeni centar* t **do**
  foreach *klijent* l *u dostupnom prostoru oko centra* t **do**
    begin lock $prviNajbliži[l], drugiNajbliži[l]$:
      $t_{fc} \leftarrow prviNajbliži[l]$;
      $t_{sc} \leftarrow drugiNajbliži[l]$;
      if $c_{t,l} < c_{t_{fc},l}$ **then**
        $drugiNajbliži[l] \leftarrow prviNajbliži[l]$;
        $prviNajbliži[l] \leftarrow t$;
      else if $c_{t,l} < c_{t_{sc},l}$ **then**
        $drugiNajbliži[l] \leftarrow t$;
      end
    end
  end
end
parallel for $i = 1{:}p$ **do**
  $gubitak \leftarrow 0$;
end
parallel foreach *klijent* l **do**
  $t_{fc} \leftarrow prviNajbliži[l]$;
  $t_{sc} \leftarrow drugiNajbliži[l]$;
  begin lock $gubitak[t_{fc}]$
    $gubitak[t_{fc}] \leftarrow gubitak[t_{fc}] + (f_r(c_{t_{sc},l}) - f_r(c_{t_{fc},l}))d_l$;
  end
end

3.4 Eksperimentalni rezultati

U ovoj sekciji, prikazani su eksperimentalni rezultati predložene PVNS metode i upoređeni su sa rezultatima više različitih metoda prikazanih u [49]. Predložena PVNS metoda implementirana je u jeziku *C♯*, na .NET platformi. Testovi su izvršeni na računaru koji ima Intel Core i7-860 2.8 GHz procesor i 8GB RAM memorije. Za implementaciju paralelizovanih delova algoritama korišćen je koncept arhitekture deljene memorije sa više niti koje se izvršavaju na različitim procesorskim jezgrima.

Algoritam 22: Procedura koja pronalazi najpogodniji centar za zatvaranje

Ulaz: $t_{in}, prviNajbliži, drugiNajbliži, gubitak$
Izlaz: t_{out}, Δ
$dobit \leftarrow 0$;
foreach *klijent* l *u dostupnom prostoru oko centra* t_{in} **do**
  $t_{fc} \leftarrow prviNajbliži[l]$;
  $t_{sc} \leftarrow drugiNajbliži[l]$;
  if $c_{t_{in},l} < c_{t_{fc},l}$ **then**
    $dobit = dobit + (f_r(c_{t_{in},l}) - f_r(c_{t_{fc},l}))d_l$;
    korekcija:
    $gubitak[t_{fc}] \leftarrow gubitak[t_{fc}] - (f_r(c_{t_{sc},l}) - f_r(c_{t_{fc},l}))d_l$;
  else if $c_{t_{in},l} < c_{t_{sc},l}$ **then**
    $gubitak[t_{fc}] \leftarrow gubitak[t_{fc}] + (f_r(c_{t_{in},l}) - f_r(c_{t_{fc},l}))d_l$;
    korekcija:
    $gubitak[t_{fc}] \leftarrow gubitak[t_{fc}] - (f_r(c_{t_{sc},l}) - f_r(c_{t_{fc},l}))d_l$;
  end
end
$t_{out} \leftarrow$ odgovarajući centar čija je vrednost u nizu *gubitak* najveća, odnosno ima vrednost $gubitak_{max}$;
$\Delta \leftarrow gubitak_{max} + dobit$;

Korišćena je .NET Task Parallel biblioteka [46]. Ova biblioteka omogućila je korišćenje paralelnih ciklusa, koji kreiraju zasebne niti za izvršavanje svake pojedinačne iteracije ciklusa.

Da bi se procenila efikasnost predloženog pristupa, izvršen je veliki broj eksperimenata. Korišćena je opadajuća funkcija $f(x) = e^{-x}$, kao i u [49]. Eksperimenti su izvedeni na sledeća dva skupa podataka.

1. Petnaest test instanci problema specijalno kreiranih za BTLP i opisanih u [49], sa sledećim parametrima:
   - $p \in \{64, 125, 187\}, r = 20, n = m = 250$,
   - $p \in \{125, 250, 375\}, r = 20, n = m = 500$,
   - $p \in \{187, 375, 562\}, r = 20, n = m = 750$,
   - $p \in \{250, 500, 750\}, r = 20, n = m = 1000$.
2. Osamnaest instanci[1], zasnovanih na *rl* instancama TSPLIB biblioteke[2], koje

[1]Kreirane BTLP instance su dostupne na adresi `http://www.matf.bg.ac.rs/~djenic/BTLP`
[2]TSPLIB instance su dostupne na adresi `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

su originalno kreirane za rešavanje problema trgovačkog putnika [117]. Ove instance su prilagođene BTLP problemu. Kako originalne *rl* instance sadrže samo jedan skup tačaka, odabrano je slučajno da pola tog skupa predstavlja skup I, a druga polovina skup J. Za broj korisnika svake stanice odabrani su slučajni brojevi iz tri različita skupa:

- $[100, 1000]$,
- $[700, 1000]$ i
- $[100, 400] \cup [700, 1000]$,

što je naznačeno u imenu kreirane instance. Tako, na primer, naziv instance *rl1304_100_400_700_1000* označava da je ona dobijena iz instance *rl1304* koristeći slučajne vrednosti iz $[100, 400] \cup [700, 1000]$. Uzimajući u obzir udaljenost između tačaka u *rl* instancama, odgovarajući poluprečnik dostupnog prostora oko potencijalnih centara r je 2000 ili 3000.

Kako funkcija $f(x) = e^{-x}$, veoma brzo opada kako x raste, rezultati na predloženim instancama se ne mogu primeniti u slučajevima velikih rastojanja. Na primer, u slučaju da je distanca između uspostavljenog centra i i klijenta j jednaka $c_{ij} = 1000$, funkcija cilja se povećava za $d_j e^{-1000}$, gde je drugi činilac proizvoda veoma blizak nuli. Zbog toga su, u toku inicijalizacije instance problema, sve koordinate podeljene brojem r i vrednost r je postavljena na $r = 1$. Na taj način, dostupan prostor oko svakog centra ostaje isti u smislu klijenata koji mogu da mu se pridruže, a primena funkcije $f(x) = e^{-x}$ na distance u dostupnom prostoru oko centra daje broj iz intervala $[1/e, 1]$, što je dosta realnije i pogodnije za računanje.

Kao uslov zaustavljanja je korišćeno 1000 uzastopnih izvršenih iteracija algoritma bez postignutog poboljšanja.

Podešavanje parametara

Kako bi se odredili parametri VNS algoritma k_{max} i k_{rvns_max} izvršeni su testovi na 6 različitih instanci koristeći 20 različitih semena za generisanje pseudo-slučajnih brojeva i sledeće parametre:

- rl1304_100_400_700_1000, p = 163, r = 2000,
- rl1304_100_400_700_1000, p = 217, r = 2000,

- rl1323_100_400_700_1000, p = 165, r = 2000,
- rl1323_100_400_700_1000, p = 220, r = 2000,
- rl1889_100_400_700_1000, p = 236, r = 2000,
- rl1889_100_400_700_1000, p = 315, r = 2000.

Za svaku od postavki parametara i svaku instancu posmatrano je najbolje dobijeno rešenje na svim pokretanjima algoritma sa različitim semenima. U tabelama 3.3 i 3.4 prikazani su rezultati analize parametara redom za parametre k_{max} i k_{rvns_max}. Potrebno je obratiti pažnju da notacija $k_{rvns\_max} = 0$ predstavlja slučaj kada je RVNS faza izostavljena iz algoritma. Za sve postavke parametara dobijeni su isti najbolji rezultati, tako da oni nisu prikazani u tabelama. U prvoj koloni tabela prikazane su različite vrednosti odgovarajućeg parametra koji se podešava. U drugoj koloni data je suma vremena t_{res} u sekundama potrebna algoritmu da sa datom postavkom parametara pronađe najbolje rešenje za sve instance. U trećoj koloni data je suma ukupnog vremena izvršavanja algoritma t_{uk} u sekundama za sve instance sa datom postavkom parametara. Ocena kvaliteta dobijenog rešenja, na svim pokretanjima algoritma izvedena je računanjem prosečnog odstupanja u odnosu na najbolje rešenje *agap* i standardnom devijacijom prosečnog odstupanja σ. Poslednje dve kolone sadrže prosečne vrednosti prosečnog odstupanja i standardne devijacije u procentima za sve testirane instance sa datom postavkom parametara.

Više testova izvedeno je koristeći vrednost parametra $k_{rvns\_max} = 0$ kako bi se pronašla što bolja vrednost parametra k_{max}. Rezultati prikazani u tabeli 3.3 pokazuju da je vreme do pronalaska rešenja najveće za $k_{max} = 5$, dok je za ostale vrednosti parametara ono stabilno oko vrednosti od $5s$. Na osnovu analize vremena do pronalaska rešenja, vrednost k_{max} ne bi trebalo da bude 5. Pošto većim vrednostima parametra k_{max} odgovaraju duža ukupna vremena izvršavanja algoritma, odabrana je vrednost $k_{max} = 10$.

Rezultati prikazani u tabeli 3.4 dobijeni su postavljanjem vrednosti parametra k_{max} na 10. Najbolji rezultat postignut je kada je $k_{rvns\_max} = 0$. Zbog toga je RVNS faza izostavljena iz algoritma i početno rešenje za osnovnu metodu promenljivih okolina slučajno je generisano.

Tabela 3.3: VNS - analiza k_{max} parametra

k_{max}	$\sum t_{res}[s]$	$\sum t_{uk}[s]$	$prosek(agap)[\%]$	$prosek(\sigma)[\%]$
5	12.21	55.83	0.0067	0.0098
10	**4.45**	**68.82**	**0.0014**	**0.0026**
15	5.33	90.08	0.0004	0.0010
20	4.66	111.57	0.0000	0.0000
25	6.69	137.77	0.0000	0.0000
30	4.28	148.16	0.0000	0.0000

Tabela 3.4: RVNS - analiza k_{rvns_max} parametra

k_{rvns_max}	$\sum t_{res}[s]$	$\sum t_{uk}[s]$	$prosek(agap)[\%]$	$prosek(\sigma)[\%]$
0	**4.45**	**68.82**	**0.0014**	**0.0026**
1	21.89	86.99	0.0013	0.0029
2	28.71	93.38	0.0015	0.0026
3	26.00	92.41	0.0009	0.0023
4	29.09	96.13	0.0015	0.0037
5	32.16	100.30	0.0016	0.0028

Poređenje dobijenih rezultata

Kako bi se prezentovala efikasnost predstavljenog paralelnog VNS algoritma u poređenju sa neparalelnom verzijom, izvršeno je testiranje obe varijante algoritma na svim test instancama i rezultati su izloženi u tabeli 3.5. Potrebno je napomenuti da su svi testovi izvršeni na procesoru koji sadrži četiri procesorska jezgra. U prvoj koloni tabele 3.5 data je grupa instanci na kojoj su izvršeni testovi. U drugoj i trećoj koloni prikazani su redom suma vremena potrebna za pronalazak najboljeg rešenja t_{res} u sekundama i suma ukupnog vremena izvršavanja algoritma t_{uk} za neparalelizovanu implementaciju algoritma u sekundama. Dalje, odgovarajuće sume vremena t_{res} i t_{uk} paralelne implementacije algoritma date su u četvrtoj i petoj koloni tabele u sekundama. Šesta i sedma kolona predstavljaju dobijeni faktor ubrzanja vremena potrebnog za pronalazak najboljeg rešenja i ukupnog vremena izvršavanja paralelne implementacije algoritma u odnosnu na neparalelizovanu. Budući da je osnovi cilj ovih testova poređenje vremena izvršavanja algoritma između paralelne i neparalelizovane verzije, a ne pronalazak najboljih rešenja, testovi su izvršeni sa jednim pokretanjem algoritma za svaku test instancu. Kako je algoritam dao iste rezultate za paralelnu i neparalelizovanu verziju, vrednost funkcije cilja nije prikazana u tabeli 3.5. Može se videti da je paralelna verzija algoritma u proseku dala oko 3.4 puta brže rezultate u odnosu na neparalelizovanu. Na malim instancama ubrzanje

nije toliko izraženo. To se može objasniti velikom cenom uspostavljanja paralelnih procesa i sinhronizacije, koja kod malih instanci može biti veća i od vremena izvršavanja samog algoritma. Međutim, za rešavanje instanci velikih dimenzija, faktor ubrzanja je blizak broju procesorskih jezgara, što je i bio cilj paralelizacije.

Tabela 3.5: Sumirani rezultati poređenja paralelne i neparalelizovane verzije VNS algoritma

	neparalelizovana verzija alg.		paralelna verzija alg.		faktor ubrzanja	
grupa test instanci	$t_{res}[s]$	$t_{uk}[s]$	$t_{res}[s]$	$t_{uk}[s]$	t_{res}	t_{uk}
LSBTLP-250-250	0.17	8.62	0.32	13.48	0.53	0.64
LSBTLP-500-500	0.63	40.39	0.73	39.22	0.86	1.03
LSBTLP-750-750	4.95	90.78	3.71	68.75	1.33	1.32
LSBTLP-1000-1000	5.10	166.45	3.48	104.45	1.47	1.59
rl1304	48.56	176.00	32.89	118.49	1.48	1.49
rl1323	38.17	167.00	26.20	114.73	1.46	1.46
rl1889	134.40	402.77	68.17	208.84	1.97	1.93
rl5915	13658.73	19688.07	3866.44	5573.26	3.53	3.53
rl5934	12106.24	18415.20	3391.35	5161.09	3.57	3.57
rl11849	92406.25	118141.86	27221.16	34828.42	3.39	3.39
sve instance	118403.19	157297.15	34614.45	46230.74	3.42	3.40

U [49] predloženi su sledeći algoritmi za rešavanje BTLP problema:

- devet algoritama zasnovanih na evolutivnom algoritmu i genetskoj lokalnoj pretrazi,
- metoda simuliranog kaljenja sa više početaka i
- hibridni algoritam koji kombinuje evolutivni algoritam i genetsku lokalnu pretragu.

U [49] izveden je eksperiment testiranja performansi prvih devet algoritama na više od hiljadu slučajno generisanih instanci problema. Konačni test izvršen je na tri odabrana algoritma: jednoj verziji evolutivnog algoritma, hibridnom algoritmu i metodi simuliranog kaljenja sa više početaka. Algoritmi su testirani na 12 grupa instanci i prikazana je prosečna vrednost funkcije cilja i prosečno vreme izvršavanja za svaku instancu. Za svaku test instancu izvršeno je pet testova sa različitim semenima za generisanje pseudo-slučajnih brojeva. Sve metode iz [49] implementirane su u MATLAB 7.0 okruženju i izvršene na računaru PC 3 GHz sa 1 GB RAM memorije.

U tabelama 3.6 i 3.7 prikazano je poređenje najboljih rezultata prikazanih u [49] i najboljih rezultata dobijenih PVNS algoritmom na test instancama iz [49]. Radi korektnog poređenja PVNS je takođe testiran sa 5 različitih semena za generisanje pseudo-slučajnih brojeva. U svakom redu tabela 3.6 i 3.7 prikazani su sledeći podaci:

- naziv instance,
- najbolje rešenje dobijeno PVNS algoritmom,
- prosečno vreme potrebno do pronalaska najboljeg rešenja u sekundama,
- prosečno ukupno vreme izvršavanja algoritma u sekundama,
- prosečno odstupanje u odnosu na najbolje rešenje u procentima i
- standardna devijacija prosečnog odstupanja u procentima.

U sedmoj i osmoj koloni tabela 3.6 i 3.7 su prikazani najbolji rezultati i prosečno vreme izvršavanja algoritma iz [49] u sekundama. Preciznije, predstavljen je najbolji od tri prikazana rezultata u [49] dobijena korišćenjem tri različita algoritma. Kako u [49] nisu prikazani rezultati metoda na svakoj pojedinačnoj instanci i u tabelama 3.6 i 3.7 su dati samo prosečni rezultati za svaku grupu instanci.

Tabela 3.6: Poređenje rezultata na LSBTLP test instancama za $n = m = 250$ i $n = m = 500$

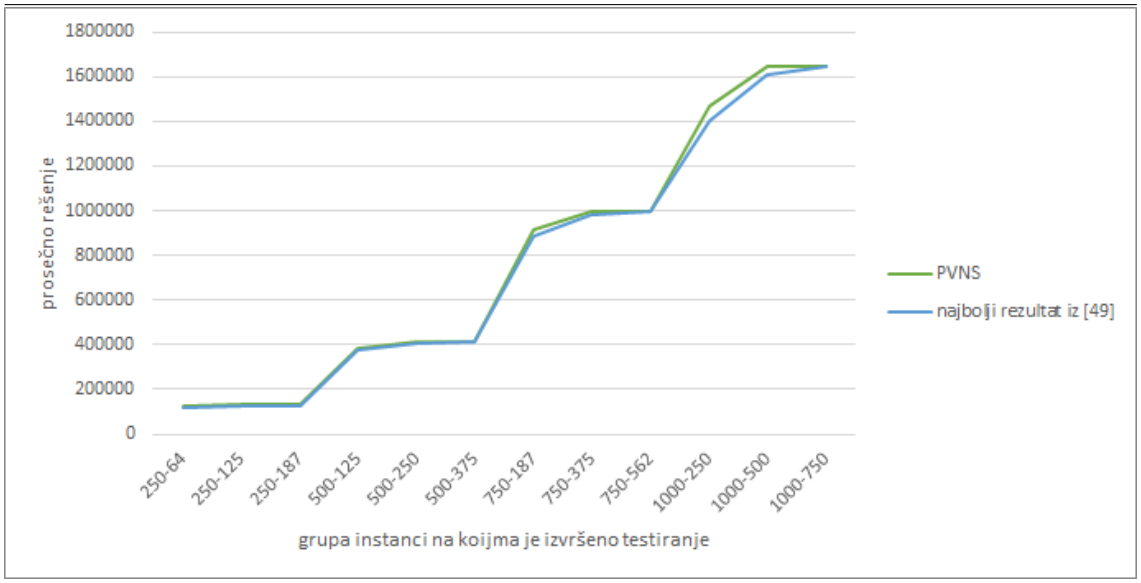
	PVNS					najbolje rešenje iz [49]	
instanca	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t[s]$
LSBTLP-250-22-64	93865.90	0.07	1.90	0.0000	0.0000		
LSBTLP-250-23-64	140302.52	0.02	1.86	0.0000	0.0000		
LSBTLP-250-24-64	147450.59	0.02	1.99	0.0000	0.0000		
prosek	**127206.33**	**0.03**	**1.92**	0.0000	0.0000	119220.40	7.96
LSBTLP-250-22-125	96721.32	0.02	2.09	0.0000	0.0000		
LSBTLP-250-23-125	145438.69	0.02	2.11	0.0000	0.0000		
LSBTLP-250-24-125	150529.17	0.02	2.17	0.0000	0.0000		
prosek	**130896.39**	**0.02**	**2.12**	0.0000	0.0000	124045.80	7.77
LSBTLP-250-22-187	96767.01	0.02	1.99	0.0000	0.0000		
LSBTLP-250-23-187	145454.03	0.02	1.94	0.0000	0.0000		
LSBTLP-250-24-187	150548.10	0.02	1.95	0.0000	0.0000		
prosek	**130923.05**	**0.02**	**1.96**	0.0000	0.0000	124328.90	7.85
LSBTLP-500-51-125	378327.97	0.05	3.79	0.0000	0.0000		
LSBTLP-500-52-125	380905.74	0.05	3.72	0.0000	0.0000		
LSBTLP-500-53-126	360714.67	0.05	3.79	0.0000	0.0000		
LSBTLP-500-54-126	425967.69	0.05	3.79	0.0000	0.0000		
prosek	**386479.02**	**0.05**	**3.77**	0.0000	0.0000	377094.00	69.69
LSBTLP-500-51-250	405197.31	0.07	4.31	0.0000	0.0000		
LSBTLP-500-52-250	402383.67	0.07	4.20	0.0000	0.0000		
LSBTLP-500-53-250	381074.04	0.07	4.20	0.0000	0.0000		
LSBTLP-500-54-250	455446.72	0.07	4.08	0.0000	0.0000		
prosek	**411025.44**	**0.07**	**4.20**	0.0000	0.0000	407848.90	68.40
LSBTLP-500-51-375	405579.65	0.04	3.95	0.0000	0.0000		
LSBTLP-500-52-375	402606.56	0.04	3.78	0.0000	0.0000		
LSBTLP-500-53-375	381105.82	0.04	3.74	0.0000	0.0000		
LSBTLP-500-54-375	455803.62	0.04	3.78	0.0000	0.0000		
prosek	**411273.91**	**0.04**	**3.81**	0.0000	0.0000	411231.20	65.00

Na osnovu podataka prikazanih u tabelama može se zaključiti da je predstavljeni PVNS algoritam, prikazan u ovom radu, dostigao bolja rešenja na svakoj grupi instanci, za kraće vreme u odnosu na [49]. Takođe, može se primetiti da su prosečno

