Univerzitet u Beogradu
Matematički fakultet

Master rad

Heuristički pristup rešavanju lokacijskog problema sa nadmetanjem

Student:
Aida Zolić

Mentor:
prof. dr Zorica Stanimirović

Beograd, 2016

Heuristički pristup rešavanju lokacijskog problema sa nadmetanjem

Student: Aida Zolić
1023/2015

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

Članovi komisije: prof. dr Filip Marić
Matematički fakultet
Univerzitet u Beogradu

prof. dr Miroslav Marić
Matematički fakultet
Univerzitet u Beogradu

Sadržaj

Heuristički pristup rešavanju lokacijskog problema sa nadmetanjem

Apstrakt

Predmet istraživanja prikazanog u ovom radu je problem maksimalnog zauzimanja tržišta koji predstavlja jednu varijantu lokacijskog problema sa nadmetanjem. Razmatrani problem je $\mathcal{NP}$-težak, a raznovrsne primene podrazumevaju ulazne podatke velikih dimenzija. Međutim, egzaktnom metodom implementiranom u CPLEX rešavaču, optimalno rešenje je pronađeno samo za veoma mali broj instanci većih dimenzija. Sa tim u vidu, predloženo je nekoliko pristupa kojim su uspešno rešene sve instance velikih dimenzija. Implementirane su tri metaheurističke metode prilagođene razmatranom problemu: metoda promenljivih okolina, optimizacija kolonijom pčela i genetski algoritam. Predložena metoda promenljivih okolina je kombinovana sa optimizacijom kolonijom pčela u okviru koncepta hibridnog metaheurističkog algoritma. Svi razvijeni heuristički pristupi su međusobno upoređeni testiranjem na instancama problema različitih dimenzija. Na instancama manjih dimenzija, heurističke metode su upoređene i sa rezultatima dobijenim CPLEX rešavačem.

ključne reči: optimizacija kolonijom pčela, metoda promenljivih okolina, genetski algoritam, lokacijski problem sa nadmetanjem, problem maksimalnog zauzimanja

Heuristic approach to solving the maximum capture problem

Abstract

The research presented in this thesis deals with the maximum capture problem which represents one variant of the competitive facility location problem. The considered problem is $\mathcal{NP}$-hard, and its various applications imply input data of large dimensions. However, exact methods implemented in the framework of commercial CPLEX solver can provide optimal solutions only for small size MAXCAP instances and few instances of larger dimensions. In order to solve large and large-scale MAXCAP problem instances, three metaheuristic methods for solving MAXCAP are implemented: variable neighborhood search, bee colony optimization and genetic algorithm. The proposed variable neighborhood search method and bee colony optimization are combined within the frame of a hybrid metaheuristic algorithm. Each of the proposed heuristic approaches is tested on instances of different dimensions and the obtained results are compared. On small dimension instances, the heuristic methods are compared to the results acquired from the CPLEX solver as well.

key words: bee colony optimization, variable neighborhood search, genetic algorithm, competitive facility location, maximum capture

Uvod

Oblast matematike i računarstva posvećena istraživanju lokacijskih problema je u stalnom razvoju. Zbog njihove široke primene u praksi, u literaturi se mogu naći brojne varijante lokacijskih problema. Predmet izučavanja ovog master rada je problem maksimalnog zauzimanja tržišta (engl. *Maximum Capture Problem*, MAXCAP), a koji spada u klasu lokacijskih problema sa nadmetanjem.

Rapidan tehnološki razvoj je doprineo primetnom poboljšanju moći i performansi računarskih sistema. Međutim, instance većine $\mathcal{NP}$-teških problema velikih dimenzija je dalje je praktično nemoguće rešiti usled ograničenja vremenskih i memorijskih resursa. U klasu $\mathcal{NP}$-teških problema spada većina problema kombinatorne optimizacije. Iz tog razloga, u radu su predložene implementacije tri heurističke metode i jedne hibridne metode za rešvanje problema maksimalnog zauzimanja.

Rad je podeljen u nekoliko poglavlja. U prvom poglavlju izložen je kratak istorijski pregled lokacijskih problema, posebno lokacijskih problema sa nadmetanjem, kao i neki od kriterijuma po kojima se oni mogu klasifikovati. U drugom poglavlju opisan je problem maksimalnog zauzimanja i postojeće metode rešavanja iz literature. U trećem poglavlju su opisane heurističke metode prilagođene razmatranom problemu: optimizacija kolonijom pčela, metoda promenljivih okolina i genetski algoritam, kao i predlog hibridizacije optimizacije kolonijom pčela i metode promenljivih okolina. U četvrtom poglavlju su navedeni rezultati implementiranih metoda na test instancama problema maksimalnog zauzimanja. Peto poglavlje sadrži analizu dobijenih rezultata i poređenja predloženih metoda. Na instancama manjih dimenzija, rezultati heurističkih metoda su upoređeni sa rezultatima egzaktne metode, koja je implementirana pomoću CPLEX rešavača. Poslednje poglavlje sadrži komentare i mogućnosti za dalja istraživanja.

1
Lokacijski problemi

1.1 Istorijat i klasifikacija lokacijskih problema

Lokacijski problemi se u najširem smislu mogu shvatiti kao pitanja oblika "Gde treba smestiti neke stvari?". Postoje podeljena mišljenja o tome ko je prvi matematički definisao lokacijski problem. Većina izvora se slaže da je to bio Ferma [1] kada je postavio sledeći zadatak: ako su date tri tačke u ravni, odrediti četvrtu tačku takvu da je suma njenih rastojanja do preostalih tačaka minimalna [9]. Toričeli [2] je ponudio nekoliko geometrijskih rešenja, a doprinos razvoju lokacijskih problema su dali i Kavalijeri [3], Vivijani [4], Roberval [5] i mnogi drugi. Ipak, veliki praktični značaj Fermaovog zadatka prvi je uočio Veber [6] više od dva veka kasnije. Veber je preformulisao Fermaov problem kako bi ga primenio na situaciju koji se često javlja u industriji. Od tri fiksirane tačke, dve je označio kao sirovine, a treću kao lokaciju korisnika. Svakoj tački dodelio je određenu težinu. Cilj je odrediti gde treba postaviti četvrtu tačku - industrijsko postrojenje, tako da troškovi transporta budu minimalni. Ovako postavljen problem Veber je rešio konstruktivnim putem. U slučaju većeg broja korisnika, Veberov problem [9] je formulisan na sledeći nacin: Za date lokacije korisnika (x_i, y_i), $i = 1, ..., n$ sa težinama w_i (cenama transporta po jedinici rastojanja), odrediti optimalnu lokaciju skladišta (x^*, y^*) tako da se minimizuju ukupni troškovi. Ili, matematički zapisano:

$$\min_{(x,y)} W(x, y) = \sum_{i=1}^{n} w_i d_i(x, y)$$

gde je $d_i(x, y) = \sqrt{(x - x_i)^2 + (y - y_i)^2}$ Euklidsko rastojanje tačaka (x, y) i (x_i, y_i) za $i = 1, 2, ..., n$.

[1] Pjer de Ferma (franc. *Pierre de Fermat*, 1601-1665), pravnik i jedan od najznačajnijih francuskih matematičara XVII veka. Bavio se između ostalog teorijom brojeva, algebrom, verovatnoćom i geometrijom. Takođe je bio jedan od začetnika diferecijalnog računa.

[2] Evanđelista Toričeli (it. *Evangelista Torricelli*, 1608–1647), italijanski fizičar i matematičar, najpoznatiji po izumu barometra.

[3] Bonaventura Kavalijeri, (it. *Bonaventura Cavalieri*, 1598-1647), italijanski matematčar. Bavio se optikom, kretanjem i logaritmima, kao i beskonačno malim veličinam i integralnim računom.

[4] Vićenco Vivijani, (it. *Vincenzo Viviani*, 1622–1703), italijanski naučnik. Toričelijev učenik i Galilejev pomoćnik, bavio se fizikom i geometrijom.

[5] Žil de Roberval, (franc. *Gilles de Roberval*, 1602–1675), francuski matematičar. Pored matematike, proučavao je astronomiju i podržao Kopernikov heliocentrični sistem.

[6] Alfred Weber (nem. *Alfred Weber*, 1868–1958), nemački ekonomista, geograf i sociolog. Dao je izuzetan doprinos razvoju moderne ekonomske geografije.

Lokacijski problemi su tokom prethodnih decenija izučavani kako sa teorijskog tako i sa praktičnog aspekta, imajući u vidu brojne oblasti primene [23]. Generalno, zadatak se odnosi na uspostavljanje određenog skupa objekata i njihovo povezivanje sa već postojećim objektima koji imaju zahteve nekog tipa. Najčešće je potrebno odrediti lokacije za neke vrste uslužnih centara koji se zbog toga uobičajeno nazivaju *snabdevači* ili *resursi*. Postojeći objekti koji tu uslugu koriste se nazivaju *klijenti*, *korisnici* ili *potrošači*. Dizajniranje distributivne ili transportne mreže je od strateške važnosti za kompanije koje posluju u raznim sektorima (industrija, trgovina, transport itd). Neke situacije koje se mogu modelirati na ovaj način su određivanje lokacija supermarketa u okviru jednog lanca, centara brze kurirske službe ili autobuskih i železničkih stanica, kao i planiranje lokacija škola, bolnica ili vatrogasnih i policijskih stanica. Bez obzira na konkretnu primenu, postavlja se zahtev da potrebe korisnika budu zadovoljene uz minimalne troškove ili maksimalan profit. Ponekad se zahteva da se ispune dodatni uslovi, na primer fiksiran broj objekata, ograničeni kapaciteti snabdevača i/ili klijenata, minimalna udaljenost na kojoj se objekti moraju nalaziti itd. Pregled lokacijskih problema i odgovarajucih matematickih modela moze se naci u [8] i [11].

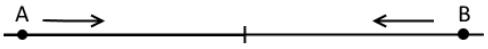
1.2 Lokacijski problemi sa nadmetanjem

Kod osnovnih varijanti lokacijskih problema podrazumeva se da kompanija koja uspostavlja snabdevače ili drži monopol na tržištu ili prihvata cenu koja preovladava na tržištu (engl. *price-taker*). Uticaj konkurencije se pod tim pretpostavkama zanemaruje. Međutim, ovako pojednostavljeni modeli često ne odgovaraju realnim situacijama u dovoljnoj meri. Klasa *lokacijskih problema sa nadmetanjem* (engl. *Competitive Facility Location*, CFL) obuhvata varijante lokacijskih problema koje su prilagođene strukturama tržišta u kojima konkurencija igra važnu ulogu.

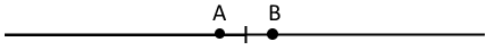
Hoteling [7] se među prvima bavio analizom uticaja nadmetanja na ponašanje konkurentskih kompanija na tržištu. U radu [21], Hoteling je predstavio *princip minimalne diferencijacije* (engl. *principle of minimum differentiation*). Po tom principu, kompanijama je često najisplativije da svoju uslugu što više prilagode usluzi koju pružaju konkurentske kompanije. Ovu tvrdnju je najjednostavnije predstaviti na linearnom modelu, kao što je urađeno radu [21] ali isti princip se može primeniti i na kompleksnije situacije. Recimo da dve firme A i B žele da otvore svoje prodavnice u ulici, svaka sa ciljem da zauzme što više kupaca. Pritom se pretpostavlja da kupci uvek biraju najbližu prodavnicu, nezavisno od vlasnika. Da je u pitanju monopol, jedna firma bi lociranjem prodavnice bilo gde u ulici prisvojila sve kupce. U slučaju duopola, ravnoteža se uspostavlja kada svaka od firmi A i B opslužuje po pola tržišta. Takva raspodela se može postići na više načina. Jedna mogućnost je da firme A i B lociraju svoje prodavnice na suprotnim krajevima ulice (slika 1.1a). Ipak, svaka od firmi i dalje teži da poveća svoj uticaj. Zbog toga će svoju prodavnicu pomerati sve više ka centru u pokušaju da preuzme deo kupaca od suparničke firme. Konačno, ispostavlja se da će prodavnice biti locirane jedna blizu drugoj i to otprilike na sredini ulice (slika 1.1b). Za kupce bi bilo najpovoljnije

[7] Harold Hoteling, (engl. *Harold Hotelling*, 1895-1973), američki statističar i ekonomista. Pored drugih doprinosa, uveo je Hotelingovo pravilo, Hotelingovu lemu i Hotelingov zakon u oblasti ekonomije i Hotelingovu T-kvadrat raspodelu u oblasti statistike.

kada bi prodavnice bile locirane na četvrtini udaljenosti od krajeva ulice. Iako u sva tri slučaja firme A i B zauzimaju po polovinu tržišta, nijedna od njih ne želi da se udalji od sredine i tako dopusti drugoj da se proširi.



(a) A i B teže da prošire zauzimanje



(b) Postignuta je ravnoteža

Slika 1.1: Hotelingov linearni model tržišta

Ulica iz Hotelingovog linearnog modela se može shvatiti kao metafora za sličnost proizvoda - kompanije uglavnom teže da uspostave neki vid ravnoteže, usklade svoj proizvod prema konkurentima. Nije teško uočiti mnogo primera ovakvog ponašanja u svakodnevnom životu, počev od malih lokalnih prodavnica, barova ili benzinskih stanica, preko velikih lanaca restorana brze hrane, supermarketa, modnih kuća, pa do oblasti u kojima primena ovog principa nije očigledna, poput političkih izbora.

U literaturi postoje brojni pokušaji klasifikacije lokacijskih problema sa nadmetanjem, međutim nijedna ne obuhvata sve poznate varijante ovih problema. Neki kriterijumi koji se mogu posmatrati prilikom klasifikacije predloženi u [23] su:

1. **Topografija prostora**
   Prostor u kom je problem definisan direktno utiče na izbor metode za njegovo rešavanje. U *kontiunalnim* problemima objekti mogu biti locirani bilo gde u prostoru ($\mathbb{R}^2, \mathbb{R}^3$...), dok *diskretni* problemi dozvoljavaju lociranje objekata na konačnom skupu lokacija. U *mrežnim* modelima se novi objekti mogu locirati na bilo kom čvoru zadate mreže. Po određivanju tipa prostora, potrebno je definisati i odgovarajuću meru rastojanja. U kontinualnom slučaju se uvodi neka metrika, recimo Euklidsko rastojanje, u diskretnom slučaju je data matrica udaljenosti, dok se u mrežnom najčešće uzima definicija najkraćeg puta. Diskretni i mrežni problemi se uobičajeno rešavaju metodama celobrojnog programiranja, dok se kontinualni problemi se uglavnom formulišu kao problemi nelinearnog programiranja i rešavaju se adekvatnim metodama nelinearne optimizacije.

2. **Vid nadmetanja**
   Lokacijski problemi se mogu klasifikovati i prema ponašanju kompanija koje se takmiče na tržištu. Kada kompanija želi da otvori nove objekte na nekim od potencijalnih lokacija, mnoge karakteristike konkurentske firme, poput lokacije njenih objekata, strategija korišćcena za njihovu lokaciju i cena proizvoda mogu biti veoma bitne. *Statički* lokacijski problemi podrazumevaju da su sve te karakteristike kompaniji poznate pre ulaska na tržište. Takođe, statički modeli pretpostavljaju da su te karakteristike fiksirane, tako da po uspostavljanju novih objekata ne dolazi do promena. Sa druge strane, *dinamički* i *sekvencijalni* CFL problemi podrazumevaju odgovarajuće reakcije konkurentskih kompanija na izmene u situaciji na tržištu. U dinamičkim modelima sve kompanije donose odluke istovremeno, dok u sekvencijalnim postoji određena hijerarhija.

3. **Broj objekata**
   Parametar p koji predstavlja broj objekata koje treba uspostaviti može biti

unapred zadat ili može biti određen optimalnim rešenjem problema. Obično se razlikuju slučajevi kada je potrebno otvoriti jedan objekat (engl. *single-facility*, $p = 1$) i više objekata (engl. *multi-facility*, $p > 1$).

4. **Način pridruživanja**
   Kada se uspostavlja više uslužnih objekata, potrebno je definisati kako se korisnici pridružuju određenom objektu. U javnom sektoru (škole, bolnice, poštanski centari i sl.) često se alokacija obavlja uporedo sa planiranjem. U drugim slučajevima (restorani, supermarketi i sl.) se ta odluka prepušta samim korisnicima. U obe situacije se razlikuju tri tipa alokacije: jednostruka alokacija (engl. *single allocation*) podrazumeva da se svaki korisnik pridružuje tačno jednom uspostavljenom objektu, višestruka alokacija (engl. *multiple allocation*) podrazumeva da se svaki korisnik može pridružiti proizvoljnom broju uspostavljenih objekata i r-alokacija (engl. *r-allocation*) koja podrazumeva da se svaki korisnik može pridružiti fiksiranom broju uspostavljenih objekata r.

5. **Karakteristike korisnika**
   Ponašanje korisnika može biti bitan faktor u određivanju optimalnih lokacija. Kaže se da su zahtevi korisnika *elastični* kada zavise od cene, udaljenosti, kvaliteta i sličnih karakteristika uslužnih objekata. Zahtevi korisnika su *neelastični* kada su unapred poznati i nepromenljivi. Ukoliko je odluka izbora snabdevača na korisnicima, njihovo ponašanje može biti *determinističko* ili *probabilističko/nepouzdano*. Determinističko ponašanje podrazumeva da se sa sigurnošću zna koji objekat će korisnik izabrati. Na primer, zna se da svaki korisnik uvek bira najbliži objekat. Probabilističko ponašanje podrazumeva da korisnikov izbor zavisi od više faktora, kao što su cena, udaljenost ili kvalitet. U tom slučaju se obično definiše *funkcija privlačnosti* koja služi da se izmeri uticaj određenog skupa karakteristika uslužnih objekata na korisnikovu odluku.

6. **Cilj problema**
   Namena objekata koje treba locirati određuje tip funkcije cilja odgovarajućeg matematičkog modela. Neke objekte korisnici smatraju poželjnim (prodavnice, restorani...), tako da je cilj problema da se oni lociraju što bliže korisnicima (engl. *pull objective*). Međutim, ponekad je neophodno uspostaviti uslužne objekte koji imaju i izvesne nepoželjne karakteristike (fabrike hemijskih proizvoda, deponije...). Takvi objekti ne smeju biti previše blizu korisnika (engl. *push objective*). Kada objekte koji se otvaraju odlikuju i poželjne i nepoželjne osobine, primenjuje se kombinacija ova dva pristupa (engl. *pull-push objective*). Mogu se nametnuti i drugi uslovi, recimo minimalno rastojanje na kom moraju biti uspostavljeni objekti.