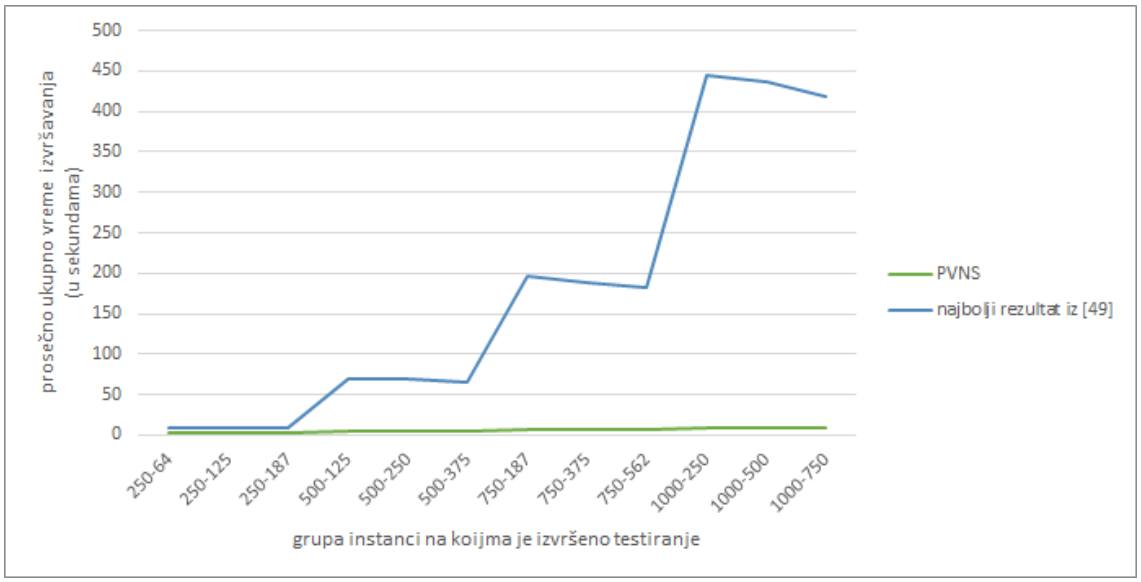
Tabela 3.7: Poređenje rezultata na LSBTLP test instancama za $n = m = 750$ i $n = m = 1000$

	PVNS					najbolje rešenje iz [49]	
instanca	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t[s]$
LSBTLP-750-71-187	824408.59	0.12	5.75	0.0000	0.0000		
LSBTLP-750-72-187	890624.52	0.11	5.84	0.0000	0.0000		
LSBTLP-750-73-187	1051600.18	0.11	5.81	0.0000	0.0000		
LSBTLP-750-74-187	898259.23	3.59	9.26	0.0000	0.0000		
prosek	**916223.13**	**0.98**	**6.66**	0.0000	0.0000	887684.20	196.97
LSBTLP-750-71-375	903043.38	0.14	6.47	0.0000	0.0000		
LSBTLP-750-72-375	961424.85	0.13	6.56	0.0000	0.0000		
LSBTLP-750-73-375	1145333.21	0.14	6.51	0.0000	0.0000		
LSBTLP-750-74-375	971620.77	0.14	6.44	0.0000	0.0000		
prosek	**995355.55**	**0.14**	**6.50**	0.0000	0.0000	983936.20	189.25
LSBTLP-750-71-562	903472.76	0.08	5.72	0.0000	0.0000		
LSBTLP-750-72-562	961812.84	0.08	6.14	0.0000	0.0000		
LSBTLP-750-73-562	1146949.03	0.09	5.99	0.0000	0.0000		
LSBTLP-750-74-562	972257.07	0.08	6.10	0.0000	0.0000		
prosek	**996122.92**	**0.08**	**5.99**	0.0000	0.0000	995704.00	182.00
LSBTLP-1000-101-250	1445607.35	1.16	9.38	0.0000	0.0000		
LSBTLP-1000-102-250	1458600.18	0.40	8.72	0.0000	0.0000		
LSBTLP-1000-103-250	1585280.80	0.47	8.78	0.0015	0.0012		
LSBTLP-1000-104-250	1376970.83	0.19	8.27	0.0000	0.0000		
prosek	**1466614.79**	**0.55**	**8.79**	0.0004	0.0003	1398179.00	444.33
LSBTLP-1000-101-500	1607707.61	0.27	9.54	0.0000	0.0000		
LSBTLP-1000-102-500	1647123.40	0.26	9.57	0.0000	0.0000		
LSBTLP-1000-103-500	1774276.83	0.43	9.61	0.0002	0.0004		
LSBTLP-1000-104-500	1549146.48	0.25	9.33	0.0000	0.0000		
prosek	**1644563.58**	**0.30**	**9.51**	0.0000	0.0001	1611487.00	437.50
LSBTLP-1000-101-750	1609095.48	0.15	9.43	0.0000	0.0000		
LSBTLP-1000-102-750	1650841.61	0.16	9.17	0.0000	0.0000		
LSBTLP-1000-103-750	1779328.92	0.19	9.55	0.0000	0.0000		
LSBTLP-1000-104-750	1552568.65	0.16	9.46	0.0000	0.0000		
prosek	**1647958.66**	**0.16**	**9.40**	0.0000	0.0000	1646697.00	417.50

odstupanje i standardna devijacija jednaki nuli na svakoj osim za dve instance. To ukazuje da su na većini instanci različita pokretanja algoritma dala najbolje rešenje, nezavisno od vrednosti početnog semena za generisanje pseudo-slučajnih brojeva i može se zaključiti da je predstavljeni PVNS algoritam veoma stabilan prilikom rešavanja instanci predloženih u [49]. Iako je cilj korektno poređenje rezultata dobijenih u [49] sa rezultatima PVNS algoritma, faktor skaliranja između procesora nije moguće precizno odrediti budući da tačan model procesora nije naveden u [49]. Međutim, poređenjem prosečnog vremena izvršavanja PVNS algoritma i algoritma koji je dao najbolje rezultate u [49], može se videti da je vreme izvršavanja PVNS algoritma u proseku 32 puta manje i može se zaključiti da faktor skaliranja ne bi značajno uticao na rezultate. Grafički prikaz rezultata prikazan je na slikama 3.2 i 3.3.



Slika 3.2: Grafički prikaz poređena rezultata dobijenih PVNS metodom i najboljih rezultata iz [49]



Slika 3.3: Grafički prikaz poređena ukupnog vremena izvršavanja PVNS metode i metode koja je dala najbolje rezultate u [49]

Detaljni rezultati

Tabele 3.8, 3.9 i 3.10 sadrže eksperimentalne rezultate PVNS algoritma na instancama iz [49] i prilagođenim *rl* instancama iz TSPLIB biblioteke. Rezultati su dobijeni izvršavanjem testova sa 20 različitih vrednosti semena za generisanje

pseudo-slučajnih brojeva. Kolone u tabelama 3.8, 3.9 i 3.10 imaju sledeću notaciju. Prva kolona predstavlja ime instance, dok druga kolona predstavlja najbolje dobijeno rešenje. U trećoj i četvrtoj koloni dati su vreme potrebno za pronalazak najboljeg rešenja u sekundama, kao i ukupno vreme izvršavanja PVNS algoritma u sekundama kod pronalaska najboljeg rešenja. Peta i šesta kolona sadrže vrednosti prosečnog odstupanja i standardne devijacije rezultata dobijenih PVNS algoritmom. Može se primetiti da su vrednosti najboljih rešenja PVNS algoritma u tabeli 3.8 slični vrednostima najboljih rešenja PVNS algoritma u tabelama 3.6 i 3.7. Razlika je u broju izvršavanja algoritma u eksperimentu jer su rezultati u tabelama 3.6 i 3.7 korišćeni za poređenje sa rezultatima iz [49]. U tabelama 3.9 i 3.10 se može videti da su vrednosti prosečnog odstupanja i standardne devijacije veoma mali, što navodi da je predstavljeni PVNS algoritam stabilan i prilikom rešavanja instanci većih dimenzija.

Grafički prikazi vremena potrebnog za nalazak najboljeg rešenja i ukupnog vremena izvršavanja PVNS metode, na prilagođenim *rl* instancama TSPLIB biblioteke, u zavisnosti od broja korisnika stanica i poluprečnika dostupnog prostora oko centara, dati su na slikama 3.4, 3.5, 3.6 i 3.7. Na osnovu prikazanih vremena može se zaključiti da su najkomplikovanije instance za rešavanje one koje imaju $d \in [700, 1000]$ korisnika. Može se videti i da su instance sa većim poluprečnikom dostupnog prostora oko centara uglavnom teže za rešavanje od onih sa manjim poluprečnikom.

Tabela 3.8: Eksperimentalni rezultati na LSBTLP instancama

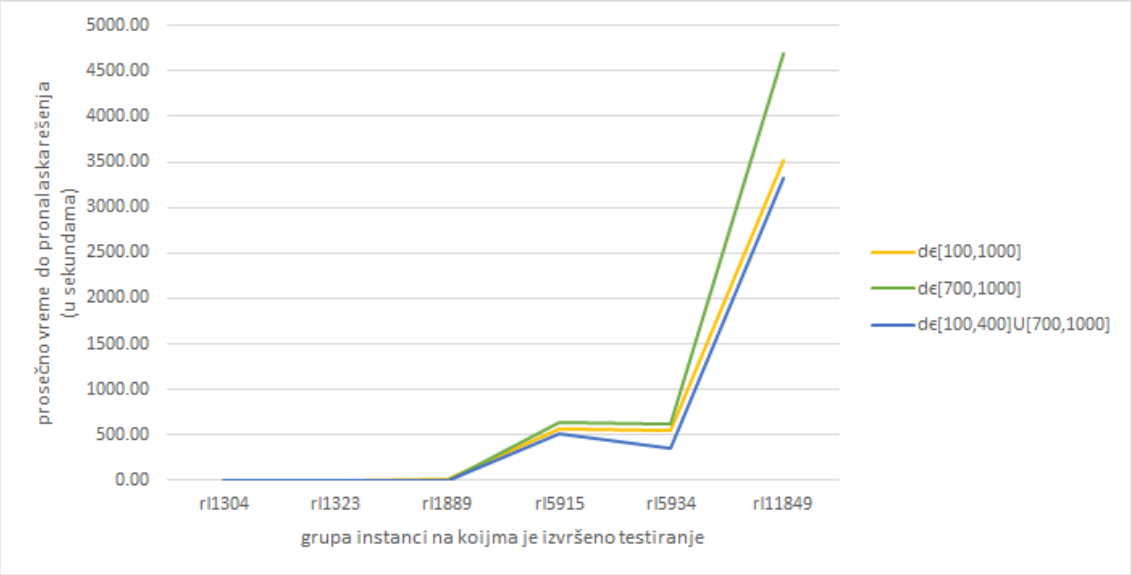
instanca	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$
LSBTLP-250-22-64	93865.90	0.01	1.68	0.0000	0.0000
LSBTLP-250-23-64	140302.52	0.01	1.79	0.0000	0.0000
LSBTLP-250-24-64	147450.59	0.01	1.90	0.0000	0.0000
LSBTLP-250-22-125	96721.32	0.02	2.10	0.0000	0.0000
LSBTLP-250-23-125	145438.69	0.02	2.24	0.0000	0.0000
LSBTLP-250-24-125	150529.17	0.02	2.29	0.0000	0.0000
LSBTLP-250-22-187	96767.01	0.01	2.10	0.0000	0.0000
LSBTLP-250-23-187	145454.03	0.01	2.09	0.0000	0.0000
LSBTLP-250-24-187	150548.10	0.01	2.12	0.0000	0.0000
LSBTLP-500-51-125	378327.97	0.05	4.06	0.0000	0.0000
LSBTLP-500-52-125	380905.74	0.04	3.72	0.0000	0.0000
LSBTLP-500-53-126	360714.67	0.04	3.77	0.0000	0.0000
LSBTLP-500-54-126	425967.69	0.04	3.68	0.0000	0.0000
LSBTLP-500-51-250	405197.31	0.06	4.02	0.0000	0.0000
LSBTLP-500-52-250	402383.67	0.06	3.94	0.0000	0.0000
LSBTLP-500-53-250	381074.04	0.06	3.97	0.0000	0.0000
LSBTLP-500-54-250	455446.72	0.06	4.11	0.0000	0.0000
LSBTLP-500-51-375	405579.65	0.03	3.74	0.0000	0.0000
LSBTLP-500-52-375	402606.56	0.03	3.63	0.0000	0.0000
LSBTLP-500-53-375	381105.82	0.03	3.53	0.0000	0.0000
LSBTLP-500-54-375	455803.62	0.03	3.62	0.0000	0.0000
LSBTLP-750-71-187	824408.59	0.09	5.36	0.0000	0.0000
LSBTLP-750-72-187	890624.52	0.09	5.66	0.0000	0.0000
LSBTLP-750-73-187	1051600.18	0.09	5.50	0.0000	0.0000
LSBTLP-750-74-187	898259.23	0.09	5.63	0.0000	0.0000
LSBTLP-750-71-375	903043.38	0.12	6.30	0.0000	0.0000
LSBTLP-750-72-375	961424.85	0.12	6.46	0.0000	0.0000
LSBTLP-750-73-375	1145333.21	0.13	6.19	0.0000	0.0000
LSBTLP-750-74-375	971620.77	0.12	6.54	0.0000	0.0000
LSBTLP-750-71-562	903472.76	0.07	5.64	0.0000	0.0000
LSBTLP-750-72-562	961812.84	0.07	5.55	0.0000	0.0000
LSBTLP-750-73-562	1146949.03	0.07	5.60	0.0000	0.0000
LSBTLP-750-74-562	972257.07	0.07	5.60	0.0000	0.0000
LSBTLP-1000-101-250	1445607.35	0.16	8.14	0.0000	0.0000
LSBTLP-1000-102-250	1458600.18	0.17	8.27	0.0000	0.0000
LSBTLP-1000-103-250	1585280.80	0.17	8.14	0.0010	0.0012
LSBTLP-1000-104-250	1376970.83	0.17	8.28	0.0000	0.0000
LSBTLP-1000-101-500	1607707.61	0.23	9.22	0.0000	0.0000
LSBTLP-1000-102-500	1647123.40	0.23	9.14	0.0000	0.0000
LSBTLP-1000-103-500	1774276.83	0.23	9.06	0.0002	0.0004
LSBTLP-1000-104-500	1549146.48	0.23	8.87	0.0000	0.0000
LSBTLP-1000-101-750	1609095.48	0.14	8.72	0.0000	0.0000
LSBTLP-1000-102-750	1650841.61	0.14	9.01	0.0000	0.0000
LSBTLP-1000-103-750	1779328.92	0.13	8.99	0.0000	0.0000
LSBTLP-1000-104-750	1552568.65	0.14	8.74	0.0000	0.0000

Tabela 3.9: Eksperimentalni rezultati na rl1304, rl1323, rl1889 TSPLIB instancama

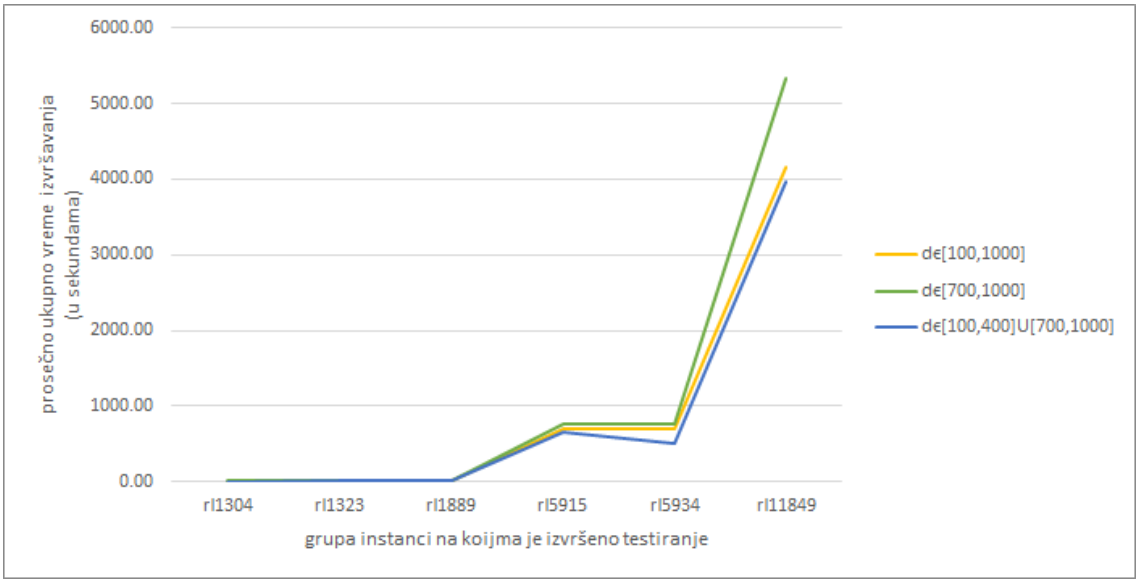
instanca	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$
rl1304_100_1000.163.2000	319457.41	0.09	5.92	0.0000	0.0000
rl1304_100_1000.163.3000	332781.07	0.13	7.88	0.0000	0.0000
rl1304_100_1000.217.2000	325269.36	0.10	5.90	0.0002	0.0004
rl1304_100_1000.217.3000	336860.09	0.15	8.15	0.0001	0.0002
rl1304_700_1000.163.2000	489046.26	0.28	6.24	0.0000	0.0000
rl1304_700_1000.163.3000	510063.02	1.09	9.45	0.0000	0.0000
rl1304_700_1000.217.2000	498640.05	0.38	6.43	0.0002	0.0007
rl1304_700_1000.217.3000	516814.34	0.16	8.13	0.0000	0.0000
rl1304_100_400_700_1000.163.2000	252248.35	0.10	5.96	0.0000	0.0000
rl1304_100_400_700_1000.163.3000	262279.92	0.14	8.22	0.0000	0.0000
rl1304_100_400_700_1000.217.2000	256615.11	0.11	6.01	0.0000	0.0000
rl1304_100_400_700_1000.217.3000	265353.78	0.16	8.08	0.0000	0.0000
rl1323_100_1000.165.2000	323716.48	0.19	6.06	0.0000	0.0000
rl1323_100_1000.165.3000	337974.11	0.20	8.15	0.0002	0.0010
rl1323_100_1000.220.2000	329927.90	0.12	6.35	0.0000	0.0001
rl1323_100_1000.220.3000	342344.61	0.23	8.56	0.0000	0.0001
rl1323_700_1000.165.2000	492593.07	0.16	6.43	0.0000	0.0000
rl1323_700_1000.165.3000	515192.57	0.43	8.55	0.0000	0.0000
rl1323_700_1000.220.2000	503007.11	0.12	6.14	0.0001	0.0005
rl1323_700_1000.220.3000	522539.65	0.19	8.51	0.0001	0.0002
rl1323_100_400_700_1000.165.2000	247882.94	0.26	6.12	0.0000	0.0000
rl1323_100_400_700_1000.165.3000	258464.17	0.41	8.60	0.0003	0.0011
rl1323_100_400_700_1000.220.2000	251932.00	0.12	6.62	0.0009	0.0015
rl1323_100_400_700_1000.220.3000	261328.68	0.32	8.45	0.0001	0.0003
rl1889_100_1000.236.2000	463225.59	2.67	11.86	0.0003	0.0007
rl1889_100_1000.236.3000	479423.47	3.84	17.04	0.0002	0.0004
rl1889_100_1000.315.2000	469924.17	4.34	13.94	0.0007	0.0006
rl1889_100_1000.315.3000	484108.24	14.88	28.65	0.0005	0.0003
rl1889_700_1000.236.2000	721418.72	0.93	10.28	0.0005	0.0021
rl1889_700_1000.236.3000	747730.37	4.03	17.43	0.0001	0.0002
rl1889_700_1000.315.2000	732624.89	0.69	10.48	0.0000	0.0000
rl1889_700_1000.315.3000	755566.96	1.00	14.71	0.0001	0.0006
rl1889_100_400_700_1000.236.2000	361163.64	1.72	10.73	0.0005	0.0008
rl1889_100_400_700_1000.236.3000	373857.00	3.20	16.80	0.0006	0.0009
rl1889_100_400_700_1000.315.2000	365617.30	0.33	9.80	0.0001	0.0003
rl1889_100_400_700_1000.315.3000	376966.80	0.54	14.53	0.0001	0.0002

Tabela 3.10: Eksperimentalni rezultati na rl5915, rl5934, rl11849 rl TSPLIB instancama

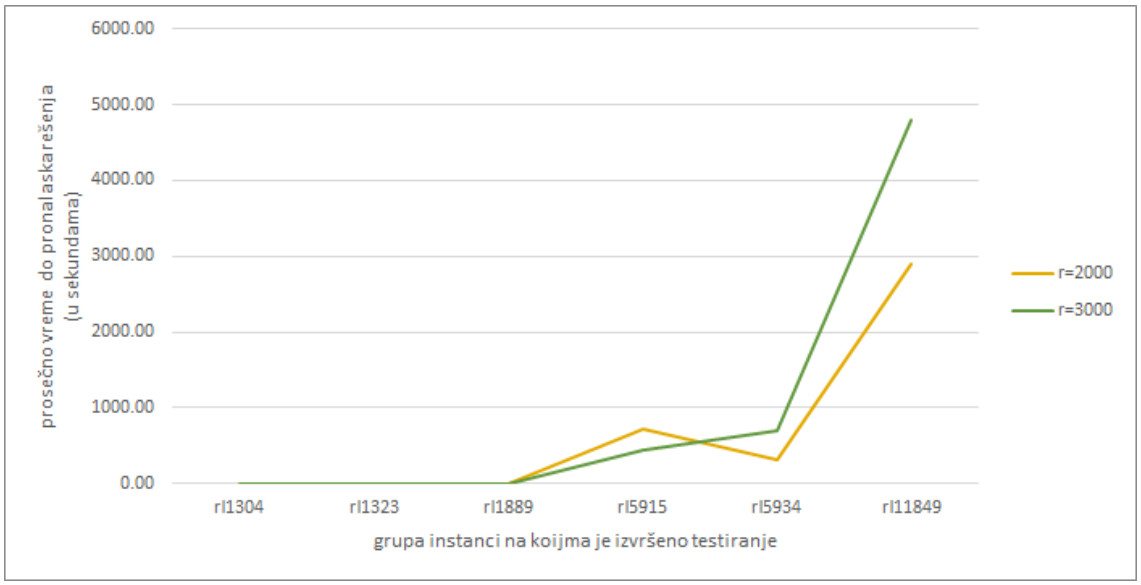
instanca	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$
rl5915_100_1000.739.2000	1520381.16	670.88	774.15	0.0029	0.0021
rl5915_100_1000.739.3000	1552703.96	707.25	887.79	0.0015	0.0008
rl5915_100_1000.986.2000	1534062.58	217.88	328.08	0.0009	0.0007
rl5915_100_1000.986.3000	1562077.07	631.51	818.45	0.0004	0.0004
rl5915_700_1000.739.2000	2352873.20	365.16	471.94	0.0025	0.0020
rl5915_700_1000.739.3000	2404980.36	899.95	1078.82	0.0019	0.0013
rl5915_700_1000.986.2000	2374782.91	461.34	567.87	0.0017	0.0011
rl5915_700_1000.986.3000	2419993.28	799.52	979.10	0.0012	0.0007
rl5915_100_400_700_1000.739.2000	1185992.30	563.89	668.13	0.0020	0.0012
rl5915_100_400_700_1000.739.3000	1210764.11	612.79	789.94	0.0017	0.0008
rl5915_100_400_700_1000.986.2000	1195318.82	238.46	346.69	0.0010	0.0007
rl5915_100_400_700_1000.986.3000	1217162.93	598.17	787.75	0.0007	0.0005
rl5934_100_1000.741.2000	1535771.51	352.09	463.55	0.0021	0.0015
rl5934_100_1000.741.3000	1567495.70	972.20	1161.92	0.0011	0.0010
rl5934_100_1000.989.2000	1548060.66	80.71	191.64	0.0004	0.0002
rl5934_100_1000.989.3000	1575915.98	808.39	994.64	0.0004	0.0003
rl5934_700_1000.741.2000	2359212.30	511.53	616.41	0.0027	0.0018
rl5934_700_1000.741.3000	2409947.84	979.80	1163.04	0.0016	0.0008
rl5934_700_1000.989.2000	2379976.48	329.79	442.51	0.0017	0.0012
rl5934_700_1000.989.3000	2424159.89	646.62	837.10	0.0008	0.0006
rl5934_100_400_700_1000.741.2000	1206897.55	357.82	465.45	0.0015	0.0013
rl5934_100_400_700_1000.741.3000	1231190.83	435.25	620.06	0.0011	0.0007
rl5934_100_400_700_1000.989.2000	1215977.65	223.77	335.66	0.0010	0.0006
rl5934_100_400_700_1000.989.3000	1237409.59	401.78	603.80	0.0006	0.0004
rl11849_100_1000.1481.2000	3086983.50	3026.81	3482.84	0.0022	0.0009
rl11849_100_1000.1481.3000	3138568.67	3810.97	4590.26	0.0010	0.0006
rl11849_100_1000.1974.2000	3104667.00	2951.53	3433.60	0.0007	0.0005
rl11849_100_1000.1974.3000	3150611.23	4303.58	5097.21	0.0003	0.0003
rl11849_700_1000.1481.2000	4786009.18	3530.73	3986.19	0.0023	0.0010
rl11849_700_1000.1481.3000	4868863.75	7948.66	8725.25	0.0009	0.0006
rl11849_700_1000.1974.2000	4815905.43	3761.40	4235.26	0.0013	0.0007
rl11849_700_1000.1974.3000	4889233.48	3537.01	4339.46	0.0006	0.0003
rl11849_100_400_700_1000.1481.2000	2405577.71	1973.52	2436.45	0.0015	0.0012
rl11849_100_400_700_1000.1481.3000	2445009.81	4910.30	5719.57	0.0009	0.0006
rl11849_100_400_700_1000.1974.2000	2417967.54	2113.54	2612.37	0.0007	0.0004
rl11849_100_400_700_1000.1974.3000	2453443.45	4257.09	5084.94	0.0005	0.0003



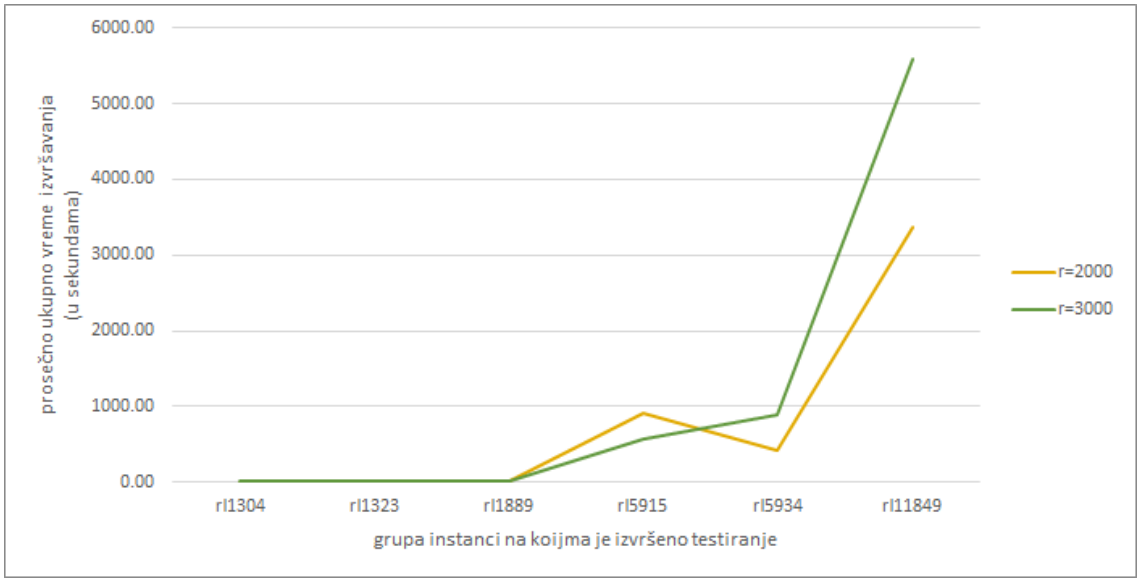
Slika 3.4: Grafički prikaz potrebnog vremena do pronalaska najboljeg rešenja PVNS metode u zavisnosti od broja korisnika stanica



Slika 3.5: Grafički prikaz ukupnog vremena izvršavanja PVNS metode u zavisnosti od broja korisnika stanica



Slika 3.6: Grafički prikaz potrebnog vremena do pronalaska najboljeg rešenja PVNS metode u zavisnosti od poluprečnika dostupnog prostora oko centara



Slika 3.7: Grafički prikaz ukupnog vremena izvršavanja PVNS metode u zavisnosti od poluprečnika dostupnog prostora oko centara

Glava 4

Problem uspostavljanja centara za produženu negu pacijenata

Zdravstvena zaštita postala je važna oblast primene operacionih istraživanja i odnosi se na dizajniranje i optimizaciju sistema zdravstvene zaštite, kao i na same usluge zdravstvene zaštite. Zdravstvena zaštita značajna je delatnost koja uključuje angažovanje velikog broja ljudi, kako zaposlenih u organizacijama koje pružaju usluge zdravstvene zaštite tako i korisnika usluga zdravstvene zaštite. Rastući troškovi nastali usled razvoja novih tehnologija i demografskih trendova (naročito starenja populacije), predstavljaju značajan problem prilikom određivanja politika savremene zdravstvene zaštite. U poslednjih deset godina postaje sve prisutnija primena operacionih istraživanja za smanjenje troškova u sistemu zdravstvene zaštite.

U ovoj glavi razmatran je *problem uspostavljanja centara za produženu negu pacijenata* (engl. Long-term Care Facility Location Problem - LTCFLP), sa ciljem da se minimizuje maksimalan broj pacijenata dodeljen jednom uspostavljenom centru. Rezultati iz ove glave su prikazani u radu [33]. LTCFLP je prvi put predstavljen u radu [77] publikovanom 2010. godine. Motivacija za nastanak ovog problema je pronalazak lokacija za izgradnju centara za produženu negu starih lica u Južnoj Koreji. LTCFLP se odnosi na minimizaciju maksimalnog opterećenja centra koji pruža produženu medicinsku zaštitu. Dat je skup lokacija potencijalnih centara, koji se poklapa sa skupom lokacija klijenata. Potražnja svakog klijenta definisana je kao broj pacijenata koji se nalaze na lokaciji klijenta. Broj centara koje je potrebno uspostaviti je ograničen određenom konstantom i pretpostavlja se da nema prethodno uspostavljenih centara. Ograničenja u pogledu kapaciteta i fiksni troškovi za date centre nisu korišćeni. Grupa pacijenata može biti opslužena od strane tačno jednog

uspostavljenog centra, i to onog koji je najbliži. Potrebno je pronaći optimalne lokacije za izgradnju centara tako da maksimalno opterećenje uspostavljenih centara bude minimizovano.

Sličan pristup se može primeniti i u drugim problemima koji proističu iz sistema zdravstvene zaštite, kao što je lociranje primarnih objekata zdravstvene zaštite, klinika i objekata za zdravstveno savetovanje. Druge oblasti primene LTCFLP i njegovih varijanti uključuju, na primer, određivanje optimalnih lokacija različitih javnih službi u okviru gradske oblasti ili regiona (kao što su škole, vrtići, sportski centri, šoping centri), dizajniranje telekomunikacionih i računarskih mreža, pronalaženje optimalnih lokacija telekomunikacionih ili računarskih habova, itd.

4.1 Matematička formulacija problema

U ovoj sekciji prikazana je matematička formulacija LTCFLP problema. U odnosu na formulaciju problema predstavljenu u [77] u ovoj sekciji je prikazana unapređena formulacija u smislu manjeg broja promenljivih korišćenih u modelu, kao i zbog uklanjanja velike konstante M iz uslova problema.

Skup lokacija na kojima se nalaze grupe pacijenata označen je sa J, što ujedno predstavlja i skup kandidata za izgradnju centara za produženu negu pacijenata. Sa c_{ij} označena je razdaljina između lokacije grupe pacijenata $j \in J$ i kandidata za izgradnju centra $i \in J$, dok d_j predstavlja broj pacijenata na lokaciji $j \in J$. Ceo broj $K > 0$ predstavlja maksimalni broj centara koji je moguće uspostaviti. Promenljiva $x_{ij} \in \{0, 1\}$ ima vrednost 1 ukoliko je klijent $j \in J$ pridružen uspostavljenom centru $i \in J$, 0 inače. Ukoliko je centar i aktivan, grupa pacijenata na lokaciji i biće pridružena centru na toj istoj lokaciji. Samim tim promenljiva x_{ii} se može, osim same provere pridruživanja, iskoristiti i kao indikator da li je centar i uspostavljen ili nije. Nenegativna realna promenljiva L_{max} predstavlja maksimalno opterećenje uspostavljenog centra.

Koristeći navedenu notaciju, LTCFLP može biti formulisan kao:

$$min \;\; L_{max} \tag{4.1}$$

pri uslovima:

$$\sum_{i \in J} x_{ij} = 1 \qquad j \in J, \tag{4.2}$$

$$x_{ij} \leq x_{ii} \qquad i, j \in J, \tag{4.3}$$

$$\sum_{i \in J: c_{ij} \leq c_{ik}} x_{ij} \geq x_{kk} \qquad j, k \in J, \tag{4.4}$$

$$\sum_{i \in J} x_{ii} \leq K, \tag{4.5}$$

$$\sum_{j \in J} d_j x_{ij} \leq L_{max} \qquad i \in J, \tag{4.6}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in J. \tag{4.7}$$

Funkcija cilja (4.1) minimizuje maksimalno opterećenje uspostavljenog centra. Ograničenja (4.2) garantuju da će svaka grupa pacijenata biti pridružena tačno jednom centru. Uslovi (4.3) obezbeđuju da se pridruživanje vrši samo ka uspostavljenim centrima. Svaka grupa pacijenata pridružena je svom najbližem uspostavljenom centru, što je omogućeno uslovima (4.4). Ograničenje (4.5) garantuje da ukupan broj uspostavljenih centara ne prelazi predodređenu gornju granicu K. Uslovi (4.6) označavaju donju granicu promenljive L_{max}, koja predstavlja maksimalno opterećenje uspostavljenih centara. Na kraju, (4.7) ukazuju na binarnu prirodu promenljivih x_{ij}.

U tabeli 4.1 prikazano je 17 lokacija određenih x i y koordinatama i broj pacijenata na svakoj od lokacija. Neka je distanca c_{ij} između dve lokacije definisana euklidskim rastojanjem između dve odgovarajuće tačke. Neka je potrebno uspostaviti najviše 4 centara, tj. $K = 4$. Tada je optimalno rešenje LTCFLP problema uspostaviti centre na lokacijama j_{10}, j_{11}, j_{13} i j_{14} i funkcija cilja ima vrednost 356. Grafički prikaz ovog rešenja je dat na slici 4.1. Crvenom bojom su prikazane lokacije na kojima su uspostavljeni centri. Može se zapaziti da opterećenja uspostavljenih centara imaju vrednosti 356, 330, 314 i 327, što pokazuje dobru izbalansiranost opterećenja centara.

4.2 Pregled relevantne literature za LTCFLP

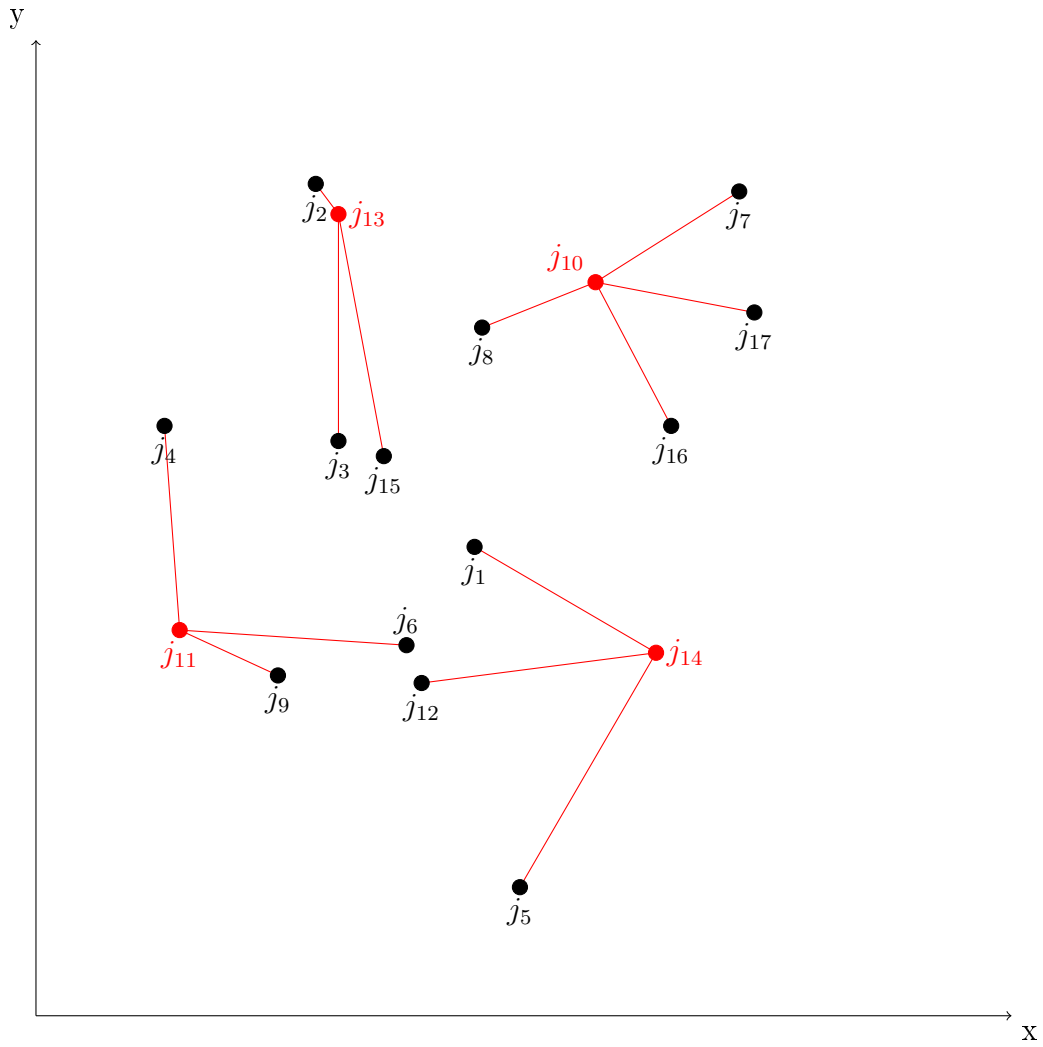
U literaturi postoji veliki broj problema operacionih istraživanja u sistemima zdravstvene zaštite koji se odnose na dizajn i efikasnost urgentnih medicinskih

Tabela 4.1: Lista klijenata i potencijalnih centara

klijent	x koordinata	y koordinata	broj pacijenata
j_1	43.00	47.00	73
j_2	22.00	95.00	96
j_3	25.00	61.00	62
j_4	2.00	63.00	64
j_5	49.00	2.00	90
j_6	34.00	34.00	73
j_7	78.00	94.00	53
j_8	44.00	76.00	81
j_9	17.00	30.00	100
j_{10}	59.00	82.00	70
j_{11}	4.00	36.00	93
j_{12}	36.00	29.00	97
j_{13}	25.00	91.00	78
j_{14}	67.00	33.00	67
j_{15}	31.00	59.00	78
j_{16}	69.00	63.00	57
j_{17}	80.00	78.00	95

usluga. Modeli lokacijskih problema se veoma primenjuju u realnim problemima, a uključuju postavljanje i reagovanje centara koji koji pružaju različite hitne usluge, kao što su zdravstvene, policijske, vatrogasne stanice [38, 53, 120, 21]. Dizajn takvih sistema obuhvata optimalno planiranje centara koji su odgovorni za preduzimanje mera u slučaju incidenata tako da su jedan ili više ciljeva minimizovani. Tipični ciljevi su minimizacija prosečnog ili maksimalnog vremena pružanja hitne usluge ili maksimizacija prostora koji svaki centar opslužuje [53, 21].

U literaturi se mogu naći različiti lokacijski problemi čija funkcija cilja uključuje balansiranje opterećenja centara pod određenim uslovima. U radu [10] istražuje se problem lociranja fiksnog broja centara po jedinici površine kako bi se minimizovalo maksimalno opterećenje lociranog centra, pretpostavljajući pridruživanje najbližem centru i ograničenje u pogledu pokrivenosti. U radu [95] predlaže se diskretan lokacijski problem koji sadrži uspostavljanje fiksnog broja centara. Svaki klijent je pridružen svom najbližem uspostavljenom centru. Štaviše, broj korisnika koji su pridruženi svakom uspostavljenom centru treba da bude ujednačen. Umesto primene MIN-MAX funkcije cilja, ujednačenost između dobavljača povećana je korišćenjem funkcije cilja zasnovanoj na rasponu, tj. minimizaciji razlika između centara sa maksimalnim brojem dodeljenih klijenata i centara sa minimalnim bro-

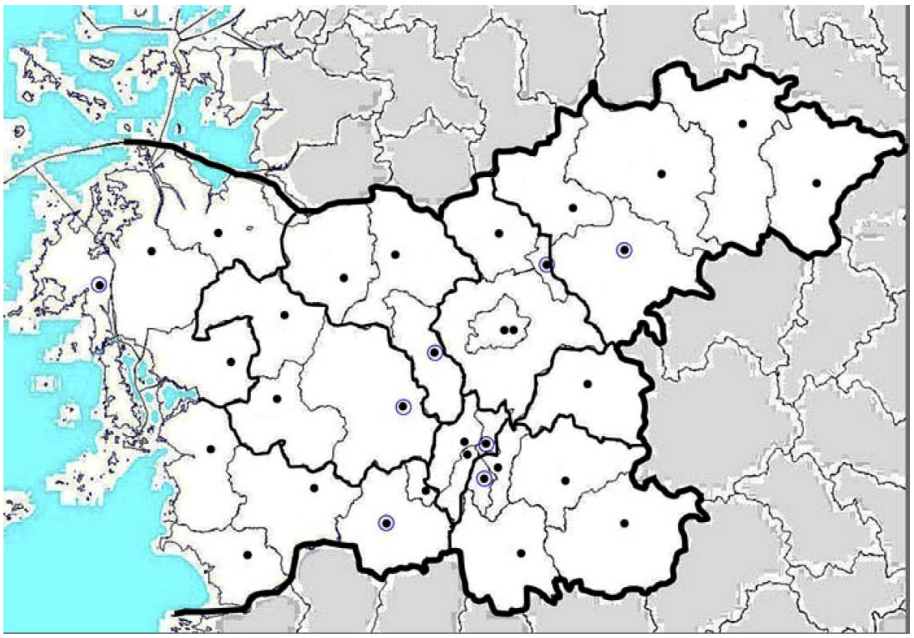


Slika 4.1: Grafički prikaz rešenja LTCFLP problema

jem dodeljenih klijenata.