7. **Cenovna politika**
   Postoje dva glavna načina za određivanje cene proizvoda. Jedan je da kompanija određuje jedinstvenu fabričku cenu proizvoda, a da troškove prevoza plaćaju korisinici (engl. *mill-price policy*). Drugi je da kompanija organizuje neki vid isporuke. U tom slučaju, cena proizvoda uključuje i troškove prevoza (engl. *delivered price policy*), koja zavisi od udaljenosti korisnika od skladišta. Ukoliko cene nisu univerzalne, već svaka kompanija određuje svoju cenu, to mogu uraditi simultano ili sekvencijalno.

1.3 Primeri lokacijskih problema sa nadmetanjem iz literature

U literaturi se može pronaći veliki broj varijanti lokacijskih problema. U ovoj sekciji će biti navedeni samo neki karakteristični primeri CFL problema i metode koje su u literaturi predložene za njihovo rešavanje.

Sekvencijalni lokacijski problemi sa nadmetanjem su inspirisani Stakelbergovom strateškom igrom [8]. U igri učestvuju dva tipa takmičara: *vođa* (engl. *leader*) i *pratilac* (engl. *follower*). Vođa ima određenu prednost (drži monopol, ima više resursa i sl.) i prvi locira svoje objekte, a pratilac uspostavlja svoje objekte u skladu sa tim. Dva osnovna sekvencijalna problema predložio je Hakimi [14]. Jedan od njih se odnosi na vođu, a drugi na pratioce. Vođa locira $p, p \geq 1$ objekata, znajući pritom da će pratioci na osnovu toga locirati $r, r \geq 1$ objekata, što se naziva *problemom* $(r|p)$ *težišta* (engl. *centroid problem*). Pratioci lociraju r objekata, pri čemu im je poznat skup X_p objekata koje je vođa već uspostavio, što predstavlja *problem* $(r|X_p)$ *medoide* (engl. *medianoid problem*). Pretpostavka je da korisnici biraju uslužni centar na osnovu udaljenosti, a između objekata na istoj razdaljini preferiraju onaj koji pripada vođi. U [39] predložena je primena *optimizacije rojem čestica* (engl. *Particle Swarm Optimization*, PCO) za rešavanje dva Hakimijeva problema. U probabilističkoj generalizaciji problema $(r|X_p)$ težišta predloženoj u [49], uvedena je Hafova funkcija za merenje privlačnosti objekta. Po Hafovom modelu, privlačnost objekta je direktno proporcionalna kvalitetu usluge i obrnuto proporcionalna udaljenosti od korisnika. Problem je prvo diskretizovan, a zatim rešen kombinacijom tri kombinatorne heuristike i metode grananja i ograničavanja (engl. *Branch and Bound*, BnB). Dalja uopštenja sekvencijalnih lokacijskih problema uključuju više nivoa (tipova takmičara) i/ili varirajuće cene. Predložena rešenja uglavnom podrazumevaju neku kombinaciju heurističkih metoda [23].

Statički problemi, odnosno problemi kod kojih su lokacije i karakteristike objekata suparničkih kompanija fiksirane i unapred poznate takođe se javljaju u dve varijante. Jedan deterministički statički problem, koji je u literaturi poznat pod nazivom problem maksimalnog zauzimanja tržišta, detaljnije je opisan u narednom poglavlju. Probabilistički problemi uglavnom koriste Hafov model privlačnosti, ali ne uvek. Recimo, Nakaniši i Kuper su definisali *multiplikativni model nadmetanja* (engl. *multiplicative competitive interaction*, MCI). U MCI modelu privlačnost je definisana kao proizvod određenih stepena težina različitih atributa objekta. Postoje i modeli koji u tu svrhu koriste eksponencijalnu funkciju. Moguće je postaviti i pitanje pronalaska optimalnih vrednosti privlačnosti. Zbog kompleksne funkcije cilja, probabilistički CFL problemi se najčešće rešavaju nekom pohlepnom (engl. *greedy*) aproksimativnom metodom.

U svim do sada navedenim primerima važi pretpostavka da korisnici imaju donekle ograničenu slobodu izbora uslužnog objekta. U praksi se često nailazi na situacije u kojima reakcije korisnika nije moguće precizno predvideti. Zbog brojnih spoljnih i promenljivih faktora (npr. gužve u saobraćaju ili samom objektu), može se dogoditi da želje i sklonosti korisnika ne odgovaraju onome što se očekivalo kada

[8] Hajrih fon Stakelberg (nem. *Heinrich von Stackelberg*, 1905-1946), nemački ekonomista, bavio se teorijom igara i industrijskim menadžmentom. Stakelbergova igra je jedan od poznatih modela duopola [46].

su planovi lokacija pravljeni. Iz tog razloga može biti neophodno izvršiti odgovarajuće izmene usluge, recimo prilagoditi cene kako bi se privukao veći broj potrošača. U radu [51] razmatran je problem maksimizacije profita kompanije koja ulazi na tržište. Cilj problema je odrediti optimalne lokacije za uspostavljanje uslužnih objekata ako se zna da će cene na tržištu zavisiti od izabrane lokacije. Takav problem sadrži implicitno ograničenje koje definiše vezu između cena i nivoa proizvodnje, što prestavlja izvesnu teškoću pri rešavanju. Da bi se to prevazišlo, u radu [51] su predložena dva modela prostornog nadmetanja, za čije su rešavanje dalje korišćene varijacione nejednačine. U radu [24], definisani su modeli u kojima izbori klijenta utiču na nivo ponuđene usluge. Kompanije teže da uz minimalne troškove ponude što bolje usluge, dok korisnici žele da dobiju najbolju uslugu po što manjoj ceni. Zadatak se sastoji u uspostavljanju nekog vida ravnoteže kojom bi bile zadovoljne obe strane, i snabdevači i potrošači. Pregled većine postojećih varijanti diskretnih lokacijskih problema sa nadmetanjem i metoda za njihovo rešavanje se može naći u [23].

Za razliku od diskretnih i mrežnih CFL problema, u kontinualnom slučaju ne mogu se eksplicitno nabrojati sve potencijalne lokacije za izgradnju objekata. Zapravo, može se reći da se u kontinualnim problemima tražene optimalne lokacije generišu (engl. *site generating*) kao rezultat rešavanja problema. Uobičajeno je da se za prostor rešenja uzima ravan $\mathbb{R}^2$, prostor $\mathbb{R}^3$ ili, opštije, prostor $\mathbb{R}^n$. Važno je paziti da se objekti lociraju u okviru prethodno definisanih *dopustivih oblasti* (engl. *feasible regions*). Dopustive oblasti se najčešće definišu relativno jednostavnim jednačinama tako da imaju formu pravougaonika, kruga ili neke slične figure. Problemi bez tako nametnutih ograničenja se takođe razmatraju, ali sa više teorijskog aspekta i nemaju veliki praktični značaj. U [34] Plastria daje pregled kontinualnih lokacijskih problema i različitih metrika koje se mogu uvesti.

Metode za rešavanje kontinualnih CFL problema se zbog prirode prostora u kom su definisani razlikuju od algoritama koji se primenjuju na diskretne i mrežne probleme. Jedan od prvih algoritama namenjenih rešavanju kontinualnih problema je *veliki kvadrat, mali kvadrat* algoritam (engl. *Big Square Small Square*, BSSS) [33]. BSSS je geometrijska procedura zasnovana na metodi grananja i ograničavanja. Jedna od modifikacija BSSS algoritma koja se pokazala kao veoma efikasna je algoritam poznat pod nazivom *veliki trougao, mali trougao* (Big Triangle Small Triangle, BTST) [10].

2
Problem maksimalnog zauzimanja tržišta

Predmet istraživanja prikazanog u ovom radu je jedna varijanta lokacijskog problema sa nadmetanjem koja se u literaturi može naći pod nazivom *problem maksimalnog zauzimanja* (engl. *Maximum Capture Problem*, MAXCAP). Ovaj model je predložio Čarls ReVel u radu [37] objavljenom 1986. godine. Motivaciju za MAXCAP pronašao je proučavanjem tipičnih zahteva i situacija u oblasti trgovine i maloprodaje. Predloženi MAXCAP problem je uveo izvesne novine u do tada izučavane i rešavane lokacijske probleme. Osnovna formulacija i metode rešavanja problema maksimalnog zauzimanja su dalje razrađeni i proširenu radu [44].

Prema kriterujumima navedenim u prethodnom poglavlju, MAXCAP se može klasifikovati kao *statički deterministički diskretni lokacijski problem sa nadmetanjem*. Problem se posmatra sa stanovišta kompanije A koja namerava da uđe na tržište. Pretpostavlja se da u tom trenutku na tržištu postoji određen broj korisnika i već uspostavljenih uslužnih objektata konkurentske kompanije B. Ne gubeći na opštosti, može se smatrati da na tržištu ne postoje još neke kompanije ili objekti. Tržište je predstavljeno mrežom čiji čvorovi odgovaraju korisnicima. Glavne karakteristike problema su sledeće:

- Pozicije konkurentskih objekata su fiksirane i poznate;
- Pozicije potrošačkih objekata su fiksirane i poznate;
- Problem je *jednoproizvodni*, tj. nezavisno od vlasnika u svim objektima se prodaje isti proizvod ili pruža ista usluga;
- Ponašanje klijenata je neelastično, tj. svaki korisnik uvek bira najbliži uslužni objekat;
- Cena po jedinici mere je fiksirana i jednaka u svim objektima obe kompanije;
- Ukoliko obe kompanije lociraju objekte na jednakoj udaljenosti od korisnika, svakoj će pripasti po polovina njegove potražnje.

Cilj kompanije A je da maksimizuje realizovano zauzimanje tržišta. Drugim rečima, iz nekog konačnog skupa potencijalnih lokacija treba odrediti tačno p lokacija za uspostavljanje objekata kompanije A koje će joj omogućiti da prisvoji maksimalnu potražnju korisnika od konkurenta.

2.1 Matematička formulacija

Matematička formulacija koja će biti izložena u ovoj podsekciji je predložena u radu [37]. Ova formulacija koristi sledeću notaciju:

I – skup indeksa lokacija korisnika;
J – skup indeksa potencijalnih lokacija uslužnih objekata;
w_i – potražnja korisnika $i \in I$;
d_{ij} – udaljenost od lokacije $i$do lokacije j;
b_i – uslužni objekat kompanije B koji je najbliži korisniku i;
d_{ib_i} – udaljenost korisnika i od njemu najbližeg uslužnog objekta kompanije B;
$P_i = \{j \in J | d_{ij} < d_{ib_i}\}$;
$T_i = \{j \in J | d_{ij} = d_{ib_i}\}$.

Skup P_i je skup indeksa lokacija sa kojih bi kompanija A potpuno pridobila korisnika i ukoliko bi tu otvorila uslužni objekat. Skup T_i sadrži indekse lokacija koje su jednako udaljene od potrošača i kao najbliži objekat kompanije B. Ukoliko bi A uspostavila objekat na nekoj od tih lokacija, svakoj kompaniji bi pripala po polovina potražnje korisnika i.

Matematička formulacija koristi tri skupa promenljivih:

$$y_i = \begin{cases} 1, & \text{ukoliko kompanija A pridobija potrošača } i; \\ 0, & \text{inače;} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{ukoliko kompanije A i B dele potražnju potrošača } i; \\ 0, & \text{inače;} \end{cases}$$

$$x_j = \begin{cases} 1, & \text{ukoliko kompanija A uspostavlja objekat na lokaciji } j; \\ 0, & \text{inače.} \end{cases}$$

Prema [37], MAXCAP se može formulisati kao problem celobrojnog linearnog programiranja na sledeći način:

$$\max \sum_{i \in I} w_i y_i + \sum_{i \in I} \frac{w_i}{2} z_i \tag{2.1}$$

pri uslovima

$$y_i \leq \sum_{j \in P_i} x_j \ , \qquad \forall i \in I \tag{2.2}$$

$$z_i \leq \sum_{j \in T_i} x_j \ , \qquad \forall i \in I \tag{2.3}$$

$$y_i + z_i \leq 1 \ , \qquad \forall i \in I \tag{2.4}$$

$$\sum_{j \in J} x_j = p \tag{2.5}$$

$$y_i \in \{0, 1\} \ , \qquad \forall i \in I \tag{2.6}$$

$$z_i \in \{0, 1\} \ , \qquad \forall i \in I \tag{2.7}$$

$$x_j \in \{0, 1\} \ , \qquad \forall j \in J \tag{2.8}$$

Funkcija cilja (2.1) definisana je kao suma potražnji korisnika koje kompanija A može da pridobije uspostavljajući p objekata. Ukoliko je $y_i = 1$, vrednost funkcije cilja se uvećava za w_i, a ukoliko je $z_i = 1$, uvećava se samo za $\frac{w_i}{2}$, jer je drugu polovinu pridobila kompanija B.

Ograničenje (2.2) dopušta da A može pridobiti potrošača i samo ukoliko uspostavi bar jedan objekat koji mu je bliži u odnosu na objekat b_i kompanije B. Slično, zbog ograničenja (2.3) kompanije A i B dele potražnju korisnika i samo ako A uspostavi bar jedan objekat tačno na udaljenosti d_{ib_i}. Ograničenje (2.4) obezbeđuje tri moguća scenarija za svakog od korisnika $i \in I$: ili ga u potpunosti pridobija kompanija A (u tom slučaju je $y_i = 1, z_i = 0$) ili njegova potražnja ravnomerno podeljena između kompanija A i B (u tom slučaju je $y_i = 0, z_i = 1$) ili ga pridobija samo kompanija B (u tom slučaju je $y_i = 0, z_i = 0$). Ograničenje (2.5) obezbeđuje da bude uspostavljno tačno p objekata kompanije A. Ograničenja (2.6), (2.7) i (2.8) definišu promenljive $x_j, j \in J, y_i, i \in I$ i $z_i, i \in I$ kao binarne.

Nakon rada [37] u kojem je MAXCAP uveden, usledila su dalja istraživanja i razvoj ove oblasti. To je dovelo do nastanka nekoliko varijanti osnovnog problema. U inicijalnoj varijanti iz [37], pretpostavka je da korisinici biraju uslužni objekat jedino na osnovu udaljenosti. Kako situacija na tržištu podleže raznim promenama, često je neophodno dodati nove uslove i pretpostavke koji će prilagoditi model realnim situacijama. Lokacije koje su u trenutku planiranja bile optimalne, ne moraju ostati takve, tako da može biti potrebno da se neki objekat ili više njih premesti. U radu [38] izložen je *problem maksimalnog zauzimanja sa relokacijom* (engl. *Maximum Capture Problem with Relocation*, MAXRELOC). U ovoj modifikaciji, dopušta se da kompanija A pored uspostavljanja novih objekata može i da relocira neke od već postojećih objekata. Zamenom ograničenja (2.5) sa

$$\sum_{j \in J} x_j = s + p \tag{2.9}$$

$$\sum_{j \in J_A} x_j = s - r \tag{2.10}$$

MAXCAP se jednostavno može preformulisati u MAXRELOC. Sa J_A je označen skup indeksa objekata komapanije A koji su već uspostavljeni i sa $s = |J_A|$ njihov broj. Parametar p određuje broj objekata koje A treba da uspostavi, dok je $r, r \leq s$ broj objekata koje treba da relocira.

Autori osnovnog MAXCAP modela su razmatrali i varijantu problema u kojoj se prodajne cene kompanija A i B razlikuju. Pri tome se pretpostavlja da su cene jednake u svim objektima jedne kompanije. Alokacije korisnika u tom slučaju osim od udaljenosti zavise i od cene. Odgovarajući model je poznat pod nazivom *problem maksimalnog zauzimanja sa cenama* (engl. *Pricing Maximum Capture Problem*, PMAXCAP) i predložen je u radu [43]. Cilj problema je odrediti optimalne lokacije i optimalnu fabričku cenu za kompaniju A koja ulazi na tržište, pri čemu se pretpostavlja neelastično ponašanje korisnika. U radu [43] je predložena i elastična varijanta problema maksimalnog zauzimanja sa cenama, PMAXMED, koja podrazumeva da se funkcija cilja formuliše koristeći pristup p-medijane. Generalno se pri rešavanju lokacijskih problema može se desiti da drugačiji vidovi ponašanja korisnika rezultuju različitim skupovima optimalnih lokacija. U [42] je posebna pažnja posvećena upravo značaju razmatranja korisničkog ponašanja. Autori rada [42] su izvršili poređenja nekoliko različitih modela pridruživanja korisnika merenjem odstu-

panja vrednosti funkcije cilja kada se uspostave lokacije optimalne za neodgovarajući model.

2.2 Postojeće metode rešavanja iz literature

U jednom od prvih radova koji se bavio problemom zauzimanja tržišta, ReVel je za rešavanje MAXCAP-a predložio egzaktnu proceduru zasnovanu na BnB metodi. Međutim, u [37] je ReVelova egzaktna procedura primenjena samo na relativno malim mrežama do 30 čvorova. Hakimi je prilikom razmatranja problema $(r|X_p)$ medijane u [14] dokazao da problemi zauzimanja tržišta, pa time i MAXCAP, spadaju u klasu $\mathcal{NP}$-teških problema. Kako većina algoritama kombinatorne optimizacije ima eksponencijalnu složenost, postalo je uobičajeno primeniti neku aproksimativnu metodu. *Algoritam za zauzimanje tržišta* (engl. *Algorithm for Market Capture*, AMACA) predložen u radu [44] predstavlja 1-optimalnu heurističku metodu namenjenu rešavanju osnovnog problema maksimalnog zauzimanja tržišta u slučaju većih dimenzija.

Ukoliko se postave određeni dodatni uslovi na MAXCAP, metode linearnog programiranja se ne mogu primeniti. Obe varijante problema maksimalnog zauzimanja sa cenama (PMAXMED i PMAXCAP) imaju nelinearnu funkciju cilja. Moguće je prvo rešiti osnovni MAXCAP i odrediti optimalne lokacije, a zatim izračunati optimalne cene. Ipak, potpuno razdvajanje u dve faze nije uvek opravdano, jer se uglavnom pretpostavlja da su optimalne lokacije i optimalne cene međusobno zavisne. U [43] je predložena hibridna heuristika sa fazom konstrukcije i dve faze popravke kojom se istovremeno određuju i lokacije i cene. Za rešavanje MAXCAP-a pri različitim načinima alokacije korisnika predloženih u [42] je konstruisana hibridizacija GRASP (engl. *Greedy Randomized Adaptive Search Procedures*) metode i tabu pretrage (engl. *Tabu Search*, TS).