Autori su u radu [77] predstavili matematičku formulaciju LTCFLP i MADI heuristiku (engl. Modified Add-Drop-Interchange) za rešavanje LTCFLP. Rešenje koje generiše MADI heuristika se koristi kao gornja granica za metodu grananja i ograničavanja (engl. Branch and Bound - BnB). Eksperimentalni rezultati izvedeni su na instanci zasnovanoj na jednoj provinciji Južne Koreje koja sadrži 33 potencijalne lokacije za izgradnju centara (slika 4.2), kao i na većem broju kreiranih test instanci koje sadrže do 70 potencijalnih lokacija za izgradnju centara. BnB metoda je dala optimalna rešenja na instancama problema manjih dimenzija, dok je MADI heuristika proizvela rešenja sa određenim odstupanjima u odnosu na optimalna re-

šenja, posebno na instancama koje sadrže $50-70$ potencijalnih lokacija za izgradnju centara.



Slika 4.2: Potencijalne lokacije za izgradnju zdravstvenih centara u Čunčong provinciji, Južna Koreja

U radu [127] predložen je evolutivni algoritam (EA) za rešavanje LTCFLP. Predloženi EA koristi odgovarajuće evolutivne operatore i više različitih strategija za sprečavanje prevremene konvergencije ka lokalnom minimumu. EA je testiran na realnoj instanci sa 33 potencijalne lokacije za izgradnju centara, i na modifikovanim ORLIB hub AP instancama iz [13] sa do 80 potencijalnih lokacija za uspostavljanje centara. Kako bi se proverila robusnost EA, algoritam je testiran na AP baziranim instancama velikih razmera sa do 400 potencijalnih lokacija za izgradnju centara. Rezultati EA upoređeni su sa rezultatima BnB metode i MADI heuristike na dostupnim malim test instancama iz [77], i optimalnim rezultatima predstavljenim u [127] dobijenim CPLEX rešavačem. Eksperimentalni rezultati pokazali su da je EA dostigao sva poznata optimalna rešenja, dobijena BnB metodom ili CPLEX rešavačem, i nadmašio MADI heuristiku. Za instance velikih razmera sa 400 potencijalnih centara, EA je dostizao svoja najbolja rešenja za kratko procesorsko vreme.

U radu [90] predložen je hibridni algoritam nazvan EA-VNS, koji kombinuje evolutivni pristup iz [127] sa metodom promenljivih okolina za rešavanje LTCFLP. Najbolje rešenje dobijeno EA algoritmom iz [127] iskorišćeno je kao početno rešenje

za VNS algoritam. Eksperimentalni rezultati, sprovedeni na istim test podacima kao u [127], pokazali su da je EA-VNS hibrid dao rezultate visokog kvaliteta u smislu oba, kvaliteta rešenja i iskorišćenog procesorskog vremena. Dobijeni rezultati pokazali su da je EA-VNS metod predložen u radu [90] bolji za rešavanje LTCFLP u odnosu na EA pristup iz [127], što se naročito pokazalo na većim test instancama.

4.3 Metoda promenljivih okolina za LTCFLP

Predloženi algoritam se sastoji iz dve faze: redukovane metode promenljivih okolina i osnovne metode promenljivih okolina [65, 101]. U prvoj fazi, RVNS metoda brzo pronalazi dobro inicijalno rešenje koje se dalje koristi kao početno rešenje za drugu fazu, odnosno osnovni VNS algoritam.

Rešenje problema kodirano je nizom binarnih brojeva dužine $n = |J|$, gde J predstavlja skup lokacija u problemu, numerisan sa $J = \{0, 1, ..., n-1\}$. Svaki binaran broj u rešenju res predstavlja jednu lokaciju. Vrednost $res(j) = 1$, $j \in \{0, 1, ..., n-1\}$, označava da je centar na lokaciji j uspostavljen, dok $res(j) = 0$ označava da nije. Kada su indeksi uspostavljenih centara poznati, pridruživanje grupa pacijenata najbližem uspostavljenom centru se dobija poređenjem odgovarajućih rastojanja. Nakon što je opterećenje svakog centra poznato, vrednost funkcije cilja se dobija poređenjem izračunatih opterećenja i određivanjem maksimalne vrednosti.

Korišćena je struktura okolina zasnovana na zameni uspostavljenog i neuspostavljenog centra. Tačnije jedna zamena se sastoji od zatvaranja jednog uspostavljenog i otvaranja drugog neuspostavljenog centra. Rešenje res_1 se nalazi u k-toj okolini rešenja res_2 ako se rešenje res_2 može dobiti iz rešenje res_1 koristeći najviše k zamena centara.

Osnovna prednost metode promenljivih okolina prikazane u ovom radu u poređenju sa VNS metodom predstavljenom u [90] je implementacija brze promene okolina u fazi lokalne pretrage. Smanjena vremenska složenost utiče na dobijanje boljih rešenja, zbog mogućnosti da se veći broj iteracija algoritma izvrši za kraće procesorsko vreme. U nastavku ove sekcije će svi aspekti predstavljenog VNS algoritma biti detaljno objašnjeni.

Struktura predstavljenog algoritma

Početno dopustivo rešenje za RVNS algoritam je slučajno generisano, tako da sadrži tačno p, $p \leq K$ jedinica u binarnom kodu rešenja, slučajno raspoređenih u nizu, dok ostalih $n - p$ bitova ima vrednost 0. U RVNS fazi algoritma se na slučajan način traži unapređenje trenutnog rešenja u njegovoj k-toj okolini, $k = 1, ..., k_{rvns\_max}$. U ovoj implementaciji vrednost parametra k_{rvns_max} je jednaka 2, što je dobijeno eksperimentalnim testiranjem različitih vrednosti parametara. RVNS algoritam se izvršava sve dok se ne ponovi 1000 uzastopnih iteracija bez popravke rezultata.

Najbolje rešenje dobijeno u RVNS fazi algoritma se koristi kao početno rešenje osnovnog VNS algoritma. U fazi razmrdavanja, trenutno najbolje rešenje se slučajno pomera u svoju k-tu okolinu, $k = 1, ..., k_{max}$, nakon čega se primenjuje lokalna pretraga. Ako je trenutno najbolje rešenje poboljšano, poboljšanje se prihvata i pretraga se nastavlja od novog boljeg rešenja. U suprotnom, trenutno najbolje rešenje ostaje isto, ali se menja veličina okoline. Maksimalna veličina okoline označena je parametrom k_{max}, gde je $k_{max} = \min\{|L|/3, 20\}$. Osnovna VNS petlja se ponavlja sve do zadovoljenja kriterijuma zaustavljanja (1000 uzastopnih iteracija bez popravke rezultata).

Unapređena lokalna pretraga

Lokalna pretraga VNS faze EA-VNS algoritma predloženog u [90] uzima u obzir skup uspostavljenih centara L, i istražuje prvu okolinu tog rešenja u potrazi za poboljšanjima. Preciznije lokalna pretraga iz [90] zatvara jedan centar na lokaciji $t_{out} \in L$, i istovremeno otvara drugi na lokaciji $t_{in} \in J \setminus L$. U cilju računanja funkcije cilja novog rešenja, neophodno je pronaći najbliži uspostavljeni centar za svakog klijenta i izvršiti neophodne dodele. Ovaj korak ima vremensku složenost $O(np)$, gde su $n = |J|$ i $p = |L|$, $p \leq K$. Lokalna pretraga korišćena u [90] razmatra sve moguće zamene, tj. njih $p(n - p)$. Zbog toga je vremenska složenost u [90] pronalaska najbolje zamene jednaka $O(p^2 \cdot n \cdot (n - p)) = O(K^2 \cdot n^2)$.

U predloženoj lokalnoj pretrazi implementirana je varijanta brze zamene okolina, koja se pokazala znatno efikasnijom u odnosu na klasičnu lokalnu pretragu u [90]. Motivacija za korišćenje brze zamene okolina u okviru lokalne pretrage je unapređenje efikasnosti računanja vrednosti funkcije cilja u potrazi za poboljšanjima. Metoda brze zamene okolina je predložena u [142] za efikasno rešavanje problema klasterova-

nja i problema medijane. Ovaj pristup je prvi put primenjen u VNS metodi u radu [59] iz 1997. godine za rešavanje problema p-medijane. U ovom radu, korišćena je ideja iz rada [59] i dizajnirana je metoda brze zamene okolina prilagođena LTCFLP problemu, sa ciljem povećanja efikasnosti lokalne pretrage u okviru VNS algoritma.

Pseudokod primenjene lokalne pretrage sa brzom zamenom okolina predstavljen je algoritmom 23. Prvo je primenjena funkcija $izracunajNajbližeCentre$ koja kao povratnu vrednost vraća nizove $prviNajbliži$ i $drugiNajbliži$. Oni sadrže indekse najbližeg i drugog najbližeg uspostavljenog centra za svakog klijenta. Funkcija $najpogodnijiCentarZaZatvaranje$ primenjena je na svaku lokaciju t_{in} koja nije uspostavljena, u cilju pronalaska uspostavljenog centra koji je najbolje zatvoriti t_{out}, ukoliko se t_{in} uspostavlja. Detaljan opis funkcije koja računa najpogodniji centar za zatvaranje dat je u narednoj podsekciji. Promena vrednosti funkcije cilja prilikom zamene centara t_{in} i t_{out} označena je sa Δ. Najbolja zamena u odnosu na sve kandidate u prvoj okolini označena je sa Δ_{best}. Ako Δ_{best} ima negativnu vrednost, zamena za koju postoji poboljšanje je pronađena. Tada se izvršava zamena, neuspostavljeni centar $best_{t_{in}}$ se uspostavlja, dok se uspostavljeni centar $best_{t_{out}}$ zatvara, i vrednost funkcije cilja se smanjuje za Δ_{best}. Na kraju, nizovi $prviNajbliži$ i $drugiNajbliži$ se ažuriraju za novodobijeno rešenje. Opisani postupak se ponavlja sve dok je moguće pronaći poboljšanja u prvim okolinama trenutnog rešenja.

Vremenska složenost pronalaska najbolje zamene u prvoj okolini je jednaka proizvodu vremenske složenosti procedure koja pronalazi najpogodniji centar za zatvaranje i faktora $n - p$. S obzirom na to da je vremenska složenost procedure koja pronalazi najpogodniji centar za zatvaranje jednaka $O(\max(n, K^2))$, jedna iteracija primenjene lokalne pretrage ima vremensku složenosti $O(n \cdot \max(n, K^2))$, što predstavlja značajno unapređenje efikasnosti u odnosu na [90].

Procedura koja pronalazi najpogodniji centar za zatvaranje

Algoritmom 24 prikazan je pseudokod procedure koja pronalazi najpogodniji centar za zatvaranje $t_{out} \in L$, ukoliko se otvara centar $t_{in} \in J \setminus L$. Neka je $L = l_1, ..., l_p$ skup uspostavljenih centara. Za dati centar t_{in} čije se uspostavljanje razmatra kreira se matrica $novoOpterećenje$ dimenzija $p \times p$ koja čuva opterećenja svih otvorenih centara (tj. broj dodeljenih pacijenata) u slučaju da se centar t_{in} uspostavi, a neki od uspostavljenih zatvori. Red i matrice $novoOpterećenje$ čuva opterećenje u slučaju da se centar i zatvori, a centar t_{in} uspostavi. Opterećenje novog otvorenog centra t_{in} se čuva na dijagonali matrice $novoOpterećenje$, na poziciji na kome se

Algoritam 23: Lokalna pretraga za rešavanje LTCFLP problema

Ulaz: *rešenje*
Izlaz: *rešenje*
$(prviNajbliži, drugiNajbliži) \leftarrow$
  $izračunajNajbližeCentre(rešenje)$;
$popravka \leftarrow true$;
while *popravka* **do**
  $popravka \leftarrow false$;
  $\Delta_{best} \leftarrow 0$;
  foreach *neuspostavljeni centar* t_{in} **do**
    $(t_{out}, \Delta) \leftarrow$
      $najpogodnijiCentarZaZatvaranje$
        $(t_{in}, prviNajbliži, drugiNajbliži)$;
    if $\Delta < \Delta_{best}$ **then**
      $\Delta_{best} \leftarrow \Delta$;
      $best_{t_{out}} \leftarrow t_{out}$;
      $best_{t_{in}} \leftarrow t_{in}$;
    end
  end
  if $\Delta_{best} < 0$ **then**
    $popravka \leftarrow true$;
    $rešenje \leftarrow zamena(rešenje, best_{t_{in}}, best_{t_{out}})$;
    $(prviNajbliži, drugiNajbliži) \leftarrow$
      $izračunajNajbližeCentre(rešenje)$;
  end
end

nalazi opterećenje centra koji se zatvara. Inicijalno, matrica *novoOpterećenje* je popunjena odgovarajućim vrednostima opterećenja trenutnog rešenja.

Kreiran je i niz *gubitak*. Vrednost na i-tom elementu niza *gubitak* predstavlja deo opterećenja koji se prebacuje sa centra i na centar t_{in} ukoliko se on uspostavi. Promenljiva *dobit* čuva deo opterećenja (broj pacijenata) koji će u svakom slučaju biti pridruženi novootvorenom centru t_{in}.

Za svaku lokaciju grupe pacijenata l, algoritam proverava kom centru će klijent l biti pridružen, ako se t_{in} otvori i jedan od uspostavljenih centara iz skupa $L = l_1, ..., l_p$ zatvori. Neka se t_{fc} odnosi na prvi najbliži uspostavljeni centar klijentu l, a t_{sc} na drugi najbliži uspostavljeni centar klijentu l. Tada vrednost $c_{t_{fc},l}$ predstavlja rastojanje između l i njemu najbližeg uspostavljenog centra, dok vrednost $c_{t_{sc},l}$ predstavlja rastojanje između l i drugog najbližeg uspostavljenog centra.

Algoritam 24: Procedura koja pronalazi najpogodniji centar za zatvaranje

Ulaz: $t_{in}, rešenje, prviNajbliži, drugiNajbliži$
Izlaz: t_{out}, Δ
for $i \in \{l_1, ..., l_p\}$ **do**
  $gubitak[i] \leftarrow 0$;
  for $j \in \{l_1, ..., l_p\}$ **do**
    $novoOpterećenje[i, j] \leftarrow opterećenje[j]$;
  end
end
$dobit \leftarrow 0$;
foreach *lokacija* l **do**
  $t_{fc} \leftarrow prviNajbliži[l]$;
  $t_{sc} \leftarrow drugiNajbliži[l]$;
  if $c_{t_{in},l} < c_{t_{fc},l}$ **then**
    $gubitak[t_{fc}] += d_l$;
    $dobit += d_l$;
  else if $c_{t_{in},l} > c_{t_{sc},l}$ **then**
    $novoOpterećenje[t_{fc}, t_{fc}] -= d_l$;
    $novoOpterećenje[t_{fc}, t_{sc}] += d_l$;
  end
end
for $i \in \{l_1, ..., l_p\}$ **do**
  $novoOpterećenje[i, i] += dobit$;
  for $j \in \{l_1, ..., l_p\}$ **do**
    $novoOpterećenje[i, j] -= gubitak[j]$;
  end
end
$t_{out} \leftarrow argmin_{i \in \{l_1,...,l_p\}}(max_{j \in \{l_1,...,l_p\}} novoOpterećenje[i, j])$;
$max \leftarrow max_{j \in \{l_1,...,l_p\}} novoOpterećenje[t_{out}, j]$;
$\Delta \leftarrow max - trenutnaFunkcijaCilja$;

Vrednost $c_{t_{in},l}$ jednaka je rastojanju između l i novog centra koji se uspostavlja t_{in}. Neka je sa d_l označen broj pacijenata na lokaciji l. Za svaku lokaciju l razlikuju se tri slučaja.

1. $c_{t_{in},l} < c_{t_{fc},l}$
   Kada je centar t_{in} bliži lokaciji l u poređenju sa prvim najbližim uspostavljenim centrom t_{fc}, grupa pacijenata sa lokacije l se prebacuje sa centra t_{fc} na centar t_{in}. Vrše se neophodne promene na nizu *gubitak* i ažurira se promenljiva *dobit*.

2. $c_{t_{fc},l} \leq c_{t_{in},l} < c_{t_{sc},l}$

U slučaju da se centar t_{fc} zatvara, grupa pacijenata sa lokacije l prelaze na lokaciju t_{in}. Budući da se opterećenje centra t_{in} u slučaju da se t_{fc} zatvara nalazi na istom mestu u matrici *novoOpterećenje* kao i opterećenje centra t_{fc}, ne vrši se izmena u podacima. Sa druge strane, ako se centar t_{fc} ne zatvara, takođe nema izmena u podacima.

3. $c_{t_{sc},l} \leq c_{t_{in},l}$
   U slučaju da se najbliži centar t_{fc} ne zatvara, nema izmena u podacima. U slučaju da se t_{fc} zatvara, grupa pacijenata sa lokacije l se prebacuje sa centra t_{fc} na centar t_{sc}. U ovom slučaju, matica *novoOpterećenje* se menja u skladu sa promenom pridruživanja date grupe pacijenata.

Nakon određivanja neophodnih prelazaka grupa pacijenata, suma opterećenja smeštena u nizu *gubitak* i vrednosti *dobit* se dodaje na odgovarajuća mesta u matricu *novoOpterećenje*. Vrednost *gubitak*[j] se oduzima od opterećenja j-tog centra, što znači da se vrednost *gubitak*[j] oduzima od svakog elementa j-te kolone matrice *novoOpterećenje*. Vrednost *dobit* se dodaje na opterećenje novog uspostavljenog centra, što znači da se vrednost *dobit* dodaje na svaki element glavne dijagonale matrice *novoOpterećenje*.

U svakom redu i matrice *novoOpterećenje*, element sa maksimalnom vrednošću obeležen je sa max_i. Vrednost max_i označava maksimalno opterećenje uspostavljenog centra, u slučaju da je centar i zatvoren, a centar t_{in} uspostavljen. Budući da je potrebno minimizovati maksimalno opterećenje uspostavljenog centra, određuje se $max = \min_i\{max_i\}$. Neka je t_{out} centar koji odgovara redu matrice gde je max pronađen. Tada je t_{out} najbolji kandidat za zatvaranje ukoliko se uspostavlja t_{in}. Odgovarajuća promena vrednosti funkcije cilja jednaka je $max - trenutnaFunkcijaCilja$, gde je $trenutnaFunkcijaCilja$ vrednost funkcije cilja pre zamene lokacija.

4.4 Eksperimentalni rezultati

Testiranje je izvršeno na modifikovanim AP instancama srednjih i velikih dimenzija sa $50 \leq |J| \leq 400$ lokacija za potencijalne centre. Instance manjih dimenzija ($|J| < 50$) nisu izazovne za predloženi metod, tako da su one izostavljene iz testiranja. AP instance su zasnovane na podacima poštanske mreže Australije i prvi put su predstavljene u radu [41] iz 1998. godine. Koriste se kao standardne instance za testiranje hab lokacijskih problema. Ove instance, prilagođene problemu LTCFLP,

su predstavljene u [127] i na njima su definisane dve vrste zahteva. Zahtev klijenta $i \in J$ predstavlja broj pacijenata d_i na lokaciji i. Za instance gde je $n = 50, 100, 200$, uski (engl. tight - T) i široki (engl. loose - L) kapaciteti korišćeni su kao uski i široki zahtevi pacijenata. Budući da AP instance sa $n = 60, 70, 80, 90, 110, 120, 130$ potencijalnih centara nemaju date kapacitete, za svaku instancu su kreirana dva tipa zahteva d_i, zasnovana na ulaznom i izlaznom protoku kroz centar i ukupnom protoku kroz mrežu:

$$d_i = W(D_i/(O_i + D_i))(1 \pm r).$$

Vrednost W predstavlja ukupan protok kroz mrežu, O_i i D_i predstavljaju izlazni i ulazni protok kroz centar i, dok vrednost r predstavlja slučajan broj iz intervala $(0, 0.5)$. Testovi su izvršeni na računaru koji ima Intel Core i7-860 2.8 GHz procesor i 8GB RAM memorije. Predloženi VNS algoritam implementiran je u C# programskom jeziku, na .NET platformi.

Podešavanje parametara

Izvršeno je testiranje algoritma za različite verzije parametra k_{rvns_max}, sa ciljem odabira vrednosti parametra sa kojim će algoritam imati najbolje performanse. U literaturi, vrednost k_{rvns_max} parametra je uglavnom mali pozitivni ceo broj između 2 i 5 [68, 62, 61]. Za potrebe podešavanja vrednosti parametra k_{rvns_max}, odabrano je 15 reprezentativnih instanci problema srednjih dimenzija. Razmatrane su sledeće vrednosti $k_{rvns\_max} = 2, 3, 4, 5$. Za svaku instancu i za svaku vrednost parametra k_{rvns_max} izvršeno je 20 testova sa 20 različitih vrednosti semena za generisanje pseudo-slučajnih brojeva i zabeleženi su sledeći rezultati: najbolja dobijena vrednost funkcije cilja (res), vreme potrebno da se dobije ova vrednost u sekundama (t_{res}) i ukupno vreme izvršavanja u sekundama (t_{uk}) kada je ova vrednost dobijena. Dobijeni rezultati prikazani su u tabeli 4.2. Iz prosečne vrednosti date u poslednjem redu tabele 4.2 može se videti da je vrednost parametra $k_{rvns\_max} = 2$ najpogodnija opcija za predloženi VNS metod. U proseku, ova vrednost parametra davala je najbolje vrednosti funkcije cilja za najkraće procesorsko vreme.

Poređenje dobijenih rezultata

Rezultati predstavljenog VNS algoritma upoređeni su sa rezultatima hibridnog algoritma EA-VNS iz [90] i evolutivnog algoritma iz [127]. Sve tri metode pokretane su na istoj platformi u eksperimentima (procesor Intel Core i7-860 2.8 GHz i 8GB

Tabela 4.2: Analiza k_{rvns_max} parametra

	$k_{rvns\_max}=2$			$k_{rvns\_max}=3$			$k_{rvns\_max}=4$			$k_{rvns\_max}=5$		
instanca	res	$t_{res}[s]$	$t_{uk}[s]$	res	$t_{res}[s]$	$t_{uk}[s]$	res	$t_{res}[s]$	$t_{uk}[s]$	res	$t_{res}[s]$	$t_{uk}[s]$
70L-50	5473.93	0.33	9.90	5473.93	0.41	10.73	5473.93	0.43	11.16	5473.93	0.53	11.39
70T-50	3248.77	0.23	9.75	3248.77	0.39	11.63	3248.77	0.44	10.40	3248.77	0.56	10.89
100L-90	3276.88	0.55	15.62	3276.88	0.83	16.53	3276.88	1.02	15.45	3276.88	1.34	15.45
110L-70	5801.22	1.66	40.17	5801.22	4.65	40.00	5801.22	11.85	44.15	5801.22	4.21	37.95
110T-40	5377.28	10.97	40.35	5334.71	26.35	55.95	5334.71	33.86	63.30	5334.71	19.99	49.53
110T-80	3214.58	2.49	42.52	3214.58	8.45	53.43	3214.58	1.90	44.29	3214.58	4.83	47.69
120T-80	3338.25	1.72	52.31	3338.25	1.78	51.23	3338.25	1.94	50.82	3338.25	4.00	54.97
130L-60	6988.03	5.57	68.06	6988.03	18.72	94.49	6988.03	33.02	95.22	6988.03	16.97	95.05
130T-60	4190.63	15.34	77.23	4190.63	26.43	96.95	4171.17	36.04	106.28	4203.44	28.47	101.79
130T-80	3363.43	7.07	77.11	3363.43	13.32	90.11	3363.43	12.23	90.65	3363.43	37.42	121.34
200L-40	9805.42	176.29	260.93	9907.32	102.09	197.47	9887.73	106.23	203.33	9922.40	228.05	322.23
200L-140	3480.51	32.27	265.50	3473.34	391.75	635.13	3480.51	77.92	304.37	3480.51	54.71	269.32
200L-150	3274.92	16.74	210.18	3274.92	16.79	237.88	3274.92	14.90	220.90	3274.92	18.07	330.90
200T-70	2632.47	186.81	381.66	2667.51	80.26	314.94	2655.62	189.20	415.13	2648.49	703.71	948.56
200T-80	2461.13	166.39	371.68	2438.82	1065.77	1369.49	2451.95	481.44	750.01	2448.99	404.87	673.30
prosek	4395.16	41.63	128.20	4399.49	117.20	218.40	4397.45	66.83	161.70	4401.24	101.85	206.02

RAM memorije). Na svakoj instanci predstavljeni algoritam je izvršavan 20 puta sa 20 različitih vrednosti semena za generisanje pseudo-slučajnih brojeva. Detaljni rezultati VNS algoritma, poznata optimalna rešenja i poređenje sa EA-VNS i EA dati su u narednoj podsekciji.

U ovoj sekciji prikazan je kratak pregled poređenja VNS, EA-VNS i EA metoda. Pregled eksperimentalnih rezultata prikazan je u tabeli 4.3. Razmatrane su 24 grupe instanci, 210 instanci ukupno. Svaka grupa sadrži instance iste dimenzije problema $|J|$ sa istim tipom zahteva (L ili T), ali različite vrednosti vrednosti parametra K. U prvoj koloni tabele 4.3 data je dimenzija problema i tip zahteva za posmatranu grupu instanci, dok je u drugoj koloni dat broj instanci u toj grupi. Za svaku od tri metaheuristike za rešavanja LTCFLP prikazano je:

- prosečno ukupno procesorsko vreme za dobijanje najboljeg rešenja u sekundama - prosek(t_{uk}), koje je izračunato na posmatranoj grupi LTCFLP instanci sa istom dimenzijom problema i tipom zahteva,
- procenat instanci iz posmatrane grupe za koju je odgovarajući metod dostigao optimalna/najbolja poznata rešenja - nr.

Najbolji rezultati u kolonama koje sadrže vrednost nr su potamnjeni.

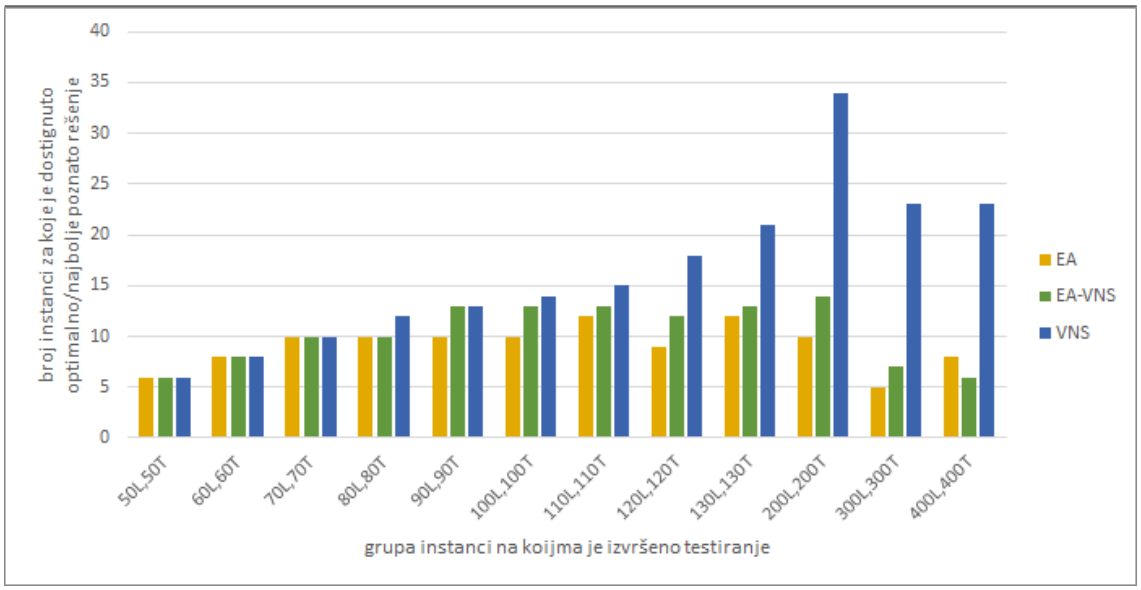
Sumirani rezultati pokazuju superiornost predstavljene VNS metode u odnosu na EA-VNS i EA u smislu kvaliteta rešenja. Za instance sa $50 \leq |J| \leq 80$ potencijalnih centara, predstavljeni VNS dostigao je optimalna ili najbolja poznata rešenja na

Tabela 4.3: Pregled poređenja rezultata VNS, EA-VNS i EA metoda

		VNS		EA-VNS		EA	
instanca	#	prosek(t_{uk})[s]	nr[%]	prosek(t_{uk})[s]	nr[%]	prosek(t_{uk})[s]	nr[%]
50L	3	2.99	**100.00**	0.23	**100.00**	154.21	**100.00**
50T	3	2.48	**100.00**	0.22	**100.00**	154.34	**100.00**
60L	4	4.67	**100.00**	52.71	**100.00**	203.35	**100.00**
60T	4	4.93	**100.00**	48.16	**100.00**	193.01	**100.00**
70L	5	8.26	**100.00**	99.08	**100.00**	221.50	**100.00**
70T	5	8.09	**100.00**	112.89	**100.00**	224.64	**100.00**
80L	6	13.45	**100.00**	192.88	83.33	251.27	83.33
80T	6	12.41	**100.00**	186.73	83.33	254.77	83.33
90L	7	19.48	**100.00**	282.13	85.71	292.68	85.71
90T	7	20.21	85.71	280.31	**100.00**	301.56	57.14
100L	8	24.76	**87.50**	456.61	75.00	305.63	75.00
100T	8	26.00	**87.50**	460.26	**87.50**	328.63	50.00
110L	9	37.02	**100.00**	774.84	66.67	229.53	66.67
110T	9	43.45	66.67	846.86	**77.78**	220.39	66.67
120L	10	57.85	**90.00**	1272.69	40.00	247.10	40.00
120T	10	50.64	**90.00**	1259.61	80.00	239.56	50.00
130L	11	54.31	**100.00**	1621.25	72.73	287.22	63.64
130T	11	51.88	**90.91**	1596.08	45.45	282.47	45.45
200L	18	343.42	**94.44**	3264.28	33.33	473.72	22.22
200T	18	228.86	**94.44**	3077.43	44.44	459.39	33.33
300L	12	990.79	**91.67**	3431.83	25.00	609.15	16.67
300T	12	1042.64	**100.00**	3415.12	33.33	617.86	25.00
400L	12	3258.94	**91.67**	3569.59	25.00	913.09	33.33
400T	12	2740.95	**100.00**	3520.40	25.00	894.63	33.33
prosek		377.02	**93.81**	1242.59	59.52	348.32	52.38

svim instancama, dok su oba EA-VNS i EA pokazala određena odstupanja od najboljih poznatih rešenja kod instanci sa $|J| = 80$ potencijalnih centara. Kod instanci sa $|J| = 90$ potencijalnih centara, VNS i EA-VNS nisu dostigli najbolje poznato rešenje za samo jednu instancu, dok EA nije dostigao najbolje poznato rešenje za četiri instance. Za instance većih dimenzija $100 \leq |J| \leq 130$ predstavljeni VNS nije dostigao najbolja poznata rešenja za osam od ukupno 76 instanci. Ipak, dobijeni rezultati su i dalje bolji u poređenju sa rezultatima koji su dobijeni EA-VNS i EA metodama. Jedina dva izuzetka su dve grupe instanci (90T i 110T), gde je EA-VNS dobio bolja rešenja od predstavljenog VNS algoritma. Prednost predstavljene VNS metode najočiglednija je kod testova na grupama instanci velikih dimenzija. Za instance koje imaju $|J| = 200$, $|J| = 300$ i $|J| = 400$ potencijalnih centara procenat dostignutih najboljih rešenja VNS metode značajno je veći u poređenju sa odgovarajućim procentima EA-VNS i EA metoda. Kada se posmatra svih 210 instanci predstavljeni VNS, EA-VNS i GA dostigli su optimalna ili najbolja poznata rešenja za 93.81%, 59.52% i 52.38% instanci redom. Grafički prikaz rezultata prikazan je na slici 4.3.

Iz vrednosti u koloni prosek(t_{uk}) tabele 4.3 može se videti da je EA-VNS algoritmu u proseku potrebno najduže procesorsko vreme za rešavanje problema, dok



Slika 4.3: Grafički prikaz poređenja broja instanci za koje je dostignuto optimalno/najbolje poznato rešenje VNS, EA-VNS i EA metodama

je EA algoritmu potrebno najkraće procesorsko vreme, koje je malo bolje u odnosu na vreme predstavljenog VNS algoritma. Ipak, korektno poređenje ne može biti ostvareno kada se poredi procesorsko vreme algoritama koji su dobili rešenja različitog kvaliteta. Da bi se izvršilo korektno poređenje vremena izvršavanja algoritma, upoređena su prosečna procesorska vremena VNS algoritma sa prosečnim procesorskim vremenima EA-VNS algoritma, ali samo na onim instancama za koje su oba algoritma dobila isto rešenje. Na isti način upoređeno je i prosečno procesorsko vreme VNS i EA algoritma. Pregled poređenja vremena prikazan je u tabeli 4.4. U proseku, predstavljeni VNS ima kraće procesorsko vreme izvršavanja u poređenju sa obe EA-VNS i EA metode, na skupu instanci na kojima su dobijena ista rešenja.

Tabela 4.4: Poređenje prosečnih procesorskih vremena

poređenje	#	VNS prosek(t_{uk})[s]	EA-VNS/EA prosek(t_{uk})[s]
VNS i EA-VNS	117	**125.23**	981.04
VNS i EA	104	**139.69**	354.85

Detaljni rezultati

U tabelama 4.5-4.10 prikazana su detaljna poređenja rezultata dobijenih predstavljenim VNS algoritmom, EA-VNS metodom iz [90] i EA metodom iz [127] za rešavanje LTCFLP problema. Navedene metode su testirane na instancama sa $50 \leq |J| \leq 400$ potencijalnih lokacija za izgradnju centara predstavljenim u [127]. Sve tri metode testirane su na istoj platformi. Rezultati i poređenja u tabelama 4.5-4.10 predstavljeni su na sledeći način. Za svaku instancu prvo je dato njeno ime, koje sadrži broj potencijalnih lokacija za izgradnju centra, slovo koje označava o kojoj vrsti zahteva se radi i maksimalan broj centara koji može biti uspostavljen. U tabeli 4.5 je pored imena instance dato optimalno rešenje za tu instancu ukoliko je ono poznato. Optimalna rešenja su dobijena rešavanjem predstavljenog LTCFLP modela CPLEX rešavačem. Na mestima na kojima se pored rešenja nalazi znak *, CPLEX rešavač je našao rešenje, ali nije dokazao njegovu optimalnost. Za predstavljeni VNS algoritam i EA-VNS iz [90] prikazani su:

- najbolje rešenje dobijeno odgovarajućom metodom, označeno sa res,
- procesorsko vreme potrebno da se dobije najbolje rešenje - t_{res} (u sekundama),
- ukupno procesorsko vreme izvršavanja algoritma kada je dobijeno rešenje res - t_{uk} (u sekundama),
- prosečno odstupanje od najboljeg rešenja - $agap$ (u procentima) i
- standardna devijacija od najboljeg rešenja - σ (u procentima).

S obzirom da je hibridna metoda EA-VNS iz [90] nadmašila EA metodu iz [127] izostavljene su detaljne informacije o rezultatima EA iz tabela 4.5-4.10. Prikazani su najbolje rešenje res i odgovarajuće ukupno vreme izvršavanja t_{exec} EA metode. Najbolja dobijena rešenja su istaknuta.

Grafički prikazi poređenja rezultata i ukupnog vremena izvršavanja EA, EA-VNS i VNS metoda dati su na slikama 4.4 i 4.5.

Tabela 4.5: Rezultati i poređenja na instancama sa $50 \leq |J| \leq 80$ potencijalnih lokacija za izgradnju centara

		VNS					EA-VNS					EA	
inst.	opt.	*res*	$t_{res}[s]$	$t_{uk}[s]$	*agap*[%]	σ[%]	*res*	$t_{res}[s]$	$t_{uk}[s]$	*agap*[%]	σ[%]	*res*	$t_{uk}[s]$
50L													
50L-20	6084.88	**6084.88**	0.07	2.42	0.00	0.02	**6084.88**	0.18	0.28	1.66	1.90	**6084.88**	145.33
50L-30	4465.10	**4465.10**	0.09	3.39	0.00	0.00	**4465.10**	0.14	0.25	5.14	4.25	**4465.10**	156.60
50L-40	3495.40	**3495.40**	0.12	3.15	0.00	0.00	**3495.40**	0.04	0.15	0.00	0.00	**3495.40**	160.71
50T													
50T-20	2292.39	**2292.39**	0.06	2.24	0.00	0.00	**2292.39**	0.17	0.27	5.93	3.16	**2292.39**	144.01
50T-30	1762.63	**1762.63**	0.09	2.96	0.00	0.00	**1762.63**	0.10	0.27	5.87	7.05	**1762.63**	154.99
50T-40	1596.90	**1596.90**	0.12	2.24	0.00	0.00	**1596.90**	0.01	0.13	0.00	0.00	**1596.90**	164.03
60L													
60L-20	9096.21*	**9096.21**	0.13	3.20	0.30	0.63	**9096.21**	17.05	60.87	0.12	0.41	**9096.21**	184.80
60L-30	6635.68*	**6635.68**	0.12	5.92	0.00	0.00	**6635.68**	4.81	87.06	0.00	0.00	**6635.68**	213.86
60L-40	5533.81	**5533.81**	0.38	5.57	0.20	0.61	**5533.81**	7.34	45.55	0.18	0.78	**5533.81**	228.31
60L-50	4628.48	**4628.48**	0.19	3.97	0.00	0.00	**4628.48**	0.08	17.35	0.19	0.83	**4628.48**	186.44
60T													
60T-20	5564.10*	**5564.09**	0.08	3.06	0.45	0.66	**5564.09**	22.43	61.64	0.32	0.10	**5564.09**	190.54
60T-30	4074.37*	**4074.37**	0.11	5.01	0.13	0.38	**4074.37**	5.10	77.38	0.06	0.27	**4074.37**	197.33
60T-40	3366.66	**3366.66**	0.22	5.73	0.69	1.06	**3366.66**	8.46	41.28	0.29	0.90	**3366.66**	193.21
60T-50	2757.52	**2757.52**	0.19	5.93	0.00	0.00	**2757.52**	0.10	12.34	0.16	0.48	**2757.52**	190.94
70L													
70L-20	-	**10542.10**	0.31	4.17	0.44	0.65	**10542.10**	33.50	96.82	0.67	0.60	**10542.10**	215.05
70L-30	7893.37*	**7893.37**	0.18	7.52	0.54	1.08	**7893.37**	22.72	157.37	0.19	0.83	**7893.37**	222.44
70L-40	6322.83*	**6322.83**	0.25	11.15	0.35	1.06	**6322.83**	4.72	125.66	0.53	1.26	**6322.83**	220.21
70L-50	5473.93	**5473.93**	0.33	9.90	0.22	0.52	**5473.93**	5.13	94.82	0.59	1.11	**5473.93**	240.65
70L-60	4712.01	**4712.01**	0.26	8.57	0.00	0.00	**4712.01**	0.08	20.75	0.00	0.00	**4712.01**	209.17
70T													
70T-20	-	**6232.06**	0.12	4.21	0.80	0.92	**6232.06**	20.70	202.59	0.53	2.68	**6232.06**	202.59
70T-30	4418.13*	**4418.13**	0.31	8.04	0.16	0.48	**4418.13**	44.98	154.11	0.24	0.57	**4418.13**	238.54
70T-40	3699.20*	**3699.20**	0.18	9.74	0.00	0.00	**3699.20**	14.66	143.97	0.00	0.00	**3699.20**	221.55
70T-50	3248.77	**3248.77**	0.23	9.75	0.00	0.00	**3248.77**	1.50	43.94	0.20	0.67	**3248.77**	240.24
70T-60	2833.78	**2833.78**	0.26	8.71	0.00	0.00	**2833.78**	0.53	19.82	0.09	0.32	**2833.78**	220.29
80L													
80L-20	-	**11722.78**	5.18	10.46	0.69	0.65	11810.81	57.31	249.44	2.94	0.68	11810.81	249.44
80L-30	8579.03*	**8579.03**	0.29	9.70	0.61	0.65	**8579.03**	70.26	242.08	0.71	0.83	**8579.03**	238.29
80L-40	6669.92*	**6669.92**	0.77	14.99	0.01	0.01	**6669.92**	45.37	306.11	0.01	0.01	**6669.92**	255.46
80L-50	5842.07*	**5842.07**	0.25	17.58	0.00	0.00	**5842.07**	1.08	215.12	0.35	1.13	**5842.07**	243.53
80L-60	5225.64	**5225.64**	0.35	17.64	0.00	0.00	**5225.64**	0.96	113.95	0.25	0.80	**5225.64**	268.16
80L-70	4521.59	**4521.59**	0.36	10.35	0.00	0.00	**4521.59**	0.22	30.57	0.53	0.71	**4521.59**	252.72
80T													
80T-20	-	**7178.49**	1.90	7.32	1.78	1.31	7197.36	54.72	246.31	2.94	0.91	7197.36	246.31
80T-30	-	**5089.14**	1.02	10.65	0.83	0.94	**5089.14**	94.59	255.35	0.72	0.61	**5089.14**	241.45
80T-40	-	**4189.18**	0.32	15.45	0.19	0.57	**4189.18**	38.20	258.14	1.76	0.70	**4189.18**	258.14
80T-50	3563.48	**3563.48**	0.27	16.36	0.00	0.00	**3563.48**	14.28	200.16	0.11	0.26	**3563.48**	259.69
80T-60	3116.19	**3116.19**	0.30	15.70	0.00	0.00	**3116.19**	14.88	137.14	1.38	1.88	**3116.19**	267.21
80T-70	-	**2893.75**	0.35	8.95	0.00	0.00	**2893.75**	0.18	23.28	0.01	0.04	**2893.75**	255.82

Tabela 4.6: Rezultati i poređenja na instancama sa $90 \leq |J| \leq 110$ potencijalnih lokacija za izgradnju centara