2.3 Primer problema maksimalnog zauzimanja

Radi ilustracije, biće naveden primer MAXCAP-a male dimenzije. Neka je uvedena kraća notacija $w = \{w_1, ..., w_{|I|}\}$, $P = \{P_1, ..., P_{|I|}\}$ i $T = \{T_1, ..., T_{|I|}\}$. Broj korisnika, broj potencijalnih lokacija, broj objekata koje kompanija A treba da otvori i broj postojećih objekata kompanije B su redom jednaki:

$$|I| = 10,$$
$$|J| = 10,$$
$$p = 2,$$
$$q = 4.$$

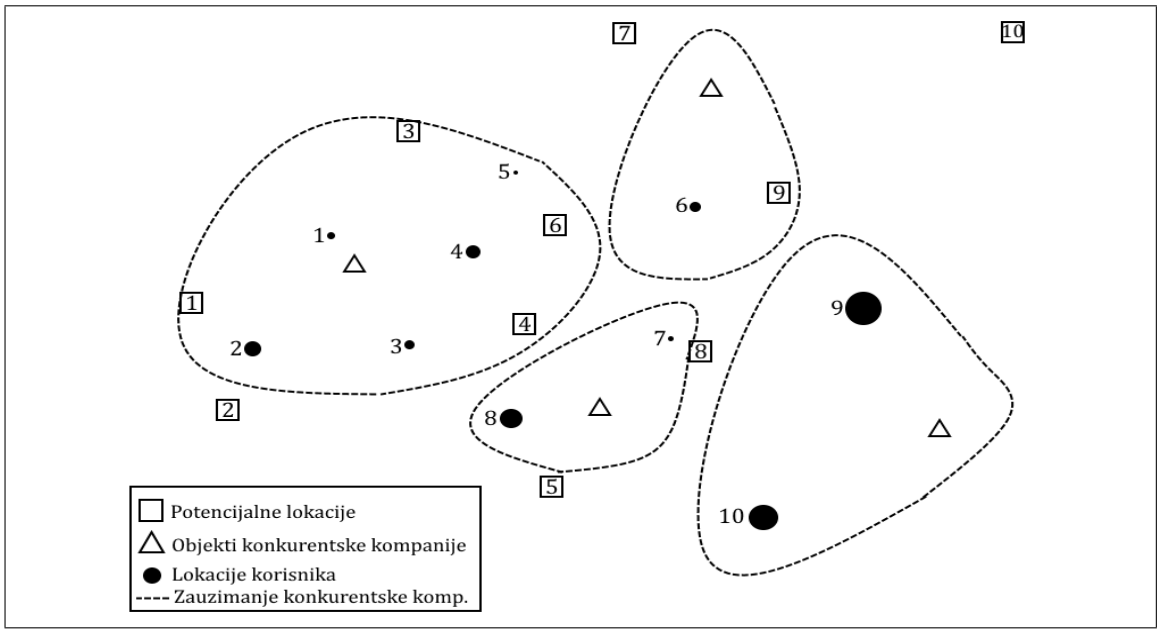
Niz w sadrži potražnje korisnika w_i za $i = 1, 2, ..., 10$ i njegovi elementi su:

$$w = \{5, 12, 7, 10, 3, 8, 2, 13, 20, 14\}.$$

Nizovi P i T, odnosno skupovi P_i i T_i koji sadrže indekse potencijalnih lokacija objekata sa kojih bi kompanija A mogla pridobiti celu ili delimičnu potražnju korisnika i, za svako $i = 1, 2, ..., 10$ su navedeni su redom u naredne dve kolone:

$P_1 = \{\}$ $T_1 = \{\}$
$P_2 = \{0, 1\}$ $T_2 = \{\}$
$P_3 = \{\}$ $T_3 = \{1, 3\}$
$P_4 = \{5\}$ $T_4 = \{3\}$
$P_5 = \{2, 5, 6\}$ $T_5 = \{3\}$
$P_6 = \{8\}$ $T_6 = \{5\}$
$P_7 = \{7\}$ $T_7 = \{\}$
$P_8 = \{4\}$ $T_8 = \{\}$
$P_9 = \{\}$ $T_9 = \{2, 7, 8\}$
$P_{10} = \{\}$ $T_{10} = \{1, 7\}$

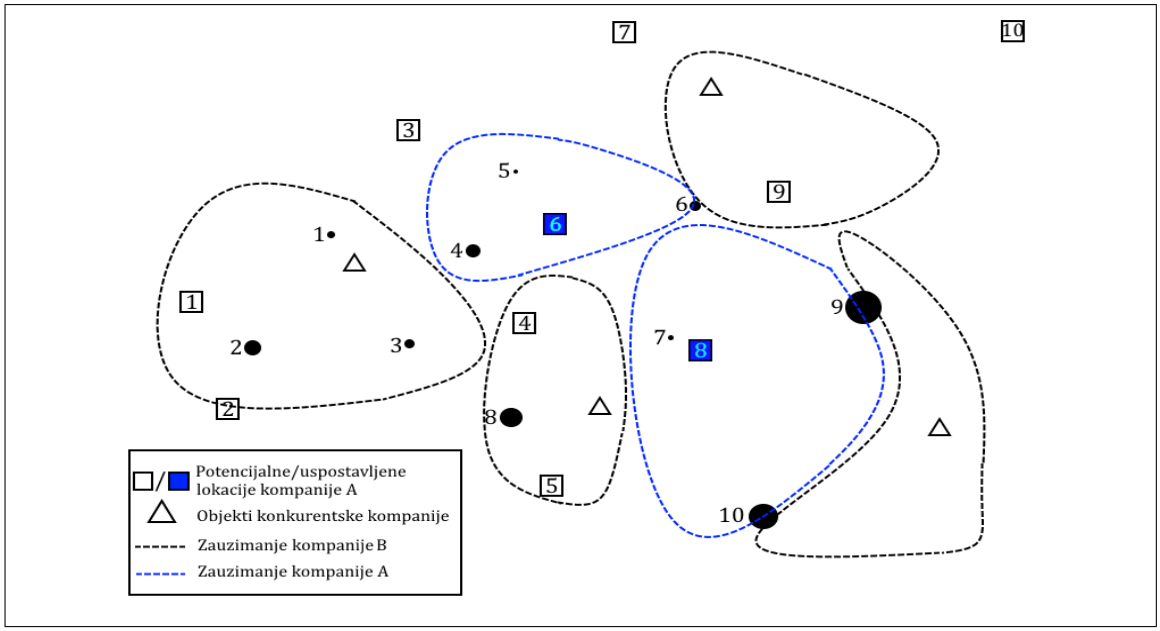
Na slici 2.1 prikazana je inicijalna situacija kada monopolistička kompanija B zauzima kompletno tržište. Kompanija A tek treba da uđe na tržište tako što treba da uspostavi dva uslužna objekta na nekim od deset potencijalnih lokacija. Veličina simbola koji predstavlja korisnički objekat odgovara količini potražnje tog korisnika.



Slika 2.1: Raspodela tržišta pre ulaska kompanije A

Na slici 2.2 je predstavljena situacija na tržištu koja odgovara optimalnom rešenju. Kompanija A je uspostavila dva uslužna objekta na lokacijama 6 i 8 ($x_6 = 1$ i $x_8 = 1$). Takvim pozicioniranjem kompanija A je kompletno prisvojila dva korisnika na lokacijama 4 i 5 ($y_4 = 1$ i $y_5 = 1$). Potražnje tri korisnika na lokacijama 6,

9 i 10 su ravnopravno podeljene između kompanija A i B ($z_6 = 1, z_9 = 1$ i $z_{10} = 1$). Vrednosti preostalih promenljivih x_j, y_i i z_i su jednake nuli. Vrednost funkcije cilja optimalnog rešenja je u ovom slučaju jednaka 36.



Slika 2.2: Raspodela tržišta posle ulaska kompanije A

3
Heurističke metode za rešavanje problema maksimalnog zauzimanja tržišta

U raznim sferama života nailazi se na situacije i procese koji se mogu formulisati kao problemi kombinatorne optimizacije. U oblastima poput ekonomije i industrije često se javlja potreba za maksimizacijom profita, kvaliteta, efikasnosti ili minimizacijom troškova, rizika, utrošenog vremena. Značaj dobijanja odgovora na takva pitanja razumljivo je od velike važnosti. Međutim, veliki broj problema kombinatorne optimizacije spada u klasu $\mathcal{NP}$ kompletnih problema, za koje još uvek nije ustanovljeno da li ih je moguće (tačno) rešiti u polinomijalnom vremenu ili ne. Čak i kada su razvijene egzaktne metode za njihovo rešavanje, one su uglavnom eksponencijalne složenosti. Zbog toga su egzaktne metode praktično neupotrebljive za ulazne podatke velikih dimenzija koje se često javljaju u realnim situacijama. Iz tog razloga se pribegava konstruisanju aproksimativne metode i prihvata se rešenje "dovoljno" blisko optimalnom. Između ostalog, u aproksimativne metode spadaju heuristike i metaheuristike. Dok su heuristike specifični algoritmi konstruisani za konkretan problem, metaheuristike su znatno opštije. One se mogu primeniti na širi opseg problema i predstavljaju neku vrstu shema iz kojih se izvode heuristike prilagođene primeni [50].

Može se reći da ljudski način reznovanja u svakodnevnom životu više odgovara metaheurističkom konceptu [45]. U praktičnim situacijama često je prihvatljivije brzo pronaći približno rešenje nego utrošiti previše vremena na nalaženje optimalnog rešenja. Formalno izučavanje heuristika i metaheuristika je započeto relativno skoro, sredinom XX veka [45]. Do sada su razvijene brojne različite metode, neretko inspirisane prirodnim ili fizičkim procesima. Klasifikacija metaheuristika može biti zasnovana na nekoliko kriterijuma. Jedna od osnovnih načina na koji se podela može izvršiti jeste prema broju rešenja koji se posmatra [50]. *S-metaheuristike* (engl. *Single-solution*) podrazumevaju da se konstruiše i/ili popravlja jedinstveno rešenje. Sa druge strane, *P-metaheuristike* (engl. *Population-based*) podrazumevaju da se istovremeno radi na skupu rešenja koji se uobičajeno naziva populacija rešenja.

U narednim podsekcijama su opisane heurističke metode prilagođene za MAX-CAP. Izložena su četiri pristupa: dve P-metaheuristike (optimizacija kolonijom pčela i genetski algoritam), jedna S-metaheuristika (metoda promenljivih okolina) i jedan predlog hibridizacije optimizacije kolonijom pčela i metode promenljivih okolina.

3.1 Optimizacija kolonijom pčela

Optimizacija kolonijom pčela (engl. *Bee Colony Optimization*, BCO) je stohastička P-metaheuristika i inspirisana je ponašanjem pčela u potrazi za hranom. Spada u grupu metoda koje koriste *inteligenciju roja* (engl. *Swarm Intelligence*, SI). BCO su predložili Lučić i Teodorović za rešavanje transportnog problema [26].

Ideja koja leži u osnovi BCO metode je imitacija ponašanja kolonije prirodnih medonosnih pčela. Pčele u koloniji mogu imati različite uloge [50, p. 255]. U svakoj koloniji su za reprodukciju zaduženi jedna kraljica i par hiljada trutova. Sve ostale poslove u košnici obavljaju pčele radilice, kojih leti može biti i po osamdeset hiljada. Radilice su zadužene da hrane i neguju leglo i maticu, sakupljaju nektar, cvetni prah, propolis, grade saće, brinu se o temperaturi i čistoći košnice, brane košnicu, proizvode med... Za potrebe konstruisanja BCO metode posmatrano je ponašanje pčela koje skupljaju hranu (engl. *foraging bees*). Pčele izviđači napuštaju košnicu kako bi istražile njenu okolinu i pronašle lokacije nektara. Po povratku, izviđači izvode specifičan ples (engl. *waggle dance*) kojim komuniciraju sa preostalim pčelama. Ples služi da precizno predstave udaljenost lokacije, kvalitet i količinu pronađenog nektara. Neopredeljene pčele iz košnice se tada mogu odlučiti da krenu na neku od reklamiranih lokacija kako bi prikupile nektar. Poput izviđača, pčele koje donose nektar sa neke izuzetno dobre lokacije mogu u košnici izvesti ples i pokušati da navedu druge pčele da ih prate. Pčele mogu i napustiti lokaciju koja iz nekog razloga nije povoljna i ponovo postati neopredeljene.

Prvi korak u primeni tih principa je formiranje kolonije veštačkih pčela. Svakoj pčeli se dodeljuje inicijalno rešenje. Prvobitna varijanta BCO metode predložena u [26] je kontruktivna i podrazumeva prazno početno rešenje koje se iterativno dograđuje. Eksprimentalni rezultati studije [6] dobijeni primenom konstruktivne BCO metode na poznati lokacijski problem p-centra nisu bili zadovoljavajući. Iz tog razloga, u [6] je konstruisana *optimizacija kolonijom pčela sa popravkom* (engl. *Improving Bee Colony Optimization*, BCOi) namenjena upravo rešavanju problema p-centra. BCOi metoda je zasnovana na ideji da se pri inicijalizaciji generišu već gotova rešenja, koja se potom određenim izmenama popravljaju u cilju pronalaska optimalnog rešenja.

Većina osnovnih koraka BCO i BCOi metoda se poklapa. Glavna razlika ove dve varijante je što se u BCO posmatraju parcijalna, a u BCOi kompletna rešenja. U svakoj iteraciji BCO, odnosno BCOi algoritma, svaka pčela izvodi *let unapred* (engl. *forward pass*) i *let unazad* (engl. *backward pass*). U zavisnosti od toga da li je početno rešenje prazno ili kompletno, pčela tokom leta unapred pokušava da izgradi, odnosno da popravi svoje rešenje. Prilikom leta unazad pčela može da napusti trenutno (loše) rešenje i postane neopredeljena (engl. *uncommitted*). U suprotnom, kaže se da je pčela ostala lojalna (engl. *loyal*) trenutnom rešenju i dobija ulogu regrutera (engl. *recruiter*).

Parametri metode su broj pčela B i broj koraka tokom leta unapred NC (broj koraka između razmene informacija među pčelama). U konstruktivnoj varijanti BCO metode, NC je obično broj koraka koji je potreban da se kompletira rešenje. Kriterijum zaustavljanja može biti ukupan broj koraka, maksimalno procesorsko vreme izvršavanja, dovoljan broj poklapanja rešenja ili kombinacija nekoliko kriterijuma. Bitnu ulogu u realizaciji BCO metode ima *ruletska selekcija* (engl. *roulette wheel selection*, RWS). Osnovna ideja ruletske selekcije je da verovatnoća izbora nekog

elementa iz skupa proporcionalno zavisi od kvaliteta u odnosu na ostale članove tog skupa. Recimo da treba slučajno izabrati jedan element iz nekog konačnog skupa $S = \{a_1, a_2, ..., a_n\}$, pri čemu svako a_i ima kvalitet v_i, $i = 1, ..., n$. Tada će pri ruletskoj selekciji, element a_i biti izabran sa verovatnoćom $p_i = \frac{v_i}{\sum_{i=1}^{n} v_i}$. U slučaju BCO metode, ako su a_i pčele, v_i su najčešće vrednosti funkcije cilja rešenja dodeljenog datoj pčeli.

Algoritam 3.1 prikazuje osnovnu strukturu obe varijante BCO metode. Neki od koraka, kao što su inicijalizacija, procena narednog koraka i evaluacija rešenja se moraju prilagoditi konkretnom problemu.

Algoritam 3.1: Osnovna struktura BCO i BCOi metoda

Ulaz: *Parametri B, NC i kriterijum zaustavljanja*;
Inicijalizacija: dodeliti (prazno) kompletno početno rešenje svakoj pčeli;
while *nije ispunjen kriterijum zaustavljanja* **do**
  `// let unapred`
  for *b = 1* **to** *B* **do**
    for *n = 1* **to** *NC* **do**
      Proceniti korake i pravilom ruletske selekcije izabrati sledeći;
    end
  end
  `// let unazad`
  for *b = 1* **to** *B* **do**
    Proceniti kvalitet (parcijalnog) rešenja pčele *b*;
  end
  for *b = 1* **to** *B* **do**
    Odrediti da li je pčela *b* lojalna ili ne;
  end
  for *b = 1* **to** *B* **do**
    if *pčela b je neopredeljena* **then**
      Pravilom ruletske selekcije izabrati regrutera
      i preuzeti njegovo rešenje;
    end
  end
  Proceniti rešenja svih pčela i izabrati najbolje;
end
return najbolje pronađeno rešenje;

Koraci određivanja lojalnosti i izbora regrutera se vrše na unapred definisan način, odnosno koristeći adekvatne matematičke formule. Najpre je neophodno odrediti normalizovanu vrednost funkcije cilja o_b. Neka je sa f_b označena vrednost funkcije cilja koja odgovara trenutnom rešenju pčele b. Neka nakon svake iteracije algoritma, f_{min} i f_{max} redom označavaju minimalnu i maksimalnu vrednost funkcije cilja svih pčela u toj iteraciji. Za optimizacioni problem tipa maksimizacije se za normalizaciju koristi formula (3.1).

$$o_b = \begin{cases} \frac{f_b - f_{max}}{f_{max} - f_{min}}, & \text{ako je } f_{max} \neq f_{min}, \\ 1, & \text{ako je } f_{max} = f_{min}. \end{cases} \tag{3.1}$$

Neka je sa o_{max} označena najveća od normalizovanih vrednosti funkcije cilja.

Verovatnoća sa kojom pčela b ostaje lojalna svom rešenju nakon što je izvršila u koraka unapred određuje se po formuli:

$$p_b^{u+1} = e^{\frac{o_b - o_{max}}{u}}, \tag{3.2}$$

kojom se obezbeđuje da verovatnoća lojanosti raste sa brojem izvršenih letova unapred.