	VNS					EA-VNS					EA	
inst.	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{uk}[s]$
90L												
90L-20	**12870.17**	9.37	15.64	1.80	0.58	13008.59	1.69	216.84	0.87	0.36	13008.59	266.89
90L-30	**9233.42**	10.10	21.43	2.22	1.36	**9233.42**	2.29	388.18	1.93	1.24	**9233.42**	269.46
90L-40	**7268.31**	7.45	27.21	2.14	1.43	**7268.31**	2.20	563.36	2.42	1.30	**7268.31**	327.04
90L-50	**6226.12**	0.46	21.45	0.01	0.06	**6226.12**	2.31	369.90	0.00	0.00	**6226.12**	299.65
90L-60	**5665.87**	0.41	19.31	0.00	0.00	**5665.87**	0.79	252.74	0.00	0.00	**5665.87**	291.72
90L-70	**5123.41**	0.45	15.46	0.00	0.00	**5123.41**	3.09	155.44	0.00	0.00	**5123.41**	294.04
90L-80	**4251.72**	0.44	15.85	0.00	0.00	**4251.72**	0.69	28.46	0.18	0.78	**4251.72**	299.96
90T												
90T-20	**8147.54**	0.49	6.69	0.64	0.66	**8147.54**	1.77	211.80	1.04	0.35	8161.16	257.26
90T-30	**5716.11**	1.92	13.34	0.32	0.24	**5716.11**	1.61	385.48	0.41	0.31	5745.06	316.79
90T-40	4422.74	26.02	45.98	0.94	0.30	**4414.21**	2.02	522.72	0.83	0.60	4467.88	303.35
90T-50	**3912.78**	0.49	23.38	0.07	0.32	**3912.78**	2.49	420.27	0.07	0.32	**3912.78**	313.58
90T-60	**3328.01**	0.51	21.08	0.08	0.23	**3328.01**	1.34	296.78	0.11	0.27	**3328.01**	307.16
90T-70	**2988.33**	0.44	18.94	0.03	0.11	**2988.33**	0.44	107.28	0.23	0.25	**2988.33**	303.90
90T-80	**2596.51**	0.44	12.06	0.00	0.00	**2596.51**	0.12	17.82	0.00	0.00	**2596.51**	308.90
100L												
100L-20	9594.24	1.80	9.42	0.58	0.53	**9539.05**	1.76	299.27	1.21	0.55	**9539.05**	274.72
100L-30	**6704.91**	7.02	20.92	0.70	0.47	6715.86	2.29	594.01	2.17	1.47	6847.77	277.87
100L-40	**5329.11**	6.80	30.26	2.01	1.41	5383.13	2.82	798.82	0.71	0.93	5383.14	313.69
100L-50	**4668.18**	1.58	33.16	0.00	0.00	**4668.18**	2.39	867.80	2.33	5.86	**4668.18**	295.38
100L-60	**4006.54**	0.51	37.72	0.00	0.00	**4006.54**	2.12	594.29	0.08	0.13	**4006.54**	297.53
100L-70	**3588.93**	0.52	27.78	0.00	0.00	**3588.93**	2.21	328.04	1.61	1.61	**3588.93**	316.85
100L-80	**3276.88**	0.50	23.24	0.00	0.00	**3276.88**	2.14	148.90	0.00	0.00	**3276.88**	331.05
100L-90	**3276.88**	0.55	15.62	0.00	0.00	**3276.88**	0.38	21.76	0.00	0.00	**3276.88**	337.92
100T												
100T-20	4172.59	14.18	21.56	2.87	0.99	**4129.54**	2.82	297.80	5.55	4.72	4203.88	312.56
100T-30	**2960.98**	2.34	18.20	0.97	0.71	**2960.98**	2.91	604.71	1.38	0.49	2986.34	309.74
100T-40	**2439.35**	22.58	48.47	0.39	0.24	2449.07	3.72	803.84	1.49	5.84	2464.96	310.28
100T-50	**2034.18**	0.80	27.44	0.38	0.65	**2034.18**	2.44	721.58	0.40	0.70	**2034.18**	329.65
100T-60	**1762.49**	0.53	29.55	0.00	0.00	**1762.49**	1.70	720.11	0.06	0.15	1769.98	339.27
100T-70	**1529.79**	0.57	29.76	0.07	0.31	**1529.79**	3.31	380.83	0.07	0.31	**1529.79**	356.27
100T-80	**1490.35**	0.49	22.84	0.00	0.00	**1490.35**	2.33	123.11	0.00	0.00	**1490.35**	325.57
100T-90	**1490.35**	0.53	10.22	0.00	0.00	**1490.35**	0.33	30.06	0.00	0.00	**1490.35**	345.66
110L												
110L-20	**15651.76**	10.50	18.88	1.64	0.89	**15651.76**	216.37	417.55	1.68	0.63	15713.08	213.52
110L-30	**10971.34**	3.41	21.24	1.03	0.69	**10971.34**	485.93	891.28	1.09	0.49	11075.01	191.81
110L-40	**8712.77**	13.25	43.76	0.56	0.44	8766.34	412.63	1284.54	0.16	0.29	**8712.77**	206.85
110L-50	**7160.93**	39.59	78.60	1.67	1.58	7211.10	704.73	1534.77	2.18	1.73	**7160.93**	226.99
110L-60	**6390.98**	4.01	51.18	0.21	0.21	**6390.98**	272.78	1074.84	0.49	1.43	**6390.98**	291.59
110L-70	**5801.22**	1.66	40.17	0.51	0.47	5824.43	183.50	908.99	0.32	0.48	5824.43	245.38
110L-80	**5365.04**	1.57	32.68	0.35	1.41	**5365.04**	91.17	584.97	1.13	2.29	**5365.04**	237.28
110L-90	**4769.66**	0.69	30.24	0.00	0.00	**4769.66**	4.64	238.27	1.25	1.89	**4769.66**	228.90
110L-100	**4456.20**	0.69	16.42	0.00	0.00	**4456.20**	0.43	38.34	0.00	0.00	**4456.20**	223.46
110T												
110T-20	9838.45	3.48	12.06	0.57	0.47	**9764.87**	192.55	421.07	1.42	0.61	9894.52	175.35
110T-30	6762.77	27.03	43.63	1.03	0.55	**6740.88**	374.10	862.29	1.25	0.70	6794.54	199.86
110T-40	**5377.28**	10.97	40.35	0.68	0.64	5384.97	426.69	1312.48	0.18	0.23	5425.05	187.02
110T-50	4482.67	70.65	112.37	3.81	1.51	4482.67	570.83	1675.04	3.31	1.91	**4455.42**	267.93
110T-60	**3937.45**	1.32	46.77	0.07	0.14	**3937.45**	202.33	1493.19	0.32	0.62	**3937.45**	211.17
110T-70	**3559.13**	0.59	43.50	0.00	0.00	**3559.13**	35.38	841.95	0.20	0.41	**3559.13**	243.35
110T-80	**3214.58**	2.49	42.52	1.46	1.23	**3214.58**	104.08	745.19	1.21	1.44	**3214.58**	247.91
110T-90	**2964.33**	0.70	33.72	0.00	0.00	**2964.33**	13.74	235.34	1.76	1.64	**2964.33**	222.73
110T-100	**2771.58**	0.67	16.12	0.00	0.00	**2771.58**	0.64	35.20	0.00	0.00	**2771.58**	228.18

Tabela 4.7: Rezultati i poređenja na instancama sa $|J| = 120, 130$ potencijalnih lokacija za izgradnju centara

	VNS					EA-VNS					EA	
inst.	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{uk}[s]$
120L												
120L-20	**17101.64**	9.13	18.90	1.52	0.77	17331.13	303.84	541.37	0.59	0.38	17240.88	221.46
120L-30	**11827.28**	26.75	48.50	1.18	0.84	11883.45	558.59	1147.17	1.19	0.57	12018.28	224.63
120L-40	**9258.17**	95.94	131.01	1.88	1.78	9307.80	688.47	1738.63	2.44	1.07	9308.19	218.61
120L-50	**7833.68**	24.78	74.83	1.69	1.71	**7833.68**	958.83	2517.06	0.82	0.46	7894.12	229.82
120L-60	**6674.50**	17.84	81.96	0.98	1.09	6723.27	1206.68	2660.88	1.62	1.25	6723.27	246.43
120L-70	**6130.43**	3.05	66.40	0.04	0.10	**6130.43**	561.31	1984.53	0.23	0.12	6147.52	261.87
120L-80	5832.84	5.44	51.52	0.11	0.15	5832.84	179.39	1240.61	0.33	0.31	**5821.67**	256.27
120L-90	**5365.30**	1.48	47.36	0.00	0.00	5404.65	15.42	516.98	0.62	0.99	**5365.30**	263.82
120L-100	**4921.73**	1.05	36.33	0.00	0.00	**4921.73**	10.60	315.96	0.60	1.29	**4921.73**	296.48
120L-110	**4675.52**	0.79	21.73	0.00	0.00	**4675.52**	0.60	63.75	0.00	0.00	**4675.52**	251.57
120T												
120T-20	**10107.83**	17.89	27.48	1.20	0.51	10178.73	245.80	536.91	0.64	0.42	10225.30	196.58
120T-30	**7063.04**	29.96	50.92	1.83	0.78	7122.82	565.23	1165.59	0.92	0.62	7238.94	205.52
120T-40	**5585.55**	13.14	49.47	0.94	0.63	**5585.55**	498.51	1838.32	1.28	0.46	**5585.55**	241.64
120T-50	4744.05	29.78	75.91	1.50	0.85	**4644.50**	1217.97	2434.18	3.04	0.96	4744.06	236.44
120T-60	**3989.90**	14.45	80.51	0.32	0.67	**3989.90**	808.11	2463.05	0.70	1.43	4150.62	248.96
120T-70	**3598.07**	10.72	67.04	0.31	0.32	**3598.07**	606.26	2224.54	0.52	0.42	3612.12	259.91
120T-80	**3338.25**	1.72	52.31	0.31	0.54	**3338.25**	107.96	819.12	1.99	0.57	**3338.25**	250.81
120T-90	**3052.15**	1.14	44.29	0.00	0.00	**3052.15**	34.57	752.89	0.23	0.44	**3052.15**	250.47
120T-100	**2867.58**	0.76	37.35	0.00	0.00	**2867.58**	7.77	322.36	0.64	1.44	**2867.58**	250.53
120T-110	**2867.58**	0.79	21.11	0.00	0.00	**2867.58**	0.22	39.17	0.00	0.00	**2867.58**	254.70
130L												
130L-20	**18443.28**	9.34	20.16	1.00	0.70	18520.87	361.81	699.06	1.12	0.73	18520.87	241.87
130L-30	**12833.83**	11.34	35.24	0.80	0.67	12855.10	729.62	1514.08	0.79	0.56	12872.41	226.54
130L-40	**9822.86**	41.22	78.80	3.33	1.19	**9822.86**	733.99	2399.85	3.31	1.15	**9822.86**	275.59
130L-50	**8489.43**	4.64	56.41	0.47	0.65	**8489.43**	1242.29	3263.96	0.75	0.98	**8489.43**	248.21
130L-60	**6988.03**	5.57	68.06	2.39	1.94	**6988.03**	1472.27	3439.47	1.95	1.79	**6988.03**	290.42
130L-70	**6455.93**	29.64	92.95	0.31	0.32	6461.47	1072.23	2311.47	0.30	0.28	**6455.93**	324.16
130L-80	**6036.56**	16.86	81.16	1.96	1.63	**6036.56**	247.66	1727.41	2.14	1.59	**6036.56**	308.87
130L-90	**5437.40**	7.15	56.51	0.52	2.01	**5437.40**	96.73	1208.06	0.96	2.32	**5437.40**	326.50
130L-100	**4969.32**	6.79	49.00	2.36	1.26	**4969.32**	72.31	774.84	3.43	1.06	5085.25	342.85
130L-110	**4782.58**	1.11	32.82	0.00	0.00	**4782.58**	25.82	430.74	0.15	0.20	4792.66	294.26
130L-120	**4637.68**	0.86	26.27	0.00	0.00	**4637.68**	0.70	64.77	0.00	0.00	**4637.68**	280.18
130T												
130T-20	**10566.16**	11.56	23.04	1.81	1.07	10631.57	369.52	635.21	1.06	0.78	10751.62	206.00
130T-30	**7292.29**	23.97	47.72	2.52	1.54	7323.36	732.97	1376.44	3.30	1.36	7394.32	285.30
130T-40	**5789.18**	10.55	44.84	1.34	0.68	5790.40	1057.69	2190.97	1.20	0.88	5830.66	276.51
130T-50	4895.13	5.32	54.83	0.43	0.38	4895.13	1095.66	2742.47	0.34	0.36	**4888.29**	273.55
130T-60	**4190.63**	15.34	77.23	1.37	1.15	4240.40	1725.25	3452.31	0.96	1.13	4240.40	283.21
130T-70	**3649.06**	32.27	99.08	1.27	1.05	**3649.06**	874.38	2517.37	1.91	1.38	**3649.06**	321.70
130T-80	**3363.43**	7.07	77.11	1.14	1.29	3460.52	319.48	1963.68	0.09	0.05	3460.52	299.71
130T-90	**3169.26**	3.31	51.39	0.03	0.11	**3169.26**	483.00	1556.19	1.72	1.15	3210.29	305.95
130T-100	**2995.21**	1.69	48.72	0.00	0.00	**2995.21**	23.70	810.32	1.89	0.44	**2995.21**	286.27
130T-110	**2995.21**	0.82	30.83	0.00	0.00	**2995.21**	1.45	266.89	0.00	0.00	**2995.21**	276.66
130T-120	**2995.21**	0.81	15.95	0.00	0.00	**2995.21**	0.20	45.04	0.00	0.00	**2995.21**	292.35

Tabela 4.8: Rezultati i poređenja na instancama sa $|J| = 200$ potencijalnih lokacija za izgradnju centara

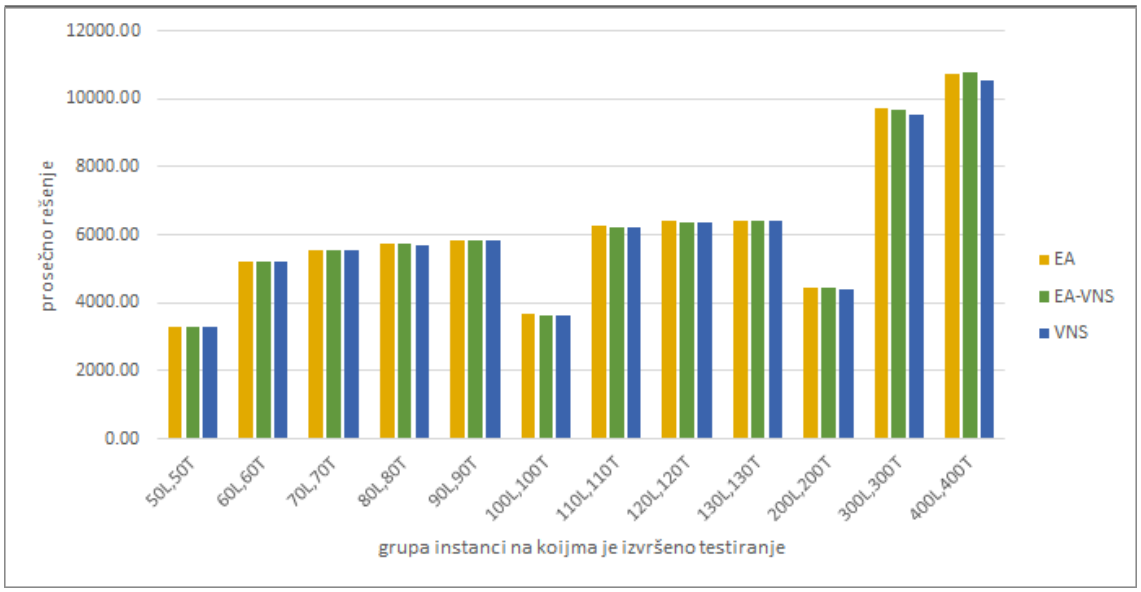
	VNS					EA-VNS					EA	
inst.	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{uk}[s]$
200L												
200L-20	18855.06	23.48	46.83	0.58	0.43	**18744.13**	1747.88	3602.96	1.52	0.52	18891.37	300.80
200L-30	**12742.08**	68.71	117.09	1.63	1.04	12854.04	2169.99	3608.99	2.61	1.15	13159.06	309.43
200L-40	**9805.42**	176.29	260.93	2.63	1.44	10065.10	1962.39	3617.31	2.36	1.54	10065.10	385.84
200L-50	**8114.31**	199.63	318.58	2.46	1.24	8319.80	1674.53	3622.82	1.24	1.00	8319.80	361.86
200L-60	**6973.09**	271.79	443.80	2.72	1.05	7193.96	1657.57	3624.20	0.87	0.56	7218.38	425.52
200L-70	**6130.96**	405.67	626.35	3.33	1.66	6280.24	1285.36	3631.33	3.47	1.57	6376.10	495.81
200L-80	**5429.41**	981.23	1209.04	3.20	1.07	5561.62	1369.71	3640.05	1.56	0.79	5636.58	477.02
200L-90	**4944.65**	379.21	649.61	2.50	1.29	5127.24	1157.29	3654.96	0.15	0.24	4995.45	484.04
200L-100	**4676.62**	97.31	364.02	1.00	0.77	4723.53	900.23	3641.57	0.62	0.72	4752.58	471.05
200L-110	**4351.33**	165.89	401.99	1.53	0.77	4411.15	639.88	3644.96	0.71	0.49	4448.17	585.27
200L-120	**4105.43**	238.64	468.80	0.58	0.23	4128.88	1519.23	3635.50	0.56	0.54	4141.33	504.67
200L-130	**3847.32**	68.93	280.92	2.16	1.08	3915.26	702.35	3623.98	1.82	1.90	3915.26	519.74
200L-140	**3480.51**	32.27	265.50	0.87	0.70	3556.89	1248.97	3618.09	1.25	2.31	3521.50	519.56
200L-150	**3274.92**	16.74	210.18	0.26	0.63	**3274.92**	889.21	3611.80	2.40	1.85	3317.12	615.68
200L-160	**3215.67**	2.98	175.60	0.00	0.00	**3215.67**	144.68	3606.49	0.06	0.27	**3215.67**	497.24
200L-170	**3215.67**	2.77	184.96	0.00	0.00	**3215.67**	43.16	3058.36	0.00	0.00	**3215.67**	515.95
200L-180	**3215.67**	2.15	104.54	0.00	0.00	**3215.67**	11.90	1135.59	0.00	0.00	**3215.67**	517.05
200L-190	**3215.67**	2.00	52.79	0.00	0.00	**3215.67**	0.72	178.01	0.00	0.00	**3215.67**	540.35
200T												
200T-20	**8110.74**	26.46	49.63	1.23	0.71	8145.22	1916.51	3603.06	1.05	0.57	8250.76	275.63
200T-30	**5607.96**	5.53	55.87	1.13	0.98	5633.25	1549.94	3607.11	1.73	0.76	5765.66	307.64
200T-40	**4290.64**	203.26	284.38	1.78	1.19	4353.27	1921.94	3616.39	1.72	0.91	4374.62	375.47
200T-50	**3561.34**	135.97	254.97	1.49	1.55	3640.60	1977.31	3618.33	2.58	1.42	3577.55	377.21
200T-60	**3024.92**	69.66	242.32	1.35	1.20	3056.09	1802.77	3624.18	2.00	1.62	3083.44	403.61
200T-70	**2632.47**	186.81	381.66	2.43	1.10	2701.66	1349.79	3625.18	1.61	1.47	2701.66	460.71
200T-80	**2461.13**	166.39	371.68	0.69	0.40	2478.24	977.97	3617.69	0.80	0.35	2478.24	467.46
200T-90	2281.66	31.82	268.54	0.47	0.34	2293.73	1383.37	3618.15	0.04	0.04	**2274.72**	411.30
200T-100	**2075.61**	326.40	535.45	2.86	1.86	2129.46	1580.01	3619.06	1.07	0.99	2107.16	463.97
200T-110	**1939.91**	278.34	487.01	0.97	0.70	1949.37	1866.80	3611.04	1.91	1.94	2017.33	520.77
200T-120	**1819.40**	23.14	191.71	0.26	0.38	**1819.40**	708.04	3615.14	0.90	0.90	1830.81	517.43
200T-130	**1590.28**	104.45	296.74	1.57	1.83	**1590.28**	746.53	3612.94	1.79	1.57	1593.91	499.34
200T-140	**1462.22**	20.42	182.89	0.02	0.06	**1462.22**	754.17	3612.27	1.48	3.66	1465.30	541.88
200T-150	**1445.45**	5.87	186.11	0.00	0.00	**1445.45**	285.62	3607.55	0.00	0.00	**1445.45**	550.73
200T-160	**1445.45**	3.50	118.41	0.00	0.00	**1445.45**	110.09	2724.07	0.00	0.00	**1445.45**	493.84
200T-170	**1445.45**	2.20	103.24	0.00	0.00	**1445.45**	61.50	1524.55	0.00	0.00	**1445.45**	529.52
200T-180	**1445.45**	2.04	68.95	0.00	0.00	**1445.45**	4.90	427.00	0.00	0.00	**1445.45**	522.44
200T-190	**1445.45**	1.98	39.89	0.00	0.00	**1445.45**	0.29	109.95	0.00	0.00	**1445.45**	550.14

Tabela 4.9: Rezultati i poređenja na instancama sa $|J| = 300$ potencijalnih lokacija za izgradnju centara

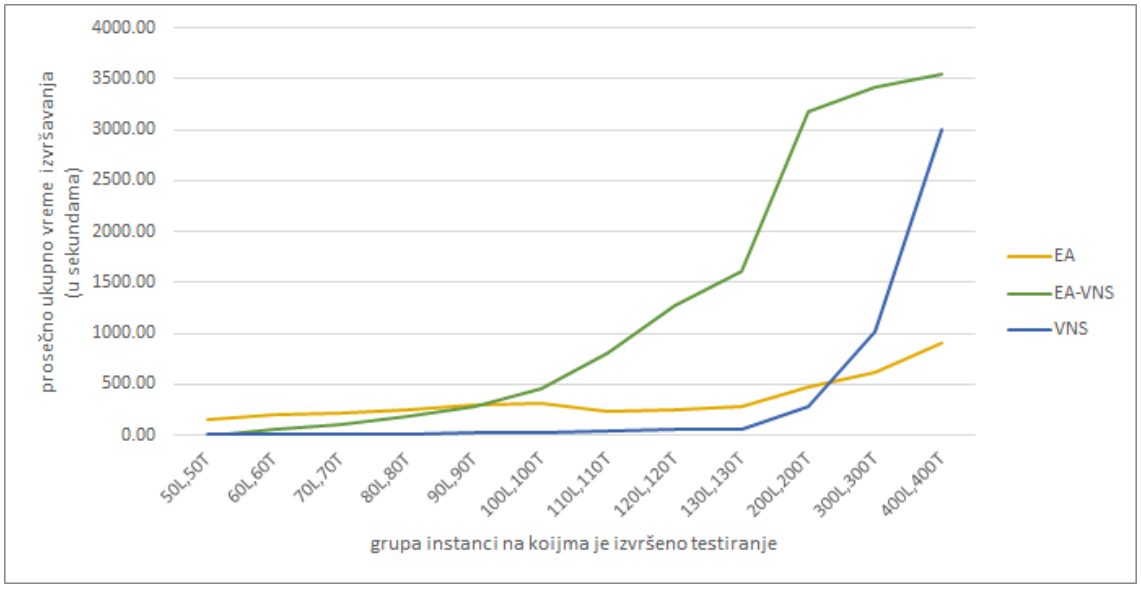
	VNS					EA-VNS					EA	
inst.	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{uk}[s]$
300L												
300L-20	40851.61	105.40	155.34	0.94	0.36	**40786.21**	1656.20	3602.85	1.39	1.11	41379.00	393.57
300L-40	**21114.97**	422.55	591.82	2.38	0.96	22012.91	2388.17	3619.79	1.22	0.99	21886.13	401.14
300L-50	**17217.41**	669.66	917.41	1.86	0.85	17562.00	2466.44	3608.70	1.29	0.92	17701.98	443.47
300L-70	**12854.12**	396.35	857.72	1.98	1.48	13331.12	1042.05	3675.17	1.03	0.73	13260.82	545.20
300L-100	**9622.75**	1011.34	1736.92	1.17	0.60	9800.51	1439.55	3679.16	0.52	0.41	9750.19	607.80
300L-120	**8173.93**	1136.67	2060.13	2.47	1.54	8262.75	2031.64	3672.79	2.11	1.38	8446.78	581.55
300L-160	**6452.64**	247.98	1517.04	1.39	1.06	6521.29	2390.14	3627.00	1.43	1.38	6652.39	686.22
300L-180	**6090.53**	192.62	1071.82	0.25	0.61	6104.17	1631.93	3657.80	2.01	1.18	6259.58	744.47
300L-200	**5586.59**	158.41	1167.14	0.57	0.48	5659.43	1630.18	3645.81	2.78	4.08	5659.43	718.96
300L-220	**5161.33**	46.51	985.01	0.10	0.17	5208.22	1065.41	3669.58	1.66	1.18	5253.47	729.70
300L-240	**4892.18**	37.13	567.23	0.07	0.30	**4892.18**	528.14	3615.29	0.89	1.14	**4892.18**	771.48
300L-280	**4892.18**	4.78	261.89	0.00	0.00	**4892.18**	31.78	1108.04	0.00	0.00	**4892.18**	686.26
300T												
300T-20	**24561.69**	66.97	116.51	0.88	0.59	24652.25	2776.99	3605.67	0.85	0.78	24907.82	394.60
300T-40	**12671.34**	133.80	291.66	1.52	0.83	12904.83	2129.22	3613.39	2.58	1.84	13165.14	410.77
300T-50	**10380.90**	521.98	768.38	1.94	1.03	10815.53	1890.18	3615.74	1.56	0.80	10776.75	477.41
300T-70	**7731.46**	1070.98	1529.82	1.43	1.46	7993.10	2545.62	3671.14	0.50	0.31	7959.29	527.68
300T-100	**5726.86**	1011.54	1779.68	1.48	1.08	5839.77	2544.65	3657.00	1.28	1.32	5813.80	625.92
300T-120	**4933.62**	2230.06	3040.70	1.56	1.01	5051.59	2530.35	3664.46	0.40	0.24	5051.59	623.33
300T-160	**3874.20**	617.14	1431.45	1.13	0.55	3940.73	2179.79	3663.82	1.93	1.36	3956.20	831.72
300T-180	**3605.98**	63.71	1161.57	0.04	0.16	**3605.98**	2167.07	3671.49	4.46	6.86	3632.14	709.22
300T-200	**3308.09**	64.30	1034.12	0.36	0.47	3344.25	2598.91	3635.10	1.56	1.45	3341.18	705.92
300T-220	**3058.65**	21.09	730.04	0.00	0.00	**3058.65**	1338.04	3640.83	3.11	4.39	**3058.65**	713.88
300T-240	**3058.65**	6.67	462.73	0.00	0.00	**3058.65**	305.97	3624.04	0.00	0.00	**3058.65**	698.56
300T-280	**3058.65**	4.64	165.05	0.00	0.00	**3058.65**	4.81	918.72	0.00	0.00	**3058.65**	695.35

Tabela 4.10: Rezultati i poređenja na instancama sa $|J| = 400$ potencijalnih lokacija za izgradnju centara

	VNS					EA-VNS					EA	
inst.	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{res}[s]$	$t_{uk}[s]$	$agap[\%]$	$\sigma[\%]$	res	$t_{uk}[s]$
400L												
400L-20	**53760.88**	192.47	279.54	0.74	0.45	54284.66	1933.06	3606.95	0.81	0.57	54535.73	580.95
400L-40	**27647.48**	774.37	1087.23	1.85	0.88	28688.97	1588.06	3625.38	0.98	0.63	28420.96	587.06
400L-80	**14683.62**	2350.60	3514.91	2.60	1.52	15279.42	2683.93	3739.58	2.25	1.48	15310.18	692.60
400L-100	**12096.20**	7396.70	9015.38	2.95	1.22	12661.21	2680.38	3719.27	1.15	0.72	12649.72	747.80
400L-140	**9181.28**	4351.46	7023.91	1.64	1.16	9427.24	2739.88	3758.96	2.95	2.62	9433.97	950.84
400L-180	**7397.88**	1815.06	4835.52	1.04	0.58	7682.46	2415.24	3858.38	9.30	9.75	7563.04	860.68
400L-240	**5963.05**	1137.50	3174.39	1.02	2.06	6124.11	1722.68	3739.02	9.60	5.27	6134.96	1099.42
400L-260	5678.94	1383.94	4126.58	1.97	4.27	5692.54	2403.79	3861.56	13.46	7.65	**5640.40**	1143.27
400L-280	**5428.72**	335.16	2750.14	1.21	3.95	5515.57	3397.88	3982.70	8.56	8.78	5473.89	1049.58
400L-300	**5280.97**	165.16	1666.71	0.22	0.94	**5280.97**	3368.33	3700.05	5.16	6.33	**5280.97**	1127.59
400L-340	**5280.97**	30.86	1002.60	0.00	0.00	**5280.97**	485.78	3649.66	0.00	0.00	**5280.97**	1013.67
400L-380	**4477.15**	10.27	630.42	0.00	0.00	**4477.15**	33.55	1593.59	0.00	0.00	**4477.15**	1103.65
400T												
400T-20	**32359.41**	68.30	158.85	0.48	0.37	32580.98	2143.32	3617.49	0.83	0.65	32678.51	614.87
400T-40	**16740.05**	363.13	667.94	1.49	0.81	17189.34	2005.82	3638.62	1.83	1.51	17099.28	573.29
400T-80	**9010.59**	828.47	1781.89	1.20	1.30	9277.12	2634.79	3661.29	2.37	2.36	9311.57	639.90
400T-100	**7438.94**	805.00	2355.16	0.95	0.59	7561.65	2688.19	3727.96	1.29	1.11	7500.55	777.36
400T-140	**5618.73**	9570.90	11961.32	2.08	1.04	5776.76	2196.96	3698.04	1.58	1.73	5766.18	819.70
400T-180	**4495.97**	2402.14	4351.46	0.62	0.51	4584.74	2759.92	3765.09	7.67	6.04	4584.74	907.02
400T-240	**3615.77**	1285.13	3561.29	0.38	0.26	3645.94	2848.19	3722.82	14.91	11.69	3687.52	1047.10
400T-260	**3452.55**	188.87	2145.43	0.01	0.02	3467.86	2791.66	3735.23	4.52	4.20	3493.14	1072.06
400T-280	**3359.67**	99.91	2185.93	0.56	2.43	3734.03	1763.77	3668.95	2.12	1.23	**3359.68**	1042.13
400T-300	**3359.67**	172.38	1899.49	0.02	0.07	**3359.67**	1244.20	3666.32	6.69	5.25	**3359.68**	1093.37
400T-340	**3359.67**	29.65	1570.04	0.00	0.00	**3359.67**	253.39	3649.91	0.80	1.59	**3359.68**	1027.30
400T-380	**3359.67**	9.11	252.63	0.00	0.00	**3359.67**	9.09	1693.05	0.00	0.00	**3359.68**	1121.52



Slika 4.4: Grafički prikaz poređena rezultata dobijenih EA, EA-VNS i VNS metodama



Slika 4.5: Grafički prikaz poređena ukupnog vremena izvršavanja EA, EA-VNS i VNS metoda

Glava 5

Zaključak

U ovom radu predstavljena je metoda promenljivih okolina za rešavanje problema određivanja lokacija autobuskih terminala i problema uspostavljanja centra za produženu negu pacijenata.

BTLP je formulisan u [49] i sadrži karakteristike problema p-medijane [59] u smislu da je svaki klijent pridružen svom najbližem uspostavljenom centru sa dodatkom opadajuće funkcije koja se primenjuje na udaljenost između klijenta i uspostavljenog centra. Osim toga, BTLP sadrži i uslov dostupnog prostora oko centara, što je karakteristika problema pokrivanja korisnika [24]. Predstavljena je paralelizovana implementacija metode promenljivih okolina za rešavanje BTLP problema. Unapređena lokalna pretraga zasnovana na brzoj zameni okolina je implementirana i karakteristika dostupnog prostora oko centara je iskorišćena za smanjenje veličine okoline pretrage. Alatom za merenje performansi procesora identifikovani su najzahtevniji delovi neparalelizovanog algoritma i paralelizacija je primenjena samo na te delove algoritma. Testovi su izvršeni na instancama iz [49], kao i na novim predstavljenim instancama zasnovanim na TSPLIB biblioteci, uvedenim u cilju testiranja algoritma na većim dimenzijama problema. Rezultati predloženog paralelnog algoritma su upoređeni sa rezultatima neparalelizovane verzije algoritme i paralelizacija je dala značajno unapređenje vremena izvršavanja u odnosu na broj procesorskih jezgara korišćen u testovima. Dalje, predloženi PVNS algoritam je nadmašio postojeće rezultati iz literature u smislu poboljšanja vrednosti funkcije cilja za značajno kraće vreme izvršavanja. Analiza rezultata je pokazala da je predstavljeni PVNS algoritam veoma stabilan pri rešavanju instanci problema većih dimenzija.

LTCFLP je formulisan u radu [77] i osnovna motivacija za njegovu formulaciju je pronalazak lokacija za izgradnju centara za produženu negu starih lica u Južnoj

Koreji. LTCFLP spada u klasu *min-max* lokacijskih problema i odnosi se na minimizaciju maksimalnog opterećenja centra koji pruža produženu medicinsku zaštitu. Za rešavanje LTCFLP problema je predložena jedna varijanta metode promenljivih okolina. Početno rešenje je dobijeno redukovanom metodom promenljivih okolina. Osnovna prednost predloženog VNS algoritma je predstavljena metoda brze zamene okolina za LTCFLP problem. Strukture podataka korišćene za primenu ove metode su kompleksnije u odnosu na poznate slične strukture iz literature. Testovi su izvršeni na poznatim instancama predstavljenim u [127]. Eksperimentalni rezultati pokazali su da je primenjena strategija značajno smanjila vreme izvršavanja i dovela do boljih rešenja u smislu kvaliteta vrednosti funkcije cilja u odnosu na rezultate dobijene ranijim VNS implementacijama za rešavanje LTCFLP problema. Vreme izvršavanja predstavljenog algoritma značajno je kraće u poređenju sa vremenima EA-VNS i EA algoritma, u slučajevima kada su dobijena ista rešenja. Na osnovu prikazanih rezultata, može se zaključiti da je predstavljena VNS metoda pokazala superiornosti u odnosu na postojeće metode za rešavanje LTCFLP problema.

Osnovni naučni doprinosi ovog rada su:

- dizajniranje strategije za paralelizaciju niskog nivoa VNS metode primenjene za rešavanje BTLP problema,
- konstrukcija unapređene lokalne pretrage za rešavanje BTLP problema,
- kreiranje instanci velikih dimenzija za BTLP problem, zasnovanih na TSPLIB biblioteci instanci,
- primena predstavljenog PVNS algoritma za rešavanje BTLP problema i predstavljanje novih, poboljšanih rezultata u odnosu na postojeće rezultate iz literature,
- konstrukcija nove strukture podataka uz pomoć koje je smanjena vremenska složenost lokalne pretrage za rešavanje LTCFLP problema i
- primena predstavljenog VNS algoritma za rešavanje LTCFLP problema i predstavljanje novih, poboljšanih rezultata u odnosu na postojeće rezultate iz literature.

Pravci daljeg rada uključuju dalje unapređenje predloženih struktura podataka primenjenih za rešavanje BTLP i LTCFLP problema. Osim za BTLP i LTCFLP

predložene strukture podataka se mogu primeniti za rešavanje drugih diskretnih lokacijskih problema koji sadrže neke od karakteristika razmatranih problema. Cilj bi bio implementirati brzu zamenu okolina za rešavanje drugih problema po uzoru na implementaciju predloženu u ovom radu i tako smanjiti vremensku složenost lokalne pretrage. Osim kao deo VNS metaheuristike, ovaj postupak se može primeniti i u drugim metaheuristikama vođenim jednim rešenjem, koje u svojoj implementaciji sadrže neki vid lokalne pretrage. Takođe, pogodno je implementirati i hibrid efikasne lokalne pretrage sa nekom populacionom metaheuristikom u cilju intenzifikacije pretrage. Pravci daljeg istraživanja mogu da obuhvate i modeliranje robusnih i fazi varijanti BTLP i LTCFLP problema, kao i dizajn metoda za njihovo efikasno rešavanje.

Literatura

[1] U. Akinc i B. M. Khumawala. „An efficient branch and bound algorithm for the capacitated warehouse location problem". U: *Management Science* 23.6 (1977), pp. 585–594.

[2] E. Alba. „Parallel evolutionary algorithms can achieve super-linear performance". U: *Information Processing Letters* 82.1 (2002), pp. 7–13.

[3] E. Alba. *Parallel metaheuristics: a new class of algorithms*. Vol. 47. John Wiley & Sons, 2005.

[4] G. C. Armour i E. S. Buffa. „A heuristic algorithm and simulation approach to relative location of facilities". U: *Management Science* 9.2 (1963), pp. 294–309.

[5] C. Audet, V. Béchard i S. L. Digabel. „Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search". U: *Journal of Global Optimization* 41.2 (2008), pp. 299–318.

[6] C. Audet, J. Brimberg, P. Hansen, S. L. Digabel i N. Mladenovic. „Pooling Problem: Alternate Formulations and Solution Methods". U: *Manage. Sci.* 50.6 (June 2004), pp. 761–776.

[7] M. Avazbeigi. „An Overview of Complexity Theory". U: *Facility Location*. Springer, 2009, pp. 19–36.

[8] S. Babaie-Kafaki, R. Ghanbari i N. Mahdavi-Amiri. „An efficient and practically robust hybrid metaheuristic algorithm for solving fuzzy bus terminal location problems". U: *Asia-Pacific Journal of Operational Research* 29.02 (2012), pp. 1–25.

[9] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[10] O. Baron, O. Berman, D. Krass i Q. Wang. „The equitable location problem on the plane". U: *European Journal of Operational Research* 183.2 (2007), pp. 578–590.

[11] O. Baron, J. Milner i H. Naseraldin. „Facility location: A robust optimization approach". U: *Production and Operations Management* 20.5 (2011), pp. 772–785.

[12] R. S. Barr i B. L. Hickman. „Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinions". U: *ORSA Journal on Computing* 5.1 (1993), pp. 2–18.

[13] J. Beasley. „Obtaining test problems via Internet". English. U: *Journal of Global Optimization* 8.4 (1996), pp. 429–433.

[14] O. Berman i D. Krass. „The generalized maximal covering location problem". U: *Computers & Operations Research* 29.6 (2002). Location Analysis, pp. 563–581.

[15] C. Blum i A. Roli. „Metaheuristics in combinatorial optimization: Overview and conceptual comparison". U: *ACM computing surveys (CSUR)* 35.3 (2003), pp. 268–308.

[16] M. L. Brandeau i S. S. Chiu. „An Overview of Representative Problems in Location Research". U: *Management Science* 35.6 (1989), pp. 645–674.

[17] J. Brimberg, Z. Drezner, N. Mladenović i S. Salhi. „A new local search for continuous location problems". U: *European Journal of Operational Research* 232.2 (2014), pp. 256–265.

[18] J. Brimberg, P. Hansen, N. Mlandinovic i E. D. Taillard. „Improvement and Comparison of Heuristics for Solving the Uncapacitated Multisource Weber Problem". U: *Oper. Res.* 48.3 (May 2000), pp. 444–460.

[19] G. Bruno i A. Genovese. „A mathematical model for the optimization of the airport check-in service problem". U: *Electronic Notes in Discrete Mathematics* 36 (2010), pp. 703–710.

[20] R. Burstall, R. Leaver i J. Sussams. „Evaluation of Transport Costs for Alternative Factory Sites–A Case Study". U: *OR* (1962), pp. 345–354.

[21] M. S. Canbolat i M. von Massow. „Locating emergency facilities with random demand for risk minimization". U: *Expert Systems with Applications* 38.8 (2011), pp. 10099–10106.

[22] F. Carrabs, J.-F. Cordeau i G. Laporte. „Variable Neighborhood Search for the Pickup and Delivery Traveling Salesman Problem with LIFO Loading". U: *INFORMS J. on Computing* 19.4 (Oct. 2007), pp. 618–632.

[23] R. L. Church, D. M. Stoms i F. W. Davis. „Reserve selection as a maximal covering location problem". U: *Biological Conservation* 76.2 (1996), pp. 105–112.

[24] R. Church i C. R. Velle. „The maximal covering location problem". U: *Papers in Regional Science* 32.1 (1974), pp. 101–118.

[25] S. A. Cook. „The complexity of theorem-proving procedures". U: *Proceedings of the third annual ACM symposium on Theory of computing*. ACM. 1971, pp. 151–158.

[26] T. Crainic, M. Gendreau, P. Hansen i N. Mladenović. „Cooperative Parallel Variable Neighborhood Search for the p-Median". English. U: *Journal of Heuristics* 10.3 (2004), pp. 293–314.

[27] M. S. Daskin. „Location modeling in perspective". U: *Network and Discrete Location: Models, Algorithms, and Applications* (1995), pp. 383–400.

[28] M. Daskin. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, 2011.

[29] S. Davari, M. H. F. Zarandi i I. B. Turksen. „A greedy variable neighborhood search heuristic for the maximal covering location problem with fuzzy coverage radii". U: *Knowledge-Based Systems* 41 (2013), pp. 68–76.

[30] I. Davydov, Y. Kochetov i E. Carrizosa. „A Local Search Heuristic for the (r| p)-Centroid Problem in the Plane". U: *Computers & Operations Research* 52 (2014), pp. 334–340.

[31] O. Díaz-Parra, J. A. Ruiz-Vanoye, B. Bernábe Loranca, A. Fuentes-Penna i R. A. Barrera-Cámara. „A Survey of Transportation Problems". U: *Journal of Applied Mathematics* 2014 (2014).

[32] H. Ding, A. Lim, B. Rodrigues i Y. Zhu. „The over-constrained airport gate assignment problem". U: *Computers & Operations Research* 32.7 (2005), pp. 1867–1880.

[33] A. Djenić, M. Marić, Z. Stanimirović i P. Stanojević. „A variable neighbourhood search method for solving the long-term care facility location problem". U: *IMA Journal of Management Mathematics* 28.2 (2016), pp. 321–338.

[34] A. Djenić, N. Radojičić, M. Marić i M. Mladenović. „Parallel VNS for bus terminal location problem". U: *Applied Soft Computing* 42 (2016), pp. 448–458.

[35] M. Dorigo. „Optimization, learning and natural algorithms". PhD thesis. Politecnico di Milano, 1992.

[36] D. Drakulić, A. Takači i M. Marić. „Fuzzy Covering Location Problems with Different Aggregation Operators". U: *Filomat* 31.2 (2017), pp. 513–522.

[37] D. Drakulić, A. Takači i M. Marić. „New Model of Maximal Covering Location Problem with Fuzzy Conditions." U: *Computing & Informatics* 35.3 (2016).

[38] Z. Drezner i H. Hamacher. *Facility Location: Applications and Theory*. Springer-Verlag Berlin Heidelberg, 2002.