Za svaku lojalnu pčelu treba izračunati verovatnoću sa kojom ona regrutuje neopredeljene pčele. Neka je $R \subset B, R \neq \emptyset$ skup svih lojalnih pčela. Svaka od neopredeljenih pčela bira po jednog regrutera pravilom ruletske selekcije i preuzima njegovo rešenje. Pčela $r \in R$ biće izabrana sa verovatnoćom p_r koja se računa po formuli:

$$p_r = \frac{o_r}{\sum_{b \in R} o_b}. \tag{3.3}$$

Kako posle procesa regrutacije postoje pčele koje imaju isto rešenje, neophodno je da pčele izvode let unapred nezavisno jedna od druge da bi se postigla diverzifikacija. U suprotnom, može doći do preuranjene konvergencije u lokalni minimum.

Formule (3.1), (3.2) i (3.3) su preuzete su iz rada [41] gde se mogu videti i alternativni načini da se izračunaju tražene verovatnoće.

3.1.1 Implementacija BCO metode i primena na MAXCAP

Za rešavanje MAXCAP-a je u ovom radu prilagođena varijanta BCO metode sa popravkom. Pojedinačni koraci predložene BCOi metode su implementirani na sledeći način.

1. **Predprocesiranje**
   Pri učitavanju podataka, vrše se izračunavanja koja pomažu u realizaciji koraka BCOi metode. Za svakog korisnika $i \in I$, čuvaju se tri parametra w_i, P_i i T_i. Vrednost parametra w_i odgovara potražnji korisnika i. Skupovi P_i i T_i sadrže indekse potencijalnih lokacija sa kojih bi kompanija A potpuno ili delimično opsluživala korisnika i. Za svaku potencijalnu lokaciju $j \in J$, pamte se skupovi $\tilde{P}_j$ i $\tilde{T}_j$. Ova dva skupa sadrže indekse korisnika čiju bi potražnju kompanija A potpuno ili polovično prisvojila od kompanije B uspostavljanjem objekta na lokaciji j.

2. **Reprezentacija rešenja**
   Tokom rada BCO metode, svaka veštačka pčela radi na popravci jednog kompletnog rešenja. Svi koraci algoritma su implementirani tako da se posmatraju jedino dopustiva rešenja. U matematičkoj formulaciji MAXCAP-a uvedena su tri skupa binarnih promenljivih koje određuju rešenje problema, a koje se u BCO implementaciji kodiraju na sledeći način. Za predstavljanje uspostavljenih lokacija se koristi binarni vektor $x = (x_1, ..., x_{|J|})$. Prema uvedenoj notaciji, važi da je $x_j = 1$ ako i samo ako kompanija A uspostavlja objekat na lokaciji j. Vrednosti elemenata vektora $y = (y_1, ..., y_{|I|})$ i $z = (z_1, ..., z_{|I|})$ odgovaraju vrednostima promenljivih y_i i z_i iz definicije problema. Važi da je $y_i = 1$ ako i samo ako je kompanija A prisvojila potražnju korisnika i i $z_i = 1$ ako i samo ako kompanije A i B dele potražnju korisnika i. Iako se

vrednosti elemenata vektora y i z mogu izračunati na osnovu x i skupova P_i i T_i, dodatnim kodiranjem ova dva vektora znatno se smanjuje broj potrebnih računskih operacija. Radi postizanja još veće efikasnosti, osim kodiranja vrednosti promeljivih iz definicije problema ažurira se i nekoliko pomoćnih skupova. Lokacije $j \in J$ se dele na skup uspostavljenih lokacija J_U i skup potencijalnih lokacija J_P. Korisnici $i \in I$ se dele na tri skupa I_A, I_{AB} i I_B u zavisnosti od vlasnika objekta kom su trenutno alocirani. Ovi skupovi su definisani na sledeći način: $I_A = \{i \in I | y_i = 1, z_i = 0\}$, $I_{AB} = \{i \in I | y_i = 0, z_i = 1\}$ i $I_B = \{i \in I | y_i = 0, z_i = 0\}$.

3. **Generisanje početnih rešenja**
   Za svaku pčelu se na početku rada metode formira jedno kompletno inicijalno rešenje. Radi bolje pretrage prostora rešenja, poželjno je obezbediti da svaka pčela krene od drugačijeg skupa uspostavljenih lokacija. Kako bi početna rešenja bila boljeg kvaliteta, inicijalnih p uspostavljenih lokacija se bira tako da se što više korisnika preuzme od kompanije B. U tu svrhu, prvo se bira korsinik $i \in I_B$ koga opslužuje kompanija B. Zatim se u skup uspostavljenih lokacija dodaje slučajno izabrana lokacija $j \in P_j$ sa koje se on može preuzeti od konkurenta. Ukoliko je $P_j = \emptyset$, lokacija se bira iz T_j, a ukoliko je i $T_j = \emptyset$, bira se proizvoljna lokacija $j \in J_P$ koja jošuvek nije uspostavljena. Korak se ponavlja p puta, čime se dobija korektno rešenje koje ispunjava sve uslove problema. Pri realizaciji izbora indeksa i i j se koristi generator slučajnih brojeva sa uniformnom raspodelom.

4. **Evaluacija rešenja**
   Kvalitet pronađenog rešenja je bolji ukoliko mu odgovara veća vrednost funkcije cilja (2.1). Sa svakom izmenom vrednosti vektora x, vrednost funkcije cilja se ažurira. Pri tome se u velikoj meri koriste pomoćni skupovi definisani u koraku predprocesiranja. Pri dodavanju uslužnog objekta j, posmatraju se samo korisnici $i \in \tilde{P}_j$ koje taj objekat može opslužiti potpuno, odnosno $i \in \tilde{T}_j$ koje može opslužiti delimično. Na tekuću vrednost funkcije cilja se dodaje w_i, $\frac{w_i}{2}$ ili 0, u zavisnosti od toga da li i trenutno pripada skupu I_A, I_{AB} ili I_B. Pomoćni skupovi i vrednosti vektora y i z se uporedo ažuriraju. Analogno, pri uklanjanju objekta j, od tekuće vrednosti funkcije cilja treba oduzeti potražnju onih korisnika koje sem j ne opslužuje nijedan drugi objekat kompanije A. Izračunata vrednost funkcije cilja se normalizuje po formuli (3.1).

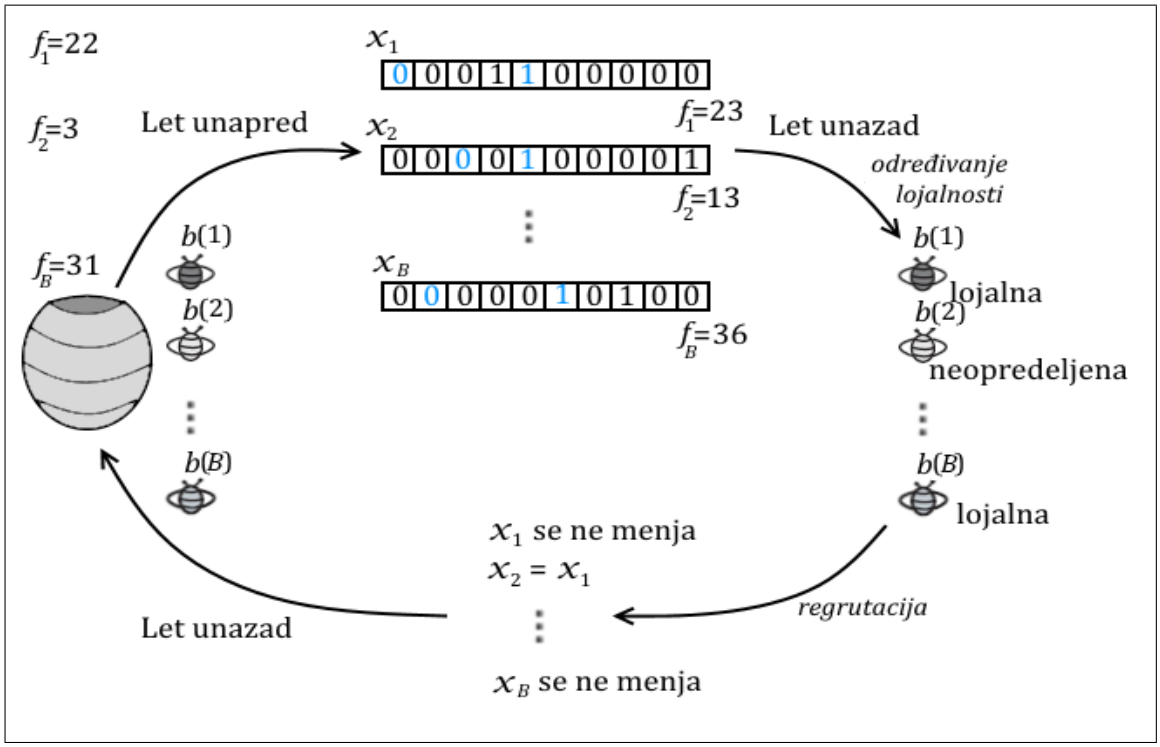
5. **Let unapred**
   Ovo je najbitiniji korak za postizanje boljeg kvaliteta rešenja. Pri implementaciji je iskorišćen stohastički pristup koji obezbeđuje različit tretman istog rešenja od strane različitih pčela, a i efikasno izvršavanje samog koraka. Modifikacije rešenja u okviru leta unapred su implementirane tako da se ne naruši njegova dopustivost. Prvo se bira slučajan korisnik i kog opslužuje kompanija B ($i \in I_B$). Ukoliko takav ne postoji, bira se korisnik koga opslužuju obe kompanije ($i \in I_{AB}$). Zatim se bira potencijalna lokacija j sa koje se potražnja korisnika i može prisvojiti. Indeks j se bira na slučajan način iz skupa P_i, odnosno T_i ako je P_i prazan. Ukoliko je i skup T_i prazan, bira se proizvoljna slobodna lokacija $j \in J_P$. Postavljanjem vrednosti elementa x_j na jedan se narušava se uslov (2.4). Zbog toga je potrebno izbaciti jedan uspostavljeni

objekat, koji se bira iz skupa J_U na slučajan način.

6. **Let unazad**
   Let unazad se sastoji iz tri faze: evaluacije rešenja, određivanja lojalnosti pčela i procesa regrutacije neopredeljenih pčela. Verovatnoća lojalnosti se određuje po formuli (3.2). Nakon toga, verovatnoća regrutovanja za lojalne pčele se računa po formuli 3.3. Svaka neopredeljena pčela preuzima rešenje jednog regrutera koji se bira po prinicipu ruletske selekcije. U toku realizacije leta unazad se za generisanje slučajnih brojeva iz opsega $[0, 1]$ koristi uniformna raspodela.

7. **Kriterijum zaustavljanja**
   Posle svake iteracije BCOi algoritma, neophodno je ažurirati globalno najbolje rešenje x_{best} i odgovarajuću vrednost funkcije cilja f_{best}. Kriterijum zaustavljanja je maksimalan broj iteracija bez pronalaska rešenja boljeg od x_{best}, što je određeno parametrom $MaxIter$. Po ispunjenju kriterijuma zaustavljanja, najbolje pronađeno rešenje se prijavljuje korisniku.

Na slici 3.1 je predstavljena ilustracija rada implementirane BCOi metode na primeru 2.3. Neka su trenutna rešenja dodeljena pčelama $b(1)$, $b(2)$ i $b(B)$ redom $x_1 = \{1, 0, 0, 1, 0, 0, 0, 0, 0, 0\}$, $x_2 = \{0, 0, 1, 0, 0, 0, 0, 0, 0, 1\}$ i $x_B = \{0, 1, 0, 0, 0, 0, 0, 1, 0, 0\}$. Odgovarajuće vrednosti funkcije cilja su $f_1 = 22$, $f_2 = 3$ i $f_B = 31$. Tokom leta unapred, prva pčela bira korisnika $8 \in I_B^{(1)}$. Nakon toga se uspostavlja lokacija $5 \in P_8^{(1)}$, a izbacuje objekat $1 \in J_U^{(1)}$. Vrednost funkcije cilja koja odgovara novom rešenju pčele $b(1)$ je 23. Analognim postupkom se modifikuju ostala rešenja. Tokom leta unazad, pčela $b(B)$ sigurno ostaje lojalna jer je $f_B = 36 = f_{max}$. Pčela $b(2)$ napušta svoje rešenje pa joj se u procesu regrutacije pridružuje rešenje x_1 izabrano primenom ruletske selekcije.



Slika 3.1: Ilustracija BCOi metode na primeru

3.2 Metoda promenljivih okolina

Metoda promenljivih okolina (engl. *Variable Neighborhood Search*, VNS) je S-metaheuristika bazirana na lokalnom pretraživanju (engl. *Local Search*, LS). Radi postizanja veće efikasnosti, okoline trenutnog rešenja se sistematski menjaju. Promena okoline se može realizovati uvođenjem različitih veličina okolina (odnosno broja istovremenih transformacija nad rešenjem) ili različitih metrika (odnosno načina transformacije rešenja). VNS metaheuristika je prvobitno predložena za primenu na problem trgovačkog putnika u radu [30] koji su 1997. godine objavili Pjer Hansen i Nenad Mladenović. Prema [30], VNS metoda je zasnovana na sledeće tri činjenice, a koje se mogu koristiti deterministički, stohastički ili kombinovano:

1. Lokalni optimum u odnosu na jednu okolinu ne mora biti optimum u odnosu na neku drugu okolinu;

2. Lokalni optimum ne mora biti i globalni optimum;

3. Za većinu problema, lokalni optimumi su relativno blizu u odnosu na razne okoline.

Metoda promenljivog spusta (engl. *Variable Neighborhood Descent*, VND) primenjuje deterministički pristup. VND se sastoji u izboru k_{max} sistematski uređenih okolina $N_1, N_2, ..., N_{k_{max}}$. U svakoj od okolina se startuje lokalna pretrage u odnosu na neko početno rešenje x. U slučaju pronalaska boljeg rešenja, lokalna pretraga se pokreće u odnosu na njega ponovo iz prve okoline. U suprotnom se razmatra naredna okolina. Algoritam 3.2 prikazuje osnovne korake VND metaheuristike.

Algoritam 3.2: Osnovna struktura VND metode

Ulaz: *Niz okolina* $N_1, N_2, ..., N_{k_{max}}$;
Inicijalizacija: izabrati početno rešenje x, $x_{best} = x, k = 1$;
while $k \leq k_{max}$ **do**
    `// faza lokalne pretrage`
    Lokalnom pretragom okoline N_k tekućeg rešenja x odrediti x';
    if x' *bolje od* x_{best} **then**
        $x_{best} = x'$;
        $k = 1$;
    else
        $k = k + 1$;
    end
end
return x_{best};

Redukovana metoda promenljivih okolina (engl. *Reduced Variable Neighborhood Search*, RVNS) je stohastička varijanta. Kao i VND, sastoji se u sistematskom kretanju kroz uređeni niz okolina $N_1, N_2, ..., N_{k_{max}}$ nekog izabranog rešenja x. Međutim, RVNS ne uključuje lokalnu pretragu već se u svakoj iteraciji primenjuje *razmrdavanje* (engl. *shaking*), odnosno iz tekuće okoline se slučajno bira jedno rešenje. Ukoliko je izabrano rešenje bolje od trenutnog, postupak se ponavlja počev od prve okoline novog rešenja, a inače se prelazi u sledeću okolinu. Osnovna struktura RVNS metode

je predstavljena algoritmom 3.3.

Algoritam 3.3: Osnovna struktura RVNS metode

Ulaz: *Niz okolina* $N_1, N_2, ..., N_{k_{max}}$;
Inicijalizacija: izabrati početno rešenje x, $x_{best} = x, k = 1$;
while $k \leq k_{max}$ **do**
    `// faza razmrdavanja`
    Izabrati slučajno rešenje x' u okolini N_k tekućeg rešenja x;
    if x' *bolje od* x_{best} **then**
        $x_{best} = x'$;
        $k = 1$;
    else
        $k = k + 1$;
    end
end
return x_{best};

RVNS može biti jako korisna za probleme velikih dimenzija, jer ne uključuje lokalnu pretragu koja je često složena i dugotrajna. Međutim, najbolji kvalitet rešenja se postižu kombinovanjem stohastičkog i determinističkog pristupa. *Osnovna metoda promenljivih okolina* (engl. *Basic Variable Neighborhood Search*, VNS) je najčešće korišćena varijanta koja predstavlja kombinaciju VND i RVNS metoda. Osnovna VNS metoda se sastoji iz sledećih faza:

- Pronalaženje početnog rešenja;
- Lokalna popravka početnog rešenja;
- Razmrdavanje;
- Lokalna pretraga;
- Pomeranje.

Prilikom faze lokalne pretrage su moguća na dva pristupa. Princip *prvog poboljšanja* (engl. *first improvement*, FI) podrazumeva prekid pretrage nakon pronalaska prvog boljeg rešenja u okolini. Princip *najboljeg poboljšanja* (engl. *best improvement*, BI) podrazumeva detaljnu pretragu cele okoline kako bi se pronašlo najbolje rešenje. Definicije okolina za fazu razmrdavanje i okolina za fazu lokalne pretrage se mogu, ali ne moraju poklapati. Struktura osnovne metode je predstavljena algoritmom 3.4.

Kriterijum zaustavljanja može biti maksimalan broj iteracija, maksimalan broj iteracija između dva poboljšanja, maksimalno utrošeno procesorsko vreme ili neka kombinacija više kriterijuma.

U literaturi su do sada predložene i brojne druge varijante i proširenja metode promenljivih okolina, kao što su: *metoda promenljivih okolina sa dekompozicijom* (engl. *Variable Neighbourhood Decomposition Search*, VNDS), *adaptivna metoda promenljivih okolina* (engl. *Skewed Variable Neighborhood Search*, SVNS), *metoda promenljivih formulacija* (engl. *Variable Neighborhood Formulation Space Search*, VNFSS), Primal-dual VNS itd. Primeri uspešne primene raznih varijanti VNS metoda za rešavanje diskretnih lokacijskih problema su prikazani u radovima [7], [15–19], [22], [29], [31], [32] itd.