[39] D.-Z. Du i K.-I. Ko. *Theory of computational complexity*. Vol. 58. John Wiley & Sons, 2011.

[40] M. Efroymson i T. Ray. „A branch-bound algorithm for plant location". U: *Operations Research* 14.3 (1966), pp. 361–368.

[41] A. T. Ernst i M. Krishnamoorthy. „Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem". U: *European Journal of Operational Research* 104.1 (1998), pp. 100–112.

[42] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia i M. Goh. „Covering problems in facility location: A review". U: *Computers & Industrial Engineering* 62.1 (2012), pp. 368–407.

[43] T. A. Feo i M. G. Resende. „A probabilistic heuristic for a computationally difficult set covering problem". U: *Operations research letters* 8.2 (1989), pp. 67–71.

[44] K. Fleszar i K. S. Hindi. „An effective VNS for the capacitated p-median problem". U: *European Journal of Operational Research* 191.3 (2008), pp. 612–622.

[45] R. L. Francis i J. Goldstein. „Location theory: A selective bibliography". U: *Operations Research* 22.2 (1974), pp. 400–410.

[46] A. Freeman. *Pro .NET 4 Parallel Programming in C#*. Expert's Voice in .NET. Apress, 2010.

[47] R. D. Galvão i C. ReVelle. „A Lagrangean heuristic for the maximal covering location problem". U: *European Journal of Operational Research* 88.1 (1996), pp. 114–123.

[48] F. García-López, B. Melián-Batista, J. Moreno-Pérez i J. Moreno-Vega. „The Parallel Variable Neighborhood Search for the p-Median Problem". English. U: *Journal of Heuristics* 8.3 (2002), pp. 375–388.

[49] R. Ghanbari i N. Mahdavi-Amiri. „Solving bus terminal location problems using evolutionary algorithms". U: *Applied Soft Computing* 11.1 (2011), pp. 991–999.

[50] F. Glover. „Future paths for integer programming and links to artificial intelligence". U: *Computers & operations research* 13.5 (1986), pp. 533–549.

[51] F. Glover. „Heuristics for integer programming using surrogate constraints". U: *Decision sciences* 8.1 (1977), pp. 156–166.

[52] F. Glover. „Tabu search—part I". U: *ORSA Journal on computing* 1.3 (1989), pp. 190–206.

[53] J. B. Goldberg. „Operations Research Models for the Deployment of Emergency Services Vehicles". U: *EMS Management Journal* 1 (2004), pp. 20–39.

[54] A. Goldman. „Optimal center location in simple networks". U: *Transportation science* 5.2 (1971), pp. 212–221.

[55] S. Goldshtein, D. Zurbalev i I. Flatow. „Performance Measurement". English. U: *Pro .NET Performance*. Apress, 2012, pp. 7–59.

[56] J. Guan i G. Lin. „Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem". U: *European Journal of Operational Research* 248.3 (2016), pp. 899–909.

[57] S. L. Hakimi. „Optimum locations of switching centers and the absolute centers and medians of a graph". U: *Operations research* 12.3 (1964), pp. 450–459.

[58] R. W. Hamming. „Error Detecting and Error Correcting Codes". U: *Bell System Technical Journal* 29.2 (1950), pp. 147–160.

[59] P. Hansen i N. Mladenović. „Variable neighborhood search for the p-median". U: *Location Science* 5.4 (1997), pp. 207–226.

[60] P. Hansen, J. Brimberg, D. Urošević i N. Mladenović. „Primal-dual variable neighborhood search for the simple plant-location problem". U: *INFORMS Journal on Computing* 19.4 (2007), pp. 552–564.

[61] P. Hansen, J. Brimberg, D. Urošević i N. Mladenović. „Solving large p-median clustering problems by primal–dual variable neighborhood search". English. U: *Data Mining and Knowledge Discovery* 19.3 (2009), pp. 351–375.

[62] P. Hansen, J. Lazić i N. Mladenović. „Variable neighbourhood search for colour image quantization". U: *IMA Journal of Management Mathematics* 18.2 (2007), pp. 207–221.

[63] P. Hansen i N. Mladenović. „Developments of Variable Neighborhood Search". U: *Essays and Surveys in Metaheuristics*. Boston, MA: Springer US, 2002, pp. 415–439.

[64] P. Hansen i N. Mladenović. „J-Means: a new local search heuristic for minimum sum of squares clustering". U: *Pattern Recognition* 34.2 (2001), pp. 405–413.

[65] P. Hansen i N. Mladenović. „Variable neighborhood search: Principles and applications". U: *European Journal of Operational Research* 130.3 (2001), pp. 449–467.

[66] P. Hansen, N. Mladenović, J. Brimberg i J. A. M. Pérez. „Variable neighborhood search". U: *Handbook of metaheuristics*. Springer, 2010, pp. 61–86.

[67] P. Hansen, N. Mladenović i J. A. Moreno Pérez. „Variable neighbourhood search: methods and applications". U: *Annals of Operations Research* 175.1 (2010), pp. 367–407.

[68] P. Hansen, N. Mladenović i D. Perez-Britos. „Variable Neighborhood Decomposition Search". English. U: *Journal of Heuristics* 7.4 (2001), pp. 335–350.

[69] J. Hartmanis i R. E. Stearns. „On the computational complexity of algorithms". U: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306.

[70] K. L. Hoffman, M. Padberg i G. Rinaldi. „Traveling salesman problem". U: *Encyclopedia of operations research and management science*. Springer, 2013, pp. 1573–1578.

[71] S. Hosseini, A. Al Khaled i S. Vadlamani. „Hybrid imperialist competitive algorithm, variable neighborhood search, and simulated annealing for dynamic facility layout problem". U: *Neural Computing and Applications* 25.7-8 (2014), pp. 1871–1885.

[72] H. Hotelling. „Stability in competition". U: *The economic journal* 39.153 (1929), pp. 41–57.

[73] A. Ilić, D. Urošević, J. Brimberg i N. Mladenović. „A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem". U: *European Journal of Operational Research* 206.2 (2010), pp. 289–300.

[74] C. A. Irawan, S. Salhi i Z. Drezner. „Hybrid meta-heuristics with VNS and exact methods: application to large unconditional and conditional vertex p p-centre problems". U: *Journal of Heuristics* 22.4 (2016), pp. 507–537.

[75] W. G. Ireson. *Factory Planning and Plant Layout*. Prentice-Hall, 1952.

[76] W. Isard. *Location and space-economy: a general theory relating to industrial location, market areas, land use, trade, and urban structure*. The Regional science studies series. M.I.T. Press, 1956.

[77] D.-G. Kim i Y.-D. Kim. „A branch and bound algorithm for determining locations of long-term care facilities". U: *European Journal of Operational Research* 206.1 (2010), pp. 168–177.

[78] M. R. Korupolu, C. Plaxton i R. Rajaraman. „Analysis of a Local Search Heuristic for Facility Location Problems". U: *Journal of Algorithms* 37.1 (2000), pp. 146–188.

[79] B. Lazović, M. Marić, V. Filipović i A. Savić. „An integer linear programming formulation and genetic algorithm for the maximum set splitting problem". U: *Publications de l'Institut Mathematique* 92.106 (2012), pp. 25–34.

[80] M. Lejeune. „A variable neighborhood decomposition search method for supply chain management planning problems". U: *European Journal of Operational Research* 175.2 (2006), pp. 959–976.

[81] A. Lösch. *The Economics of Location*. The Economics of Location v. 1. Yale University Press, 1954.

[82] H. R. Lourenço, O. C. Martin i T. Stutzle. „Iterated local search". U: *International series in operations research and management science* (2003), pp. 321–354.

[83] F. V. Louveaux i D. Peeters. „A dual-based procedure for stochastic facility location". U: *Operations research* 40.3 (1992), pp. 564–573.

[84] R. Love, J. Morris i G. Wesolowsky. „Facilities Location: Models & methods". U: *Methods, North-Holland, New York-Amsterdam-London* (1988).

[85] E. Mansfield i H. H. Wein. „A model for the location of a railroad classification yard". U: *Management Science* 4.3 (1958), pp. 292–313.

[86] M. Maric, M. Tuba i J. Kratica. „One genetic algorithm for hierarchical covering location problem". U: *Proceedings of the 9th WSEAS International Conference on Evolutionary Computing (EC'08)*. 2008, pp. 122–126.

[87] F. Marić. „Formalization and implementation of modern SAT solvers". U: *Journal of Automated Reasoning* 43.1 (2009), pp. 81–119.

[88] M. Marić. „An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem". U: *Computing and Informatics* 29.2 (2012), pp. 183–201.

[89] M. Marić. „Variable Neighborhood Search for Solving the Capacitated Single Allocation Hub Location Problem". U: *Serdica Journal of Computing* 7.4 (2013), 343p–354p.

[90] M. Marić, Z. Stanimirović i S. Božović. „Hybrid metaheuristic method for determining locations for long-term health care facilities". English. U: *Annals of Operations Research* 227.1 (2015), pp. 3–23.

[91] M. Marić, Z. Stanimirović, A. Djenić i P. Stanojević. „Memetic Algorithm for Solving the Multilevel Uncapacitated Facility Location Problem". U: *Informatica* 25.3 (2014), pp. 439–466.

[92] M. Marić, Z. Stanimirović i N. Milenković. „Metaheuristic methods for solving the bilevel uncapacitated facility location problem with clients' preferences". U: *Electronic Notes in Discrete Mathematics* 39 (2012), pp. 43–50.

[93] M. Marić, Z. Stanimirović, N. Milenković i A. Đenić. „Metaheuristic approaches to solving large-scale Bilevel Uncapacitated Facility Location Problem with clients' preferences". U: *Yugoslav Journal of Operations Research* 25.3 (2015), pp. 361–378.

[94] M. Marić, Z. Stanimirović i P. Stanojević. „An efficient memetic algorithm for the uncapacitated single allocation hub location problem". U: *Soft Computing-A Fusion of Foundations, Methodologies and Applications* (2013), pp. 1–22.

[95] A. Marín. „The discrete facility location problem with balanced allocation of customers". U: *European Journal of Operational Research* 210.1 (2011), pp. 27–38.

[96] J. C. Martin. *Introduction to Languages and the Theory of Computation*. Vol. 4. McGraw-Hill NY, 1991.

[97] O. Martin, S. W. Otto i E. W. Felten. „Large-step Markov chains for the traveling salesman problem". U: (1991).

[98] S. Mišković, Z. Stanimirović i I. Grujičić. „An efficient variable neighborhood search for solving a robust dynamic facility location problem in emergency service network". U: *Electronic Notes in Discrete Mathematics* 47 (2015), pp. 261–268.

[99] S. Mišković, Z. Stanimirović i I. Grujičić. „Solving the robust two-stage capacitated facility location problem with uncertain transportation costs". U: *Optimization Letters* 11.6 (2017), pp. 1169–1184.

[100] A. Mjirda, B. Jarboui, J. Mladenović, C. Wilbaut i S. Hanafi. „A general variable neighbourhood search for the multi-product inventory routing problem". U: *IMA Journal of Management Mathematics* 27.1 (2016), pp. 39–54.

[101] N. Mladenović i P. Hansen. „Variable neighborhood search". U: *Computers & Operations Research* 24.11 (1997), pp. 1097–1100.

[102] N. Mladenović, J. Brimberg, P. Hansen i J. A. Moreno-Pérez. „The p-median problem: A survey of metaheuristic approaches". U: *European Journal of Operational Research* 179.3 (2007), pp. 927–939.

[103] N. Mladenović, M. Dražić, V. Kovačevic-Vujčić i M. Čangalović. „General variable neighborhood search for the continuous optimization". U: *European Journal of Operational Research* 191.3 (2008), pp. 753–770.

[104] N. Mladenović, M. Labbé i P. Hansen. „Solving the p-center problem with tabu search and variable neighborhood search". U: *Networks* 42.1 (2003), pp. 48–64.

[105] N. Mladenović, R. Todosijević i D. Urošević. „An efficient GVNS for solving traveling salesman problem with time windows". U: *Electronic Notes in Discrete Mathematics* 39 (2012), pp. 83–90.

[106] N. Mladenović, D. Urošević, S. Hanafi i A. Ilić. „A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem". U: *European Journal of Operational Research* 220.1 (2012), pp. 270–285.

[107] J. M. Moore. *Plant layout and design*. Prentice Hall, 1962.

[108] L. N. Moses. „Location and the theory of production". U: *The Quarterly Journal of Economics* 72.2 (1958), pp. 259–272.

[109] R. Muther. *Practical plant layout*. McGraw-Hill College, 1955.

[110] I. H. Osman i G. Laporte. „Metaheuristics: A bibliography". U: *Annals of Operations research* 63.5 (1996), pp. 511–623.

[111] M. Pedemonte, S. Nesmachnow i H. Cancela. „A survey on parallel ant colony optimization". U: *Applied Soft Computing* 11.8 (2011), pp. 5181–5197.

[112] M. P. Pérez, F. A. Rodríguez i J. M. Moreno-Vega. „A hybrid VNS–path relinking for the p-hub median problem". U: *IMA Journal of Management Mathematics* 18.2 (2007), pp. 157–171.

[113] J. Plesník. „A heuristic for the p-center problems in graphs". U: *Discrete Applied Mathematics* 17.3 (1987), pp. 263–268.

[114] Y. Rapp. „Planning of exchange locations and boundaries". U: *Ericsson Technics* 2 (1962), pp. 1–22.

[115] R. Reed. *Plant layout: Factors, principles, and techniques*. RD Irwin, 1961.

[116] C. R. Reeves, ur. *Modern Heuristic Techniques for Combinatorial Problems*. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[117] G. Reinelt. „TSPLIB–A Traveling Salesman Problem Library". U: *ORSA Journal on Computing* 3.4 (1991), pp. 376–384.

[118] M. G. Resende i R. F. Werneck. „A fast swap-based local search procedure for location problems". English. U: *Annals of Operations Research* 150.1 (2007), pp. 205–230.

[119] C. S. Revelle, H. A. Eiselt i M. S. Daskin. „A bibliography for some fundamental problem categories in discrete location science". U: *European Journal of Operational Research* 184.3 (2008), pp. 817–848.

[120] C. ReVelle i H. Eiselt. „Location analysis: A synthesis and survey". U: *European Journal of Operational Research* 165.1 (2005), pp. 1–19.

[121] E. Rolland, D. A. Schilling i J. R. Current. „An efficient tabu search procedure for the p-Median Problem". U: *European Journal of Operational Research* 96.2 (1997), pp. 329–342.

[122] J. Sáez-Aguado i P. C. Trandafir. „Some heuristic methods for solving p-median problems with a coverage constraint". U: *European Journal of Operational Research* 220.2 (2012), pp. 320–327.

[123] M. Sipser. *Introduction to the Theory of Computation*. Vol. 2. Thomson Course Technology Boston, 2006.

[124] A. Smithies. „Optimum location in spatial competition". U: *Journal of Political Economy* 49.3 (1941), pp. 423–439.

[125] L. V. Snyder. „Facility location under uncertainty: a review". U: *IIE Transactions* 38.7 (2006), pp. 547–564.

[126] Z. Stanimirovic, M. Maric, N. Radojicic i S. Bozovic. „Two efficient hybrid metaheuristic methods for solving the load balance problem". U: *Applied and Computational Mathematics* 13.3 (2014), 332–349.

[127] Z. Stanimirović, M. Marić, S. Božović i P. Stanojević. „An efficient evolutionary algorithm for locating long-term care facilities". U: *Information Technology And Control* 41.1 (2012), pp. 77–89.

[128] P. Stanojević i M. Marić. „Solving Large Scale Instances of Hub Location Problems with a Sub-problem Using an Exact Method". U: *IPSI Transactions on Internet Research* 11.1 (2015), pp. 1–6.

[129] P. Stanojević, M. Marić i Z. Stanimirović. „A hybridization of an evolutionary algorithm and a parallel branch and bound for solving the capacitated single allocation hub location problem". U: *Applied Soft Computing* 33 (2015), pp. 24–36.

[130] B. H. Steven. „An application of game theory to a problem in location strategy". U: *Papers in Regional Science* 7.1 (1961), pp. 143–157.

[131] T. Stützle. „Local search algorithms for combinatorial problems". U: *Darmstadt University of Technology PhD Thesis* 20 (1998).

[132] A. Takači, I. Štajner-Papuga, D. Drakulić i M. Marić. „An Extension of Maximal Covering Location Problem based on the Choquet Integral". U: *Acta Polytechnica Hungarica* 13.4 (2016).

[133] Đ. Takači, M. Marić, G. Stankov i A. Djenić. „Efficiency of using VNS algorithm for forming heterogeneous groups for CSCL learning". U: *Computers & Education* 109 (2017), pp. 98–108.

[134] R. Todosijević, D. Urošević, N. Mladenović i S. Hanafi. „A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem". U: *Optimization Letters* 11.6 (2017), pp. 1109–1121.

[135] A. M. Turing. „On computable numbers, with an application to the Entscheidungsproblem". U: *Proceedings of the London mathematical society* 2.1 (1937), pp. 230–265.

[136] M. S. Uddin. „Hybrid genetic algorithm and variable neighborhood search for dynamic facility layout problem". U: *Open Journal of Optimization* 4.04 (2015), p. 156.

[137] D. Valinsky. „Symposium on Applications of Operations Research to Urban Services—A Determination of the Optimum Location of Fire-Fighting Units in New York City". U: *Journal of the Operations Research Society of America* 3.4 (1955), pp. 494–512.

[138] S. Voß, S. Martello, I. H. Osman i C. Roucairol. *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Springer Science & Business Media, 2012.

[139] A. Weber i C. Friedrich. *Theory of the location of industries*. Materials for the study of business. University of Chicago Press, 1929.

[140] A. Weber. *Ueber den standort der industrien*. Vol. 2. Ripol Klassik, 1909.

[141] S. Wersan, J. Quon i A. Charnes. „Systems analysis of refuse collection and disposal practices". U: *American Public Works Association Yearbook, American Public Works Association* (1962).

[142] R. Whitaker. „A fast algorithm for the greedy interchange for large-scale clustering and median location-problems". U: *Infor* 21.2 (1983), pp. 95–108.

[143] B. Yu, H. Zhu, W. Cai, N. Ma, Q. Kuang i B. Yao. „Two-phase optimization approach to transit hub location – the case of Dalian". U: *Journal of Transport Geography* 33 (2013), pp. 62–71.

[144] C. Zhang, Z. Lin i Z. Lin. „Variable neighborhood search with permutation distance for QAP". U: *Knowledge-Based Intelligent Information and Engineering Systems*. Springer. 2005, pp. 905–905.

[145] M. Živković. *Algoritmi*. Matematički fakultet, 2000.

Biografija autora

Aleksandar Đenić rođen je 12. februara 1987. godine u Beogradu gde je završio osnovnu školu „Veljko Dugošević" kao đak generacije i dobitnik Vukove diplome. Matematičku gimnaziju u Beogradu uspešno je završio 2006. godine. Tokom školovanja učestvovao je i nagrađivan je na takmičenjima iz matematike, fizike i informatike, kao i na sportskim takmičenjima. Obrazovanje nastavlja na Matematičkom fakultetu, Univerziteta u Beogradu gde je diplomirao sa prosečnom ocenom 9,74 2010. godine na smeru Računarstvo i informatika. Master akademske studije upisao je 2010. godine, a završio ih je 2011. godine sa prosečnom ocenom 10,00 i odbranjenom master tezom pod nazivom „Rešavanje nekih problema zadovoljenja ograničenja primenom egzaktnih i heurističkih metoda" pod mentorstvom dr Miroslava Marića. Po završetku master studija upisao je doktorske studije na istom fakultetu, modul Informatika. Položio je sve ispite na doktorskim studijama sa prosečnom ocenom 10,00.

Od 2010. do 2012. godine, Aleksandar Đenić je bio zaposlen kao saradnik u nastavi, a potom 2012. godine je izabran u zvanje asistenta za naučnu oblast Računarstvo i informatika na Matematičkom fakultetu, Univerziteta u Beogradu, gde je bio zaposlen do 2016. godine. Držao je vežbe iz sledećih predmeta: Programiranje 1, Programiranje 2, Arhitektura računara, Edukativni softver, Teorija operativnih sistema i Metodika nastave računarstva.

Od 2011. godine do 2016. godine je bio učesnik na naučnom projektu broj 174010 pod nazivom „Matematički modeli i metode optimizacije velikih sistema", pod rukovodstvom dr Nenada Mladenovića, na Matematičkom institutu SANU, u okviru tekućeg Programa istraživanja naučnog i tehnološkog razvoja, finansiranog od strane Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije.

Nakon rada na fakultetu, veoma inspirisan naučnim dostignućima i radom sa mladim ljudima, posvetio se preduzetništvu, odnosno osnivanju i razvoju visoko tehnološke kompanije zajedno sa svojim timom. Najveći uspeh ostvaruje tokom 2017. godine, kada sa svojim timom dobija investiciju za najinovativnije timove prestižnog holandskog investicionog fonda.

Прилог 1.

Изјава о ауторству

Потписани-а Александар Ђенић

број индекса 2043/2011

Изјављујем

да је докторска дисертација под насловом

Решавање дискретних локацијских проблема применом методе променљивих околина

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, _____

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Александар Ђенић

Број индекса 2043/2011

Студијски програм Информатика

Наслов рада Решавање дискретних локацијских проблема применом методе променљивих околина

Ментор др Мирослав Марић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

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

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, _____

_____

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Решавање дискретних локацијских проблема применом методе променљивих околина

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, _____

_____

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.