Algoritam 3.4: Osnovna struktura VNS metode

Ulaz: *Niz okolina* $N_1, N_2, ..., N_{k_{max}}$;
Inicijalizacija: izabrati početno rešenje x, $x_{best} = x, k = 1$;
while $k \leq k_{max}$ **do**
    `// faza razmrdavanja`
    Izabrati slučajno rešenje x' u okolini N_k tekućeg rešenja x;
    `// faza lokalne pretrage`
    Lokalnom pretragom okoline rešenja x' odrediti x'';
    if x'' *bolje od* x_{best} **then**
        $x_{best} = x''$;
        $k = 1$;
    else
        $k = k + 1$;
    end
end
return x_{best};

3.2.1 Implementacija VNS metode i primena na MAXCAP

VNS implementacija za MAXCAP ima sledeće elemente.

1. **Reprezentacija rešenja**
   Za predstavljanje uspostavljenih lokacija koristi se vektor binarnih vrednosti $x = (x_1, x_2, ..., x_{|J|})$. Važi da je vrednost elementa x_j jednaka 1 ako je objekat kompanije A na poziciji j uspostavljen, a 0 inače. Pritom se kodiraju samo dopustiva rešenja, odnosno vektor x ispunjava uslov (2.5). Binarne promenljive y_i i z_i iz definicije problema koje predstavljaju alokacije korisnika se ne kodiraju direktno. U cilju postizanja veće efikasnosti, uvodi se notacija $\tilde{y_i} = |\{j \in P_i | x_j = 1\}|$ i $\tilde{z_i} = |\{j \in T_i | x_j = 1\}|$, pri čemu važi da je $y_i = \text{sgn}(\tilde{y_i})$ i $z_i = (1 - \text{sgn}(\tilde{y_i})) \cdot \text{sgn}(\tilde{z_i})$. Vrednosti promenljivih $\tilde{y}$ i $\tilde{z}$ su predstavljene elementima celobrojnih vektora $\tilde{y}$ i $\tilde{z}$.

2. **Generisanje početnog rešenja**
   Početno rešenje se generiše kroz dve faze. Najpre se konstruiše vektor $\tilde{x}$ čijih je prvih p elemenata postavljeno na 1, a preostalih $|J| - p$ elemenata je postavljeno na 0. Početno rešenje se generiše permutovanjem vektora $\tilde{x}$ primenom ugrađene funkcije programskog jezika C++ `random_shuffle`. Ovim postupkom se postiže slučajan izbor dopustivog inicijalnog rešenja x. Vrednosti odgovarajućih vektora $\tilde{y}$ i $\tilde{z}$ se računaju na osnovu x i skupova P i T.

3. **Izračunavanje funkcije cilja**
   Nakon generisanja početnog rešenja, inicijalna vrednost funkcije cilja f se umesto po formuli (2.1), računa korišćenjem vrednosti promenljivih $\tilde{y}$, $\tilde{z}$ i formule (3.4).

$$\max \sum_{i \in I} w_i \cdot \text{sgn}(\tilde{y_i}) + \sum_{i \in I} \frac{w_i}{2} \cdot (1 - \text{sgn}(\tilde{y_i})) \cdot \text{sgn}(\tilde{z_i}) \tag{3.4}$$

4. **Definicija okolina**
   Okoline koje se koriste u fazi razmrdavanja su uređene u rastući niz $N_1, ..., N_{k_{max}}$. Za dati skup uspostavljenih lokacija predstavljen nizom x, rešenje x' koje pripada okolini N_k rešenja x se dobija zamenom k uspostavljenih lokacija, $k = 1, 2, ...k_{max}$. S obzirom da je neophodno uspostaviti tačno p objekata kompanije A, dovoljno je razmatrati okoline N_k veličine $k \leq \min\{N_{k_{max}}, p, |J| - p\}$. U fazi lokalne pretrage se koristi okolina N_1, odnosno vrši se izmena jedne uspostavljene lokacije.

5. **Faza lokalne pretrage**
   Izmena jedne uspostavljene lokacije se vrši invertovanjem vrednosti dva elementa niza x. Iz tog razloga se faza lokalne pretrage može shvatiti kao specijalan slučaj 2-optimalne pretrage. Tokom LS faze, primenjen je pristup najboljeg poboljšanja, pri čemu je u implementaciji ostavljena mogućnost jednostavnog prelaska na pristup prvog poboljšanja prosleđivanjem odgovarajuće vrednosti logičkog parametra BI. Ukoliko parametar BI ima vrednost `true` tada se primenjuje pristup najboljeg poboljšanja, a inače se primenjuje pristup prvog poboljšanja. Algoritam 3.5 prikazuje pseudokod lokalne pretrage u okviru funkcije koja vraća logičku vrednost `true` ukoliko je u okolini N_1 tekućeg rešenja uspešno pronađeno bolje rešenja, a logičku vrednost `false` inače.

Algoritam 3.5: Faza lokalne pretrage

Ulaz: *Parametar BI (podrazumevano* `true`*)*;
Izlaz: `true` *ukoliko postoji bolje rešenje,* `false` *inače*;
`//` `Neka je` $J_0 = \{j \in J | x_j = 0\}$ `i` $J_1 = \{j \in J | x_j = 1\}$
$max = 0$; `// maksimalno poboljšanje`
for *svaki par indeksa* $i \in J_1$ *i* $j \in J_0$ **do**
    $x' = x$;
    $x'[i] = 0$;
    $x'[j] = 1$;
    Izračunati razliku vrednosti funkcije cilja d za x i x';
    if $d > max$ **then**
        $max = d$;
        $x'_{max} = x'$;
        if *bestImprovement* = `false` **then**
            break;
        end
    end
end
if $max > 0$ **then** `// postoji bar jedno bolje rešenje`
    $x = x'_{max}$;
    return `true`;
else
    return `false`;
end

6. **Faza razmrdavanja**
   U fazi razmrdavanja se koristi niz od k_{max} ugnježdenih okolina $N_1, ..., N_{k_{max}}$. Za fiksirano $k \in \{1, 2, ..., k_{max}\}$, razmrdavanje rešenja u okolini N_k se vrši slučajnim izborom k uspostavljenih i k slobodnih lokacija. Odgovarajući elementi vektora x se potom invertuju. Algoritam 3.6 prikazuje implementaciju faze razmrdavanja u okolini N_k. Za realizaciju ovog koraka se koristi funkcija `random_shuffle` kojom se permutuje niz indeksa uspostavljenih lokacija J_1 i niz indeksa slobodnih lokacija J_0. Nakon toga se, ne gubeći na opštosti, za invertovanje može izabrati prvih k elemenata svakog od pomenutih nizova.

Algoritam 3.6: Faza razmrdavanja

Ulaz: *Veličina okoline* k;
// Neka je $J_0 = \{j \in J | x_j = 0\}$ i $J_1 = \{j \in J | x_j = 1\}$
Primeniti `random_shuffle` na J_0 i J_1;
for $i = 1$ **to** k **do**
    $x[J_1[i]] = 0$;
    $x[J_0[i]] = 1$;
end

7. **Procedura za ubrzavanje faze lokalne pretrage**
   Na primerima velikih dimenzija, lokalna pretrage može zahtevati veliki broj izračunavanja vrednosti funkcije cilja. Kako se primenjuje BI pristup, tokom LS faze se pretražuje cela okolina N_1 veličine $\mathcal{O}(|J|)$. Zbog toga se, umesto računanja vrednosti funkcije cilja po formuli (3.4) (odnosno (2.1)) koristi drugi pristup. Na početku rada metode se za svaku potencijalnu lokaciju j odrede skupovi $\tilde{P}_j$ i $\tilde{T}_j$. Skup $\tilde{P}_j$ sarži indekse korisnika koji mogu prisvojiti uspostavljenjem objekta na lokaciji j, a $\tilde{T}_j$ sadrži indekse korisnika čija se potražnja uspostavljenjem objekta na lokaciji j može deliti sa kompanijom B. Za dati skup uspostavljenih objekata (koji je predstavljen nizom x), može se odrediti razlika u vrednosti funkcije cilja rešenja koje bi se dobilo ukoliko se uspostavljeni objekat na lokaciji i relocira na j. Vrednost te razlike d se korišćenjem vrednosti $\tilde{y}, \tilde{z}$ i uvedenih skupova $\tilde{P}$ i $\tilde{T}$ računa prema formuli (3.5).

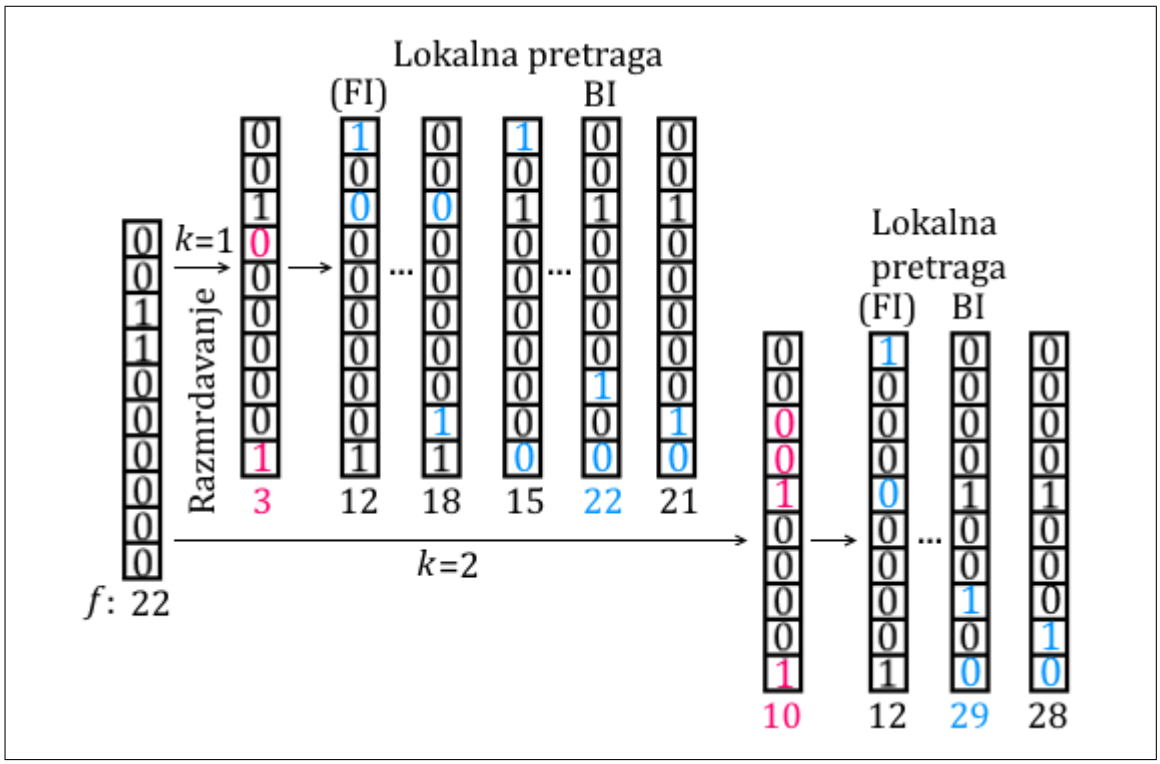
$$\begin{aligned} d = & \sum_{k \in P_j} (1 - \operatorname{sgn}(\tilde{y_k})) \frac{w_k}{1 + \operatorname{sgn}(\tilde{z_k})} \\ & + \sum_{k \in T_j} (1 - \operatorname{sgn}(\tilde{y_k}))(1 - \operatorname{sgn}(\tilde{z_k})) \frac{w_k}{2} \\ & - \sum_{k \in P_i \setminus P_j} \operatorname{sgn}(\tilde{y_k})(1 - \operatorname{sgn}(\tilde{y_k} - 1)) \frac{w_k}{1 + \operatorname{sgn}(\tilde{z_k})} \\ & - \sum_{k \in T_i \setminus T_j} \operatorname{sgn}(\tilde{y_k}) \operatorname{sgn}(\tilde{z_k})(1 - \operatorname{sgn}(\tilde{z_k} - 1)) \frac{w_k}{2} \end{aligned} \tag{3.5}$$

   Potražnje korisnika koje bi kompanija A uspostavljanjem objekta j preuzela od kompanije B treba sabrati, a potražnje korisnika koji bi uklanjem objekta i bili pridruženi kompaniji B treba oduzeti. Ova procedura je zasnovana na pretpostavci da su potencijalne lokacije tako raspoređene da su skupovi $\tilde{P}$ i $\tilde{T}$

relativno mali u odnosu na ukupan broj klijenata. Pri lokalnoj pretrazi traži se par indeksa (i, j) za koje d dostiže maksimalnu vrednost. Tek ukoliko je takav par indeksa uspešno pronađen, vrše se odgovarajuće izmene nizova x, $\tilde{y}$ i $\tilde{z}$, a na tekuću vrednost funkcije cilja se dodaje izračunata vrednost d.

8. **Kriterijum zaustavljanja**
   Kriterijum zaustavljanja predložene VNS metode je maksimalan broj iteracija bez pronalaska boljeg rešenja, što se određuje parametrom $MaxIter$. Nakon zadovoljenja kriterijuma zaustavljanja, algoitam staje sa radom i do tada najbolje pronađeno rešenje se prijavljuje korisniku.

Ilustracija izvršavanja metode promenljivih okolina na primeru 2.3 je prikazana na slici 3.2. Prvo je generisano slučajno početno rešenje $x = (0, 0, 1, 1, 0, 0, 0, 0, 0, 0)$. Odgovarajuća vrednost funkcije cilja je $f = 22$. Tokom faze razmrdavanje se bira slučajno rešenje iz okoline N_1. Kako lokalna popravka ne daje bolje rešenje čak ni korišćenjem BI pristupa, ponovo se obavlja razmrdavanje početnog rešenja u odnosu na okolinu N_2. Ukoliko se koristi BI pristup, lokalnom pretragom okoline novog slučajno izabranog rešenja se pronalazi rešenje $x' = (0, 0, 0, 0, 1, 0, 0, 1, 0, 0)$ sa vrednošću funkcije cilja $f' = 29 > 22$. Međutim, lokalna pretraga sa LI pristupom ne bi dala bolje rešenje, jer bi se pretraga zaustavila nakon pronalaska rešenja x" $= (1, 0, 0, 0, 0, 0, 0, 0, 0, 1)$. Vrednost funkcije cilja koja odgovara rešenju x" jednaka je 12, što nije bolje od 22.



Slika 3.2: Ilustracija VNS metode na primeru

3.3 Genetski algoritam

Jedna važna grupa heurističkih metoda je zasnovana na Darvinovoj [1] teoriji evolucije. Tokom 60-ih godina XX veka, nekoliko grupa naučnika inspirisanih principima Darvinove teorije se istovremeno bavilo razvojem takozvanih *evolutivnih algoritama* (engl. *Evolutionary Algorithm*, EA). U Berlinu su Rehenberg [35] i Švafel [40] razvili *evolutivnu strategiju* (nem. *Evolutionsstrategie*, ES), dok su u San Dijegu Fogel i Bremerman radili na *evolutivnom programiranju* (engl. *Evolutionary Programming*, EP) [12]. Holand je 1975. godine u knjizi [20] predložio *genetski algoritam* (engl. *Genetic Algorithm*, GA). dok je *genetsko programiranje* (engl. *Genetic Programming*, GP) Koza predložio u radu [25].

Osnovna ideja genetskih algoritama je simulacija procesa evolucije. Prema Darvinu, bolje prilagođene jedinke imaju veću šansu da prežive i putem razmnožavanja prenesu svoj genetski materijal u narednu generaciju. Na taj način, celokupna vrsta se kontinuirano usavršava. Generalno, genetski algoritam se sastoji u iterativnoj popravci inicijalne populacije rešenja koja se često nazivaju *jedinke* ili *hromozomi*. Svaka jedinka odgovara jednom rešenju u pretraživačkom prostoru. Jedinke se evaluiraju na osnovu funkcije prilagođenosti koja odgovara vrednosti funkcije cilja. U svakom koraku algoritma, jedinke se podvrgavaju genetskim operatorima. Operator selekcije obezbeđuje da verovatnoća sa kojom jedinka prelazi u narednu generaciju zavisi od njene prilagođenosti. Primena operatora ukrštanja i mutacije na selektovane jedinke simulira proces reprodukcije. Algoritam 3.7 je jedna vrlo uopštena shema genetskog algoritma. Gotovo svi koraci se mogu na veliki broj različitih načina prilagoditi konkretnoj primeni. U literaturi nalazimo brojne primene GA za rešavanje $\mathcal{NP}$-teških optimizacionih problema: [13, p. 412] i [36]. Primene GA za rešavanje raznih diskretnih lokacijskih problema se mogu naći u radovima [1], [2], [4], [27], [28], [47],[48], itd.

3.3.1 Implementacija GA metode i primena na MAXCAP

U ovoj podsekciji je opisan način primene GA na MAXCAP. Predložena implementacija ima sledeće elemente.

1. **Reprezentacija rešenja**
   Za reprezentaciju promenljivih x_j iz formulacije problema se koristi binarni vektor $x = (x_1, x_2, ..., x_{|J|})$. Predložena metoda je implementirana tako da se razmatraju samo dopustiva rešenja. Tačno p gena x_j ima vrednost 1 i označava lokacije na kojima kompanija A uspostavlja objekte. Vrednosti preostalih $|J| - p$ gena su postavljene na 0 što odgovara slobodnim lokacijama.

2. **Generisanje inicijalne populacije**
   Za generisanje jedinki prve populacije može se iskoristiti bilo koja od generalnih tehnika za populacione metaheuristike [50, pp. 193-198]. U ovom radu

[1] Čarls Darvin (engl. *Charles Darwin*, 1809–1882), engleski prirodnjak i geolog. Proglašen je za jednog od najuticajnih ljudi u istoriji. Poznata Darvinova knjiga *"Postanak vrsta pomoću prirodne selekcije ili Održavanje favorizovanih rasa u borbi za život"* [5] je imala ključnu ulogu u popularizaciji za tadašnje vreme novog načina razumevanja procesa razvoja života na Zemlji.

Algoritam 3.7: Osnovna struktura GA metode

Ulaz: *GA parametri i kriterijum zaustavljanja*;
Inicijalizacija: generisati početnu populaciju jedinki;
while *nije zadovoljen kriterijum zaustavljanja* **do**
  if *ispunjen uslov za ukrštanje* **then**
    Procesom selekcije izabrati jedinke-roditelje;
    Izabrati parametre ukrštanja;
    Izvršiti ukrštanje;
  end
  if *ispunjen uslov za mutaciju* **then**
    Izabrati gene za mutaciju;
    Primeniti mutaciju;
  end
  Izvršiti evaluaciju jedinki-potomaka na osnovu funkcije prilagođenosti;
  Izabrati sledeću generaciju jedniki;
end

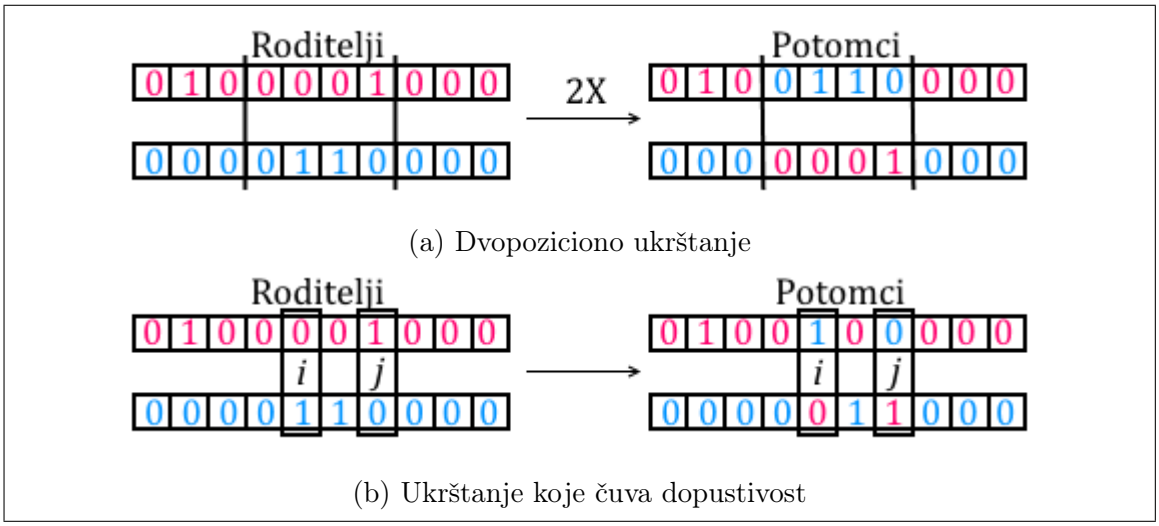
je primenjena pseudo-slučajna strategija. Svakoj jedinki odgovara dopustivo početno rešenje. Poželjno je da se različitim jedinkama dodele različita rešenja. Iz tog razloga se najpre formira binarni vektor $\tilde{x}$ dužine $|J|$. Vrednosti prvih p elemenata vektora $\tilde{x}$ se postave na 1, a ostatak na 0. Primenom funkcije programskog jezika C++ `random_shuffle` na $\tilde{x}$ se za svaku jedinku generiše slučajna permutacija vektora $\tilde{x}$.

3. **Evaluacija i selekcija**
   Prilagođenost jedinke direktno odgovara vrednosti funkcije cilja rešenja koje joj odgovara. Inicijalna vrednost funkcije cilja se računa po formuli (2.1), a ta vrednost se zatim ažurira uporedo sa primenom genetskih operatora. Prilikom selekcije je potrebno obezbediti da verovatnoća izbora određene jedinke zavisi od njene prilagođenosti. Međutim, kako ne bi došlo do preuranjene konvergencije u lokalni maksimum, slabije prilagođene jedinke se ne smeju potpuno isključiti. Proces selekcije se može realizovati putem ruletske selekcije, rangiranjem prema funkciji prilagođenosti ili na neki drugi način [50, pp. 206-208]. U ovom radu je primenjena *turnirska selekcija* (engl. *tournament selection*) koja se sastoji u sledećem. Iz tekuće populacije se na slučajan način bira unapred zadat broj jedinki. Taj broj je određen parametrom N_{tour} koji predstavlja veličinu turnira. Od izabranih N_{tour} jedinki se zatim selektuje najbolje prilagođena jedinka. Time se i slabije prilagođenim jedinkama daje određena šansa da budu selektovane za reprodukciju.

4. **Reprodukcija**
   Reprodukcija se realizuje primenom genetskih operatora *mutacije* i *ukrštanja*. Ukrštanje se implementira kao binarni (ili n-arni) operator kojim se kombinuje genetski materijal dve (ili n) izabrane jedinke. U Holandovoj osnovnoj strukturi GA [20] se ukrštanje primenjuje uvek, dok se mutacija koristi u manjoj meri. Moguće su i druge konfiguracije (uvek oba operatora, uvek tačno jedan, jedan ili dva...). Mutacija je unarni operator kojim se vrše male promene u genetskom zapisu radi bolje pretrage prostora rešenja. Zbog ograničenja (2.5),

genetski operatori bi mogli narušiti dopustivost rešenja. Jedan način na koji se to može izbeći je predložen u [47] za hab lokacijske probleme, a ovde je prilagođen za MAXCAP.

5. **Operator ukrštanja**
   Par izabranih roditelja se ukršta sa verovatnoćom koja je određena parametrom p_{cross}. Za binarne probleme se često koristi jednopoziciono (1X), dvopoziciono (2X) ili višepoziciono ukrštanje [50]. Međutim, primenom ovih tipova ukrštanja se može narušiti uslov (2.5), što je ilustrovano na slici 3.3a. Jedan pristup je da se jedinke kojima odgovaraju nedopustiva rešenja odbace tokom procesa selekcije. Recimo, odgovarajuća vrednost funkcije prilagođenosti se može postaviti na 0. Drugi pristup, koji je primenjen u ovom radu, je konstruisanje operatora ukrštanja kojim se čuva dopustivost rešenja. Osnovna ideja je preuzeta iz [47] i sastoji se u sledećem. Traže se prvi gen sa leva na desno i na kom jedan roditelj ima vrednost 0, a drugi 1, i prvi gen sa desna na levo j na kom prvi roditelj ima vrednost 1, a drugi 0. Vrednosti gena na tim pozicijama se razmenjuju. Ovaj korak se ponavlja do trenutka kada i postane veće ili jednako j. Roditelji se zatim zamenjuju svojim potomcima. Ilustracija primene operatora ukrštanja na primeru 2.3 je prikazana na slici 3.3b.

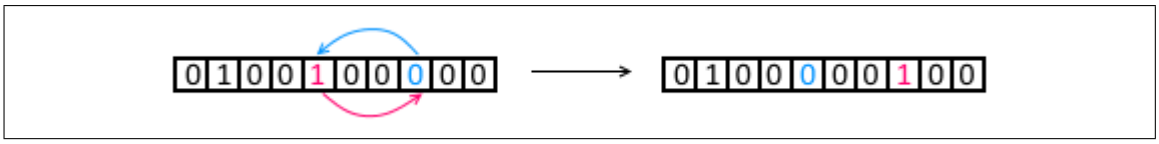


(a) Dvopoziciono ukrštanje

(b) Ukrštanje koje čuva dopustivost

Slika 3.3: Ilustracija operatora ukrštanja

6. **Operator mutacije**
   Operator ukrštanja ponekad nije dovoljan da se obezbedi neophodna raznovrsnost genetskog materijala, pogotovo pri pojavi *zaleđenih* (engl. *frozen*) gena. Kaže se da je gen zaleđen ukoliko visok procenat jedinki ima istu vrednost gena na toj poziciji. Dobro implementiran operator mutacije može značajno uticati na efikasnost i bolju pretragu prostora rešenja. Pri korišćenju binarne reprezentacije se operator mutacije uobičajeno implementira kroz invertovanje bitova [50]. U ovom radu mutacija je implementirana tako da se ne naruši uslov (2.5). Preciznije, mutacija jedinke se sastoji u slučajnom izboru dva gena sa suprotnim vrednostima koji zamenjuju pozicije (odnosno, bitovi na tim pozicijama se invertuju). Operator mutacije se primenjuje sa verovatnoćom $p_{mut} = \frac{0.4}{|J|}$ ukoliko je gen zaleđen, a sa verovatnoćom $p_{mut} = \frac{1}{|J|}$ inače. Ilustracija operatora mutacije za primer 2.3 je predstavljena na slici 3.4.



Slika 3.4: Ilustracija operatora mutacije

7. **Nova generacija**
   U radu je implementiran stacionarni GA sa elitističkom strategijom. Naime, u cilju očuvanja dobrog genetskog materijala, određen broj najbolje prilagođenih jedinki bez promene direktno prelazi u sledeću generaciju. Genetski operatori se primenjuju samo na preostale jedinke.

8. **Kriterijum zaustavljanja**
   Kriterijum zaustavljanja je određen parametrom $MaxGen$ koji predstavlja maksimalan broj GA generacija. Nakon zadovoljenja kriterijuma zaustavljanja, korisniku se prijavljuje rešenje koje odgovara najbolje prilagođenoj jedinki.

3.4 Predlog hibridizacije optimizacije kolonijom pčela i metode promenljivih okolina

Svaka od predloženih heurističkih metoda za rešavanje MAXCAP-a ima određene prednosti ali i nedostatke. U cilju dobijanja kvalitetnijih rešenja, konstruisana je hibridna metoda koja kombinuje dve prethodno opisane metaheuristike: optimizaciju kolonijom pčela sa popravkom i metodu promeljivih okolina. Predložena impelementacija BCOi metode je prilično efikasna, dok je implementacija VNS metode sporija ali pronalazi rešenja boljeg kvaliteta. U osnovi hibridne metode leži BCOi, uz određene modifikacije leta unapred. Ideja hibridizacije je da se u prvobitnu stohastičku implementaciju uvedu koncepti preuzeti iz VNS metode, ali tako da se ne izgubi mnogo na efikasnosti. Precizinije, skup veštačkih pčela je podeljen u dva skupa. Prvi podskup čini B_1 pčela koje tokom leta unapred svoje rešenje popravljaju primenom VNS metode. Počevši od trenutnog rešenja, svaka pčela iz ovog podskupa obavlja fazu razmrdavanja i fazu lokalne pretrage. Korak se ponavlja do zadovoljenja nekog kriterijuma zaustavljanja. Drugi podskup čini preostalih B_2 pčela koje istražuju prostor rešenja na stohastički način, odnosno bez faze lokalne pretrage.

Predložena hibridna metoda koristi nekoliko parametara. B_1 predstavlja broj pčela koje primenjuju VNS tokom leta unapred, a B_2 broj pčela koje primenjuju stohastičku pretragu prostora rešenja. Ukupan broj pčela koje učestvuju u realizaciji metode je predstavljen parametrom B i jednak je $B_1 + B_2$. Parametar NC označava broj koraka tokom leta unapred, a parametar k_{max} broj okolina koje se koriste prilikom VNS metode. Potrebno je definisati kriterijum zaustavljanja VNS metode u okviru leta unapred, kao i kriterijum zaustavljanja celokupne hibrdne metode. U tu svrhu se može iskoristiti bilo koji od uobičajenih kriterijuma zaustavljanja: maksimalan broj iteracija, maksimalan broj iteracija bez pronalska boljeg rešanja, utrošeno procesorsko vreme itd.

Ukoliko se vrednost parametra B_1 postavi na 0, hibridna metoda se svodi na BCO. Ukoliko se vrednost parametra B_2 postavi na 0, dobija se BCO metoda u

kojoj sve pčele uvek primenjuju VNS tokom leta unapred. Variranjem vrednosti ova dva parametra se tako može podešavati rad metode u skladu sa karakteristikama razmatranog problema i instancama koje se rešavaju, kao i vremenskim i memorijskim resursima koji stoje na raspolaganju.

Algoritam 3.8 prikazuje osnovne korake predložene hibridne metode. Osnovna razlika u odnosu na algoritam BCOi metode 3.1 je što se uvode dve varijante leta unapred.

Algoritam 3.8: Osnovna struktura BCO-VNS metode

Ulaz: *Parametri B_1, B_2, NC, k_{max} i kriterijum zaustavljanja*;
Inicijalizacija: dodeliti kompletno početno rešenje svakoj pčeli;
while *nije ispunjen kriterijum zaustavljanja* **do**
  `// let unapred`
  for $b = 1$ **to** B_1 **do**
    for $n = 1$ **to** NC **do**
      while *nije ispunjen kriterijum zaustavljanja za VNS* **do**
        Primeniti korake algoritma 3.4 na tekuće rešenje pčele b.
      end
    end
  end
  for $b = 1$ **to** B_2 **do**
    for $n = 1$ **to** NC **do**
      Proceniti korake i pravilom ruletske selekcije izabrati sledeći;
    end
  end
  `// let unazad`
  for $b = 1$ **to** B **do**
    Proceniti kvalitet rešenja pčele b;
  end
  for $b = 1$ **to** B **do**
    Odrediti da li je pčela b lojalna ili ne;
  end
  for $b = 1$ **to** B **do**
    if *pčela b je neopredeljena* **then**
      Pravilom ruletske selekcije izabrati regrutera
      i preuzeti njegovo rešenje;
    end
  end
  Proceniti rešenja svih pčela i izabrati najbolje;
end
return najbolje pronađeno rešenje;

Moguće su dalje nadogradnje predložene hibridne metode. Faza lokalne pretrage je ključni deo VNS metode, ali u zavisnosti od dimenzije problema može biti izuzetno zahtevna. Detaljna pretraga velikih okolina može trošiti značajno procesorsko vreme i resursa, naročito u slučaju većeg broja pčela koje istovremeno primenjuju VNS. Zbog toga ima smisla da vrednost parametra B_1 bude obrnuto

proporcionalna veličini okoline za lokalnu pretragu, kako bi se obezbedila veća brzina izvršavanja.

3.4.1 Implementacija BCO-VNS metode i primena na MAXCAP

Implementacija predložene hibridne metode za MAXCAP se sastoji iz sledećih elemenata. Pojedinačni segmenti su preuzeti iz implementacija BCO i VNS metoda predloženih u radu uz neophodne modifikacije.

1. **Predprocesiranje**
   Korak predprocesiranja hibridne metode je implementiran na isti način kao korak 1 iz predložene implementacije BCO metode opisane u podsekciji 3.1.1. Radi postizanja bolje efikasnosti BCO-VNS metode, osim parametara w_i, P_i i T_i za svakog korisnika $i \in I$, čuvaju se informacije o skupovima $\tilde{P}_j$ i $\tilde{T}_j$ za svaku potencijalnu lokaciju $j \in J$.

2. **Reprezentacija rešenja**
   Svakoj veštačkoj pčeli odgovara jedno kompletno rešenje. Kodiranje rešenja se vrši po notaciji opisanoj u koraku 2 predložene implementacije BCO metode u podsekciji 3.1.1. Preciznije, za predstavljanje uspostavljenih lokacija koristi se binarni vektor $x = (x_1, ..., x_{|J|})$, a za reprezentaciju pridruživanja korisnika objektima kompanije A koriste se binarni vektori $y = (y_1, ..., y_{|I|})$ i $z = (z_1, ..., z_{|I|})$.

3. **Generisanje početnih rešenja**
   Bez obzira da li pčela pripada skupu B_1 ili B_2, početna rešenja se generišu na isti način. Kako bi se obezbedila inicijalna rešenja boljeg kvaliteta, primenjuje se ista procedura kao u okviru predložene BCO metode, a koja je opisana u koraku 3 podsekcije 3.1.1.

4. **Izračunavanje funkcije cilja**
   Vrednost funkcije cilja se ažurira nakon svake izmene vrednosti elemenata vektora x na način koji je opisan u koraku 4 iz predložene implementacije BCO metode u podsekciji 3.1.1.

5. **Let unapred**
   Ovo je najbitiniji korak za postizanje boljeg kvaliteta rešenja i jedini je modifikovan u odnosu na predloženu implementaciju BCO metode. Pčele iz skupa B_2 obavlja stohastički let unapred opisan u odgovarajućem koraku 5 u podsekciji 3.1.1. Stohastički pristup preuzet iz BCO metode obezbeđuje potpuno različit tretman istog rešenja od strane različitih pčela. Pčele iz skupa B_1 primenjuju modifikaciju leta unapred, a koja se sastoji u primeni VNS metode na trenutno rešenje koje odgovara datoj pčeli. Sistematična varijanta leta unapred se sastoji iz faze razmrdavanja i faze lokalne pretrage, koje su opisane u narednim koracima. Obe varijante leta unapred su implementirane tako da se očuva dopustivost rešenja, odnosno ispunjenost uslova (2.4).

6. **Definicija okolina**
   Definicija okolina je opisana u koraku 4 iz podsekcije 3.2.1 o predloženoj imple-

mentaciji VNS metode. Za fazu razmrdavanja koristi se niz od $N_{k_{max}}$ okolina, dok se za fazu lokalne pretrage koristi okolina N_1.

7. **Faza lokalne pretrage**
   Lokalna pretraga je implementirana u okviru funkcije čija je osnovna struktura prikazana na algoritmu 3.5. Tokom faze lokalne pretrage se primenjuje pristup najboljeg poboljšanja. Prilikom pretrage, traže se lokacije $i \in J_P$ i $j \in J_U$ takve da se uspostavljanjem objekta na lokaciji i i uklanjanjem objekta na lokaciji j postiže najveće poboljšanje vrednosti funkcije cilja.

8. **Faza razmrdavanja**
   Osnovna struktura implementacije faze razmrdavanja u okolini N_k prikazana na algoritmu 3.6. U odnosu na pristup iskorišćen u okviru predložene VNS metode, funkcija je modifikovana tako da se koriste uvedeni pomoćni skupovi. Umesto funkcije `random_shuffle`, za izbor lokacije objekata koje treba uspostaviti ili ukloniti iz skupova J_P, odnosno J_U koristi se generator slučajnih brojeva sa uniformnom raspodelom.

9. **Let unazad**
   Implementacija leta unazad je uniformna za sve pčele $b \in B$, tako da je i ovaj korak preuzet iz implementacije BCO metode. Let unazad se sastoji iz evaluacije rešenje, određivanja lojalnosti i izbora regrutera. Detaljniji opis je naveden u okviru koraka 6 u podsekciji 3.1.1.

10. **Kriterijum zaustavljanja**
    Kriterijum zaustavljanja za VNS metodu tokom modifikovanog leta unapred je određen maksimalnim brojem koraka bez pronalaska boljeg rešenja. Ovaj broj je određen parametrom $MaxIter_{VNS}$. Nakon zadovoljenja kriterijuma zaustavljanja, pčela završava let unapred i prelazi na sledeći korak, odnosno let unazad. Slično, kriterijum zaustavljanja hibridne metode je maksimalan broj iteracija bez izmene najboljeg globalnog rešenja x_{best}. Rešenje x_{best} i odgovarajuća vrednost funkcije cilja f_{best} se proveravaju i, ako je potrebno, ažuriraju nakon svake BCO iteracije. Po zadovoljenju kriterijuma zaustavljanja, najbolje globalno pronađeno rešenje se prijavljuje korisniku.

4
Eksperimentalni rezultati

Sva testiranja su izvršena na Intel(R) Core i5 procesoru pod Windows 8.1 operativnim sistemom. Rešenja dobijena predloženim heurističkim metodama na razmatranim test instancama su upoređena međusobno, kao i sa optimalnim rešenjima dobijenim egzaktnim rešavačem CPLEX, verzija 12.1. Vremensko ograničenje izvršavanja rešavača CPLEX postavljeno je na 4 sata. Sve heurističke metode predložene u radu su implementirane u okviru konzolne aplikacije u programskom jeziku C++.

4.1 Test instance

Za generisanje adekvatnih test instanci razmatranog problema, iskorišćene su postojeće test instance iz ORLIB biblioteke [3] za *lokacijske probleme sa kapacitetom* (engl. *Capacitated Facility Location Problems*, CFLPs), *lokacijske probleme bez kapaciteta* (engl. *Uncapacitated Facilty Location Problems*, UFLPs) i različite varijante *hab lokacijskih problema* (engl. *Hub Location Problems*, HLPs), [3]. Modifikacijom postojećih instanci dobijene su test instance za MAXCAP.

Instance za CFLP i UFLP preuzete iz ORLIB biblioteke sadrže sledeće podatke: broj potencijalnih lokacija i broj klijenata; kapacitet (zanemaruje se u slučaju UFLP) i cenu uspostavljanja za svaku potencijalnu lokaciju; potražnju i cene alokacije svakoj od potencijalnih lokacija za svakog korisnika. Instanca se prilagođava za MAXCAP tako što se kapaciteti i fiksirani troškovi uspostavljanja potencijalnih lokacija zanemaruju, a za svakog korisnika i se generišu skupovi P_i i T_i. Neka su sa c^i_{min} i c^i_{max} i c^i_B redom označene minimalna i maksimalna cena alokacije korisnika i. Uz pretpostavku da su objekti konkurentske kompanije dobro locirani, može se smatrati se da će vrednost c^i_B koja predstavlja cenu alokacije korisnika i kompaniji B biti bliska minimalnoj. Za svakog korisnika i, vrednost c^i_B se bira na slučajan način iz segmenta $[c^i_{min}, c^i_{min} + 0.3 \cdot (c^i_{max} - c^i_{min})]$. Zatim, indeksi lokacija za koje je cena alokacije manja od c^i_B se smeštaju u skup P_i, a indeksi lokacija za koje je cena alokacije jednaka c^i_B u skup T_i.

Instance koje se u literaturi koriste za razne varijante HLPs sadrže: broj habova koje treba uspostaviti, koordinate čvorova, matricu protoka, cene prikupljanja, transfera i distribucije robe preko habova. Iz skupa potencijalnih lokacija za habove se na slučajan način bira $|J|$ čvorova koji predstavljaju potencijalne lokacije za izgradnju objekata snabdevača u problemu MAXCAP. Ostali čvorovi predstavljaju lokacije korisnika za koje je potrebno generisati skupove P_i i T_i. Na osnovu matrice protoka polazne hab instance, računa se količina protoka (robe) koja stiže u svaki čvor. Čvorovi koji će odgovarati potencijalnim lokacijama se sortiraju u nerastući poredak

na osnovu izračunatih vrednosti. Iz skupa prvih $0.3 \cdot |J|$ tako uređenih čvorova, na slučajan način se bira q čvorova koji predstavljaju objekte kompanije B. Za svakog korisnika i se, na osnovu njegovih koordinata, određuje najbliži objekat b_i kompanije B. U skup P_i se smeštaju indeksi čvorova - potencijalnih lokacija čije je Euklidsko rastojanje od korisnika i manje u odnosu na rastojanje korisnika i do lokacije b_i. U skup T_i se smeštaju indeksi čvorova koji su na istom Euklidskom rastojanju od korisnika i kao čvor b_i. Poređenje se vrši do na tačnost koja je određena parametrom ϵ, a čija je vrednost utvrđena eksperimentalnim putem za svaku konkretnu polaznu hub test instancu.

Instance generisane na prethodna dva načina su relativno malih dimenzija, tako da ih je uglavnom moguće rešiti i egzaktnim metodama. Kako bi se ispitalo ponašanje predloženih implementacija heurističkih metoda na instancama problema koje su nedostižne za egzaktne metode, generisano je nekoliko instanci za MAXCAP znatno većih dimenzija. Vrednosti parametara $|I|$ i $|J|$ su postavljene na značajno veće vrednosti u odnosu na dimenzije instanci iz prethodna dva generisana skupa. Skupovi P_i i T_i su formirani slučajnim izborom indeksa. Odnos broja elemenata skupova P_i i T_i i ukupnog broja potencijalnih lokacija $|J|$ je određen posmatranjem odgovarajućih prosečnih vrednosti na instancama manjih dimenzija. Za svaku MAXCAP instancu dobijenu nekim od opisana tri načina, varirane su vrednosti parametra p.

Radi lakšeg prikaza rezultata, instance korišćene u radu su podeljene u četiri grupe na osnovu vrednosti parametara $|I|$ i $|J|$: instance malih dimenzija (M), instance srednjih dimenzija (S), instance velikih dimenzija (V) i izuzetno velikih dimenzija (Rnd). Prve tri grupe sadrže po 13 instanci dobijenih transformacijom postojećih instanci iz literature. Četvrta grupa je sačinjena od 10 slučajno generisanih instanci.

Tekstualni fajlovi sa ulaznim podacima formatirani su sledeći način. U prvom redu su navedeni parametri $|I|, |J|, p$ i q. Naredni red sadrži vrednosti potražnji korisnika w_i. Zatim slede broj elemenata i sami elementi skupa P_i, odnosno T_i u posebnom redu za svako i. Primer problema maksimalnog zauzimanja naveden u 2.3 predstavljen u opisanom formatu je sledećeg oblika:

```
10 10 2 4
5 12 7 10 3 8 2 13 20 14

0
0

2 0 1
0

0
1 3

1 5
1 3
```

```
3 2 5 6
1 3

1 8
1 5

1 7
0

1 4
0

0
2 7 8

0
1 7
```

4.2 Način prezentovanja eksperimentalnih rezultata

Svaka od predloženih heurističkih metoda je testirana po 10 puta na istom skupu instanci. Rezultati testiranja su navedeni u tabelama 4.1–4.8 na sledeći način. Dimenzije instance $|I|$ i $|J|$ su navedene u prvoj koloni tabele. Druga kolona tabele sadrži vrednost parametra p.

Naredne dve kolone odnose se na rezultate dobijene rešavačem CPLEX. Navedeno je optimalno rešenje (*opt*) i vreme izvršavanja (t) u sekundama. U kolonama *opt* i t je simbolom '-' označeno da za datu instancu CPLEX rešavač nije pronašao optimalno rešenje usled memorijskog ili vremenskog ograničenja.

Preostale kolone se odnose na heurističku metodu i sadrže najbolje rešenje dobijeno heuristikom (*sol*), prosečno vreme za koje heuristika prvi put dostiže najbolje rešenje (t_{best}) u sekundama, prosečno ukupno vreme izvršavanja heurističke metode (t_{tot}) u sekundama, prosečno odstupanje od najboljeg poznatog rešenja (*agap*) u procentima i standardnu devijaciju (σ) u procentima. Vrednosti *agap* i *sigma* se računaju po sledećim formulama:

$$agap = \frac{\sum_{i=1}^{10} gap_i}{10}, \tag{4.1}$$

$$\sigma = \sqrt{\frac{1}{10}\sum_{i=1}^{10}(gap_i - agap)^2}. \tag{4.2}$$

Neka je za datu instancu sa sol_{best} označeno optimalno ili najbolje poznato rešenje, a sa sol_i najbolje rešenje dobijeno heuristikom i-tom izvršavanju. Za svako $i = 1, 2, ..., 10$, vrednosti gap_i iz (4.1) se računaju po formuli (4.3) i predstavljaju

procentualno odstupanje sol_i od sol_{best}:

$$gap_i = 100 \cdot \frac{|sol_i - sol_{best}|}{|sol_{best}|}. \tag{4.3}$$

4.3 Eksperimentalni rezultati dobijeni BCO metodom

Parametri implementirane BCO metode su određeni eksprimentalnim putem. Broj veštačkih pčela B je postavljen na 10, a broj koraka tokom leta unapred NC na 1. Dopušteno je maksimalno $MaxIter = 500$ iteracije metode bez popravke tekućeg najboljeg rešenja BCO metode.

U tabeli 4.1 se mogu videti rezultati koje je BCO metoda postigla na instancama malih i srednjih dimenzija. BCO metodom je optimalno rešenje pronađeno za manje vrednosti parametra p, dok su u ostalim slučajevima dobijena rešenja nešto slabijeg kvaliteta u odnosu na rešenja koja je dao CPLEX rešavač.

Tabela 4.1: Rezultati BCO metode na instancama malih i srednjih dimenzija

ULAZ		CPLEX		BCO				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	5823.5	0.06	5823.5	3.40	16.65	0.000	0.000
	3	8193.5	0.07	8193.5	11.11	23.95	1.236	1.113
	4	10324	0.06	10272	13.66	26.35	1.635	1.350
	5	12402.5	0.10	12259.5	8.22	20.46	3.540	1.564
	6	14363.5	0.13	14160	11.72	23.64	3.618	1.025
$\|I\| = 1000$	7	16320	0.09	15804	9.27	20.77	5.345	1.593
	8	18108.5	0.10	17304	8.50	19.56	5.839	0.956
$\|J\| = 100$	9	19884.5	0.07	18970	8.30	19.06	6.769	1.147
	10	21554.5	0.06	20045	5.83	16.37	8.755	1.053
	20	33967	0.05	31666	7.32	15.50	8.544	1.154
	30	41260.5	0.04	38488.5	4.67	10.99	7.411	0.504
	40	45315.5	0.03	43394.5	3.94	8.93	4.891	0.434
	50	47673	0.02	46416.5	2.23	6.32	3.143	0.258
	2	13519.5	2.63	13519.5	13.00	63.74	0.000	0.000
	3	19318.5	3.02	19318.5	21.03	67.30	1.953	1.467
	4	24532.5	3.66	24412.5	21.96	64.16	3.164	1.259
	5	29698.5	4.26	28986.5	33.23	71.82	3.639	0.876
	6	34680.5	4.60	33582	31.59	68.60	4.869	0.890
$\|I\| = 3000$	7	39542	4.85	37810.5	24.77	62.51	5.975	0.910
	8	44186	6.54	42288.5	33.25	64.04	6.246	0.934
$\|J\| = 100$	9	48697.5	8.33	46478.5	18.60	47.78	6.459	1.040
	10	52983.5	9.12	50761.5	30.04	58.28	7.123	1.091
	20	86302.5	38.16	80243.5	25.00	45.23	8.346	0.701
	30	105817.5	159.32	99734.5	11.69	27.81	6.754	0.621
	40	117687	9.79	113634	13.18	25.86	4.379	0.478
	50	125857	1.97	122241	9.64	19.71	3.229	0.263

Rezultati BCO metode na grupi instanci većih dimenzija su prikazani u tabeli 4.2. Iako su sve test instance velikih dimenzija uspešno rešene predloženom implementacijom BCO metode, dobijeno rešenje se poklapa sa optimalnim samo za manje vrednosti parametra p. Sa druge strane, vreme izvršavanja BCO metode je dosta kraće od vremena izvršavanja CPLEX rešavača.

Tabela 4.2: Rezultati BCO metode na instancama velikih dimenzija

ULAZ		CPLEX		BCO				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	27607	3.45	27607	46.47	100.44	1.023	0.990
	3	39184	4.96	38894.5	46.37	95.64	1.953	1.467
	4	48064	20.88	47049.5	29.74	73.75	3.759	0.946
	5	–	–	55071.5	53.99	96.19	4.847	1.134
	6	–	–	62262.5	31.34	72.40	6.245	0.763
$\|I\| = 3000$	7	–	–	68315.5	40.85	75.23	5.808	0.623
	8	–	–	73829.5	17.37	50.18	6.220	0.680
$\|J\| = 300$	9	–	–	77817	23.58	59.53	7.727	0.440
	10	88220	1937.94	83444.5	25.26	58.57	7.038	0.935
	20	115516	6344.87	108907	13.43	30.79	6.945	0.747
	30	129145	3838.74	121903	7.25	18.58	6.444	0.361
	40	135887	1541.02	129195	7.37	15.77	5.341	0.351
	50	140700	428.19	134571	5.07	12.00	4.900	0.279
	10	–	–	927119	235.88	473.55	0.126	0.006
	20	–	–	1710000	226.16	449.84	0.144	0.144
	30	–	–	2420000	132.98	335.57	0.147	0.005
	40	–	–	3070000	117.28	305.21	0.144	0.004
$\|I\| = 10000$	50	–	–	3660000	138.48	312.57	0.141	0.141
	60	–	–	4230000	136.89	303.19	0.137	0.137
$\|J\| = 1000$	70	–	–	4720000	105.16	264.53	0.134	0.002
	80	–	–	5180000	142.13	291.70	0.128	0.128
	90	–	–	5625460	90.99	219.68	0.127	0.002
	100	–	–	6033460	90.78	214.51	0.123	0.001

4.4 Eksperimentalni rezultati dobijeni VNS metodom

U fazi razmrdavanja se koristi tri k_{max} okolina čija je vrednost određena eksperimentalnim putem. Zbog definicije samih okolina, vrednost parametra k_{max} zavisi od parametra p i postavljena je na 2 kada kada je $p < 3$, a inače na 3. Kriterijum zaustavljanja VNS metode je maksimalan broj iteracija bez pronalaska boljeg rešenja koji je određen parametrom $MaxIter$. Vrednost parametra $MaxIter$ je postavljena na 500.

U tabeli 4.3 su navedeni rezultati VNS metode na instancama malih i srednjih dimenzija. Najbolje rešenje pronađeno VNS metodom se poklapa sa rešenjem koje je dobijeno rešavačem CPLEX na 23 od ukupno 24 instance iz ove dve grupe.

Tabela 4.3: Rezultati VNS metode na instancama malih i srednjih dimenzija

ULAZ		CPLEX		VNS				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	5823.5	0.06	5823.5	0.12	8.08	0.000	0.000
	3	8193.5	0.07	8193.5	0.72	12.09	0.000	0.000
	4	10324	0.06	10324	1.10	15.62	0.000	0.000
	5	12402.5	0.10	12402.5	0.56	18.24	0.000	0.000
	6	14363.5	0.13	14363.5	4.30	24.06	0.003	0.009
$\|I\| = 1000$	7	16320	0.09	16320	1.90	23.69	0.000	0.000
	8	18108.5	0.10	18108.5	2.46	29.76	0.007	0.020
$\|J\| = 100$	9	19884.5	0.07	19884.5	3.98	33.76	0.000	0.000
	10	21554.5	0.06	21554.5	10.28	42.96	0.000	0.000
	20	33967	0.05	33967	14.32	67.56	0.205	0.205
	30	41260.5	0.04	41260.5	22.14	83.78	0.084	0.104
	40	45315.5	0.03	45315.5	26.46	96.70	0.014	0.025
	50	47673	0.02	47673	15.39	77.55	0.000	0.000
	2	13519.5	2.63	13519.5	0.58	36.80	0.000	0.000
	3	19318.5	3.02	19318.5	1.19	52.50	0.000	0.000
	4	24532.5	3.66	24532.5	2.11	69.50	0.000	0.000
	5	29698.5	4.26	29698.5	4.32	87.70	0.000	0.000
	6	34680.5	4.60	34680.5	9.41	103.73	0.056	0.036
$\|I\| = 3000$	7	39542	4.85	39542	21.85	127.39	0.108	0.325
	8	44186	6.54	44186	17.64	127.26	0.109	0.166
$\|J\| = 100$	9	48697.5	8.33	48697.5	33.60	163.83	0.068	0.136
	10	52983.5	9.12	52983.5	33.29	183.26	0.104	0.228
	20	86302.5	38.16	86302.5	91.45	336.84	0.513	0.377
	30	105817.5	159.32	105745	197.43	485.70	0.266	0.196
	40	117687	9.79	117687	221.25	541.82	0.110	0.093
	50	125857	1.97	125857	187.93	522.27	0.066	0.045

Rezultati VNS metode na instancama velikih dimenzija si prikazani u tabeli 4.4. VNS metodom su uspešno rešene sve test instance ove grupe. Rešenje koje je dobijeno VNS metodom je istog ili malo lošijeg kvaliteta u odnosu na najbolje poznato rešenje za datu instancu.

Tabela 4.4: Rezultati VNS metode na instancama velikih dimenzija

ULAZ		CPLEX		VNS				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	27607	3.45	27607	1.70	107.69	0.000	0.000
	3	39184	4.96	39184	3.17	164.59	0.000	0.000
	4	48064	20.88	48064	3.02	211.55	0.000	0.000
	5	–	–	56676.5	103.76	378.02	0.000	0.000
	6	–	–	65120.5	76.72	381.01	0.000	0.000
$\|I\| = 3000$	7	–	–	71364.5	60.77	400.37	0.098	0.114
	8	–	–	77551	207.86	818.01	0.169	0.107
$\|J\| = 300$	9	–	–	83610	173.21	870.08	0.323	0.325
	10	88220	1937.94	88220	556.75	1140.64	0.194	0.203
	20	115516	6344.87	115281	800.97	2012.65	0.363	0.119
	30	129145	3838.74	129103	584.01	1705.56	0.221	0.181
	40	135887	1541.02	135803	1092.46	2495.58	0.261	0.139
	50	140700	428.19	140540	1408.15	3041.46	0.192	0.065
	10	–	–	1060000	735.32	1562.02	0.008	0.003
	20	–	–	1980000	1901.61	3607.08	0.001	0.002
	30	–	–	2810000	2924.45	5356.57	0.001	0.001
	40	–	–	3560000	1594.36	4706.62	0.000	0.000
$\|I\| = 10000$	50	–	–	4244620	10755.80	15373.20	0.000	0.000
	60	–	–	4868410	10802.80	16876.30	0.000	0.000
$\|J\| = 1000$	70	–	–	5440000	23978.30	29014.60	0.000	0.000
	80	–	–	5945500	13384.50	19854.10	0.000	0.000
	90	–	–	6410920	11799.20	15937.40	0.012	0.003
	100	–	–	6860000	13246.00	20092.10	0.000	0.000

4.5 Eksperimentalni rezultati dobijeni GA metodom

Parametri predložene implementacije genetskog algoritma su određeni eksperimentalnim putem. Broj jedinki u populaciji je 150. U svakoj iteraciji metode 50 elitnih jedinki direktno prelazi u narednu generaciju. U procesu izbora jedinki za ukrštanje, primenjuje se turnirska selekcija sa veličinom turnira $N_{tour} = 5$. Na selektovani par jedinki-roditelja, primenjuje operator ukrštanja sa verovatnoćom $p_{cross} = 0.85$. Geni jedinki-potomaka se mutiraju sa verovatnoćom p_{mut}. Vrednost parametra p_{mut} je jednaka $\frac{1}{|J|}$ na normalnim genima, a $\frac{0.4}{|J|}$ na zaleđenim genima. Smatra se da je gen zaleđen ukoliko više od 95% populacije ima istu vrednost na tom genu. Maksimalan broj generacija je postavljen na 40.

Rezultati GA metode na instancama malih i srednjih dimenzija su predstavljeni u tabeli 4.5, a rezultati GA metode na instancama velikih dimenzija u tabeli 4.6.

Tabela 4.5: Rezultati GA metode na instancama malih i srednjih dimenzija

ULAZ		CPLEX		GA				
	p	*opt*	t	*sol*	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	5823.5	0.06	5823.5	27.52	118.48	5.69	3.40
	3	8193.5	0.07	7851	70.29	174.52	4.16	2.67
	4	10324	0.06	9716	170.67	198.26	8.90	4.61
	5	12402.5	0.10	11012.5	31.24	120.44	5.87	3.05
	6	14363.5	0.13	12172.5	106.88	124.15	4.02	2.13
$\|I\| = 1000$	7	16320	0.09	13654.5	16.31	94.03	5.97	3.41
	8	18108.5	0.10	14977	32.73	81.21	2.20	1.44
$\|J\| = 100$	9	19884.5	0.07	16315.5	47.69	80.54	2.61	2.48
	10	21554.5	0.06	17696.5	15.85	78.71	5.26	3.00
	20	33967	0.05	26965.5	103.13	129.05	1.30	0.78
	30	41260.5	0.04	35058	51.55	129.65	3.72	2.08
	40	45315.5	0.03	38998.5	107.00	134.89	0.35	0.33
	50	47673	0.02	43959	66.59	110.75	2.04	1.24
	2	13519.5	2.63	13017	84.39	205.87	1.54	1.41
	3	19318.5	3.02	18340.5	43.05	136.08	3.35	1.89
	4	24532.5	3.66	22703	179.94	257.06	2.09	1.93
	5	29698.5	4.26	27440	47.30	118.05	2.96	2.23
	6	34680.5	4.60	31579.5	125.22	210.91	2.52	2.01
$\|I\| = 1000$	7	39542	4.85	35552	23.73	148.45	2.20	1.19
	8	44186	6.54	40958.5	23.20	136.21	4.18	3.19
$\|J\| = 300$	9	48697.5	8.33	42285	54.25	127.16	1.30	1.08
	10	52983.5	9.12	46709.5	8.49	133.44	1.90	1.04
	20	86302.5	38.16	76208.5	24.03	81.68	3.10	1.73
	30	105817.5	159.32	93675.5	31.82	79.87	1.79	0.98
	40	117686.5	9.79	107384	14.97	75.72	1.59	0.82
	50	125856.5	1.97	117514	16.04	78.29	1.22	0.85

Tabela 4.6: Rezultati GA metode na instancama velikih dimenzija

ULAZ		CPLEX		GA				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	27607	3.45	26791	87.12	333.56	1.47	1.10
	3	39184	4.96	37397	138.16	386.52	2.33	1.55
	4	48064	20.88	45084.5	298.44	349.49	1.04	0.88
	5	–	–	52587.5	44.97	270.35	0.85	0.85
	6	–	–	59676	107.52	284.24	1.94	1.12
$\|I\| = 3000$	7	–	–	66356	243.69	401.37	2.10	1.07
	8	–	–	71547	172.47	360.41	2.54	1.28
$\|J\| = 300$	9	–	–	73861.5	138.18	334.38	0.91	1.04
	10	88220	1937.94	79526	143.49	382.99	1.66	1.43
	20	115516	6344.87	105404	166.30	279.44	1.75	1.06
	30	129145	3838.74	117982	51.41	269.03	1.04	0.90
	40	135887	1541.02	123818	51.25	270.84	0.36	0.21
	50	140700	428.19	129553	54.97	292.36	0.59	0.44
	10	–	–	860756	652.05	982.59	0.66	0.47
	20	–	–	1636790	582.60	869.81	1.02	0.77
	30	–	–	2320000	309.66	916.28	0.82	0.64
	40	–	–	2970000	536.70	846.91	0.49	0.35
$\|I\| = 10000$	50	–	–	3540000	558.12	819.42	0.49	0.35
	60	–	–	4113240	283.00	741.77	0.32	0.32
$\|J\| = 1000$	70	–	–	4607360	529.40	794.74	0.82	0.58
	80	–	–	5077250	258.26	786.80	0.50	0.48
	90	–	–	5536770	569.96	842.52	0.50	0.46
	100	–	–	5898970	567.80	854.16	1.29	1.22

4.6 Eksperimentalni rezultati dobijeni BCO-VNS metodom

Vrednosti parametara predložene hibridne metode su određene eksperimentalnim putem. Broj pčela koje u toku leta unapred primenjuju VNS je postavljen na 2, a broj pčela koje primenjuju stohastički let unapred na 8. Za sve pčele, broj koraka tokom leta unapred je postavljen na 1. Modifikovani let unapred obuhvata maksimalno 2 iteracije VNS metode bez pronalaska boljeg rešenja. U okviru faze razmrdavanja se koristi $k_{max} = 2$ okolina za razmrdavanje. Broj iteracija metode bez popravke tekućeg najboljeg rešenja je ograničen na 50.

U tabeli 4.7 su prikazani rezultati hibridne BCO-VNS metode na test instancama malih i srednjih dimenzija. Rešenje pronađeno hibridnom metodom se na svim instancama prve grupe i na 11 od ukupno 13 instanci druge grupe poklapa sa optimalnim rešenjem koje je dobijeno CPLEX rešavačem.

Tabela 4.7: Rezultati hibridne BCO-VNS metode na instancama malih i srednjih dimenzija

ULAZ		CPLEX		BCO-VNS				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	5823.5	0.06	5823.5	0.26	5.90	0.000	0.000
	3	8193.5	0.07	8193.5	0.92	7.91	0.000	0.000
	4	10324	0.06	10324	0.58	8.86	0.000	0.000
	5	12402.5	0.10	12402.5	0.69	9.95	0.000	0.000
	6	14363.5	0.13	14363.5	2.50	12.99	0.000	0.000
$\|I\| = 1000$	7	16320	0.09	16320	1.34	12.89	0.000	0.000
	8	18108.5	0.10	18108.5	2.31	14.80	0.000	0.000
$\|J\| = 100$	9	19884.5	0.07	19884.5	3.07	16.65	0.000	0.000
	10	21554.5	0.06	21554.5	4.03	18.91	0.000	0.000
	20	33967	0.05	33967	17.59	40.18	0.001	0.002
	30	41260.5	0.04	41260.5	17.80	41.75	0.001	0.001
	40	45315.5	0.03	45315.5	21.75	49.42	0.000	0.000
	50	47673	0.02	47673	16.83	45.53	0.000	0.000
	2	13519.5	2.63	13519.5	0.88	20.02	0.000	0.000
	3	19318.5	3.02	19318.5	1.30	25.13	0.000	0.000
	4	24532.5	3.66	24532.5	1.94	29.28	0.000	0.000
	5	29698.5	4.26	29698.5	2.82	35.42	0.000	0.000
	6	34680.5	4.60	34680.5	5.71	45.55	0.000	0.000
$\|I\| = 1000$	7	39542	4.85	39542	9.14	52.65	0.000	0.000
	8	44186	6.54	44186	8.78	49.10	0.000	0.000
$\|J\| = 300$	9	48697.5	8.33	48697.5	20.99	65.57	0.000	0.000
	10	52983.5	9.12	52983.5	24.04	77.26	0.000	0.001
	20	86302.5	38.16	86183.5	61.06	145.46	0.004	0.003
	30	105817.5	159.32	105745	99.02	206.33	0.002	0.002
	40	117687	9.79	117687	118.36	239.06	0.001	0.001
	50	125857	1.97	125857	117.20	231.01	0.001	0.001

U tabeli 4.8 su predstavljeni rezultati hibridne metode na instancama velikih dimenzija. Predloženom hibridizacijom su uspešno rešeni svi test primeri. Dobijeno rešenje se na samo četiri instance na poklapa sa rešenjem koje je dao rešavač CPLEX. Na instancama koje CPLEX nije rešio, najbolje rešenje je dobijeno upravo hibridnom BCO-VNS metodom.

Tabela 4.8: Rezultati hibridne BCO-VNS metode na instancama velikih dimenzija

ULAZ		CPLEX		BCO-VNS				
	p	opt	t	sol	t_{best}(s)	$t_{tot}(s)$	agap(%)	σ(%)
	2	27607	3.45	27607	1.95	46.94	0.000	0.000
	3	39184	4.96	39184	3.16	67.23	0.000	0.000
	4	48064	20.88	48064	7.20	89.16	0.000	0.000
	5	–	–	56676.5	50.09	147.25	0.000	0.000
	6	–	–	65120.5	19.91	132.69	0.000	0.000
$\|I\| = 3000$	7	–	–	71361.5	81.56	206.29	0.000	0.000
	8	–	–	77551	108.22	248.24	0.001	0.002
$\|J\| = 300$	9	–	–	83610	91.14	240.95	0.001	0.001
	10	88220	1937.94	88220	72.75	239.93	0.002	0.002
	20	115516	6344.87	115406	257.38	547.03	0.002	0.001
	30	129145	3838.74	129116	663.19	1067.30	0.002	0.001
	40	135887	1541.02	135878	717.04	1215.89	0.001	0.001
	50	140700	428.19	140651	1090.76	1651.86	0.001	0.001
	10	–	–	1060000	58.22	282.77	0.000	0.000
	20	–	–	1980000	79.49	452.16	0.000	0.000
	30	–	–	2810000	388.96	1183.99	0.000	0.000
	40	–	–	3560000	1358.04	2579.34	0.000	0.000
$\|I\| = 10000$	50	–	–	4244620	2479.22	3676.66	0.001	0.001
	60	–	–	4870000	2960.40	4350.89	0.002	0.001
$\|J\| = 1000$	70	–	–	5440000	5718.17	7471.51	0.002	0.001
	80	–	–	5930000	7477.06	9493.60	0.003	0.002
	90	–	–	6430000	9521.39	12122.60	0.000	0.000
	100	–	–	6860000	13360.74	15868.75	0.000	0.000

5
Analiza eksperimentalnih rezultata

Test instance su generisane tako da se može posmatrati uticaj različitih parametara na performanse metoda i izvršiti njihovo poređenje prema više kriterijuma. Kao što je i očekivano, sa povećavanjem razlike između p i $|J|$, rastu vreme i broj iteracija potrebnih za izvršavanje egzaktne metode. Na instancama manjih dimenzija, vreme izvršavanja CPLEX rešavača je uglavnom kraće od heurističkih metoda. Sa druge strane, implementiranim metaheuristikama su uspešno rešene sve test instance velikih dimenzija na koje se egzaktne metode ne mogu primeniti pod datim vremenskim i memorijskim ograničenjima.

Implementacija BCO metode sadrži dosta stohastičkih elemenata. Zbog toga je vreme izvršavanja relativno malo čak i na instancama većih dimenzija. Ipak, BCO metodom su pronađena rešenja koja su uglavnom nešto slabijeg kvaliteta u odnosu na najbolje poznato rešenje date instance.

Glavni korak VNS metode koji utiče na kvalitet pronađenog rešenja je faza lokalne pretrage. Ova faza se može izvršavati dugo, ukoliko je okolina koja se pretražuje velikih dimenzija. Definicija okoline koja je implementirana u radu podrazumeva da je veličina okoline za lokalnu pretragu reda $\mathcal{O}(|J|)$. Analizom dobijenih rezultata se može potvrditi da ukupno vreme izvršavanja VNS metode brzo raste sa povećanjem dimenzije test instance. Eksperimentalnim putem je utvrđeno da primena pristupa prvog poboljšanja u okviru lokalne pretrage ne donosi zadovoljavajuće poboljšanje u vremenu izvršavanja, a pritom smanjuje kvalitet dobijenih rešenja.

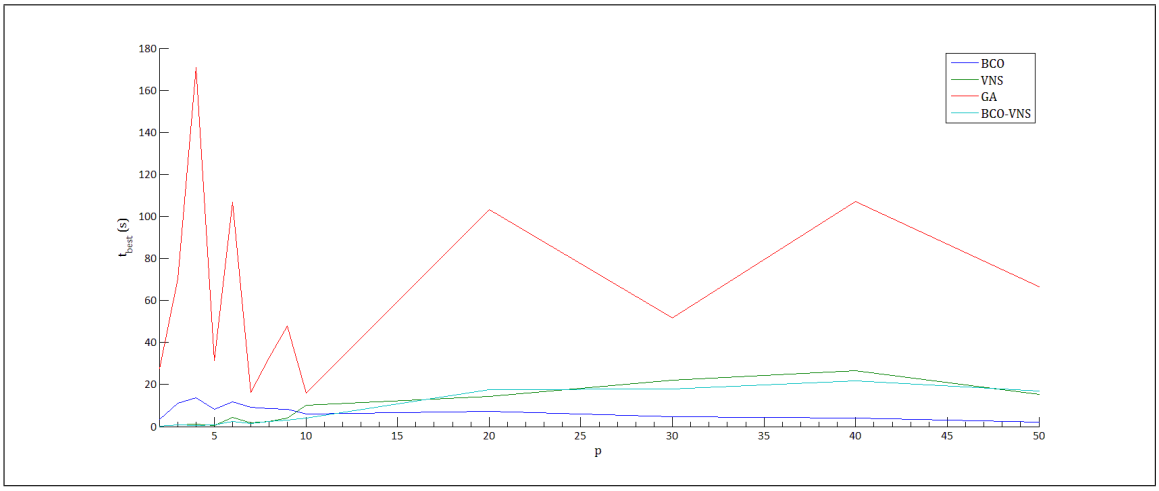
Od četiri predložene metode, rezultati najslabijeg kvaliteta su postignuti genetskim algoritmom. Međutim, moguće je da bi se povoljnijim izborom parametara metode mogli postići bolji rezultati, što je neophodno utvrditi dodatnim eksperimentima. Karakteristično je da vreme izvršavanja GA metode manje zavisi od dimenzija test instance u odnosu na ostale predložene metode.

Glavna ideja predložene hibridizacije VNS i BCO metoda jeste da se iskombinuju efikasnost optimizacije kolonijom pčela i kvalitet pretrage metode promenljivih okolina. Hibridna BCO-VNS metoda je uspešno rešila sve test instance. U velikom broju testova, rešenje koje je pronađeno hibridnom metodom se poklapa sa optimalnim rešenjem koje je dao rešavač CPLEX. Na problemima dimenzija za koje egzaktna metoda nije adekvatna, rešenja dobijena hibridnom heuristikom su takođe prilično dobrog kvaliteta.

Performanse predloženih heurističkih metoda su upoređene na osnovu četiri kriterijuma: t_{best}, t_{tot}, $agap$ i σ. U narednim podsekcijama je izložena analiza dobijenih rezultata za svaku od četiri grupe test instanci.

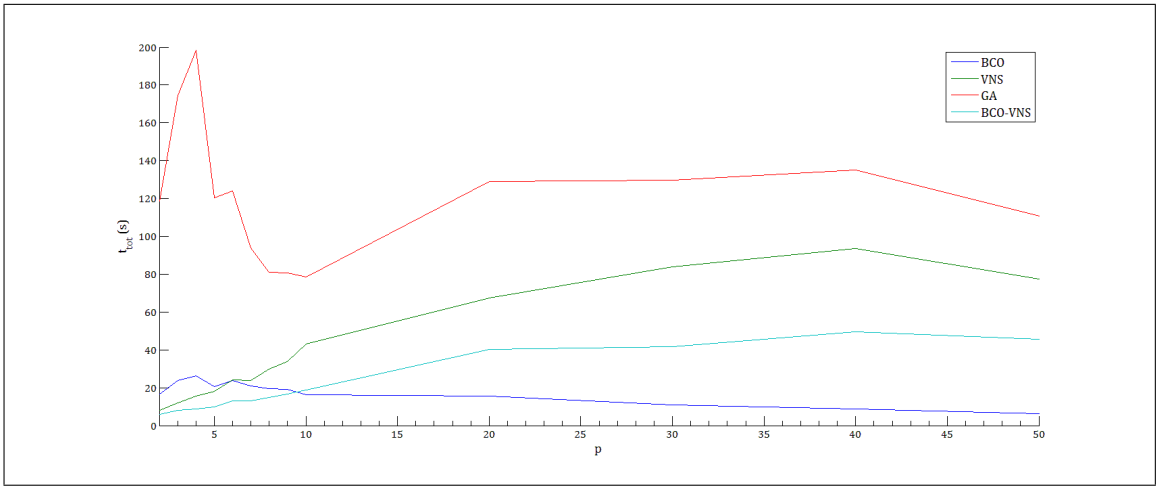
5.1 Analiza eksprimentalnih rezultata na instancama malih dimenzija

Grupa M sadrži 13 test instanci malih dimenzija. Vrednosti parametara su sledeće: broj korisnika $|I|$ je jednak 1000, broj potencijalnih lokacija $|J|$ je 100, dok parametar p uzima vrednosti iz skupa $\{1, 2, 3, ..., 10, 20, 30, 40, 50\}$. Na slici 5.1 predstavljen je grafik zavisnosti prosečnog vremena neophodnog algoritmu da pronađe najbolje rešenje od parametra p. Na slici 5.2 je predstavljen grafik zavisnosti ukupnog utrošenog vremena od parametra p.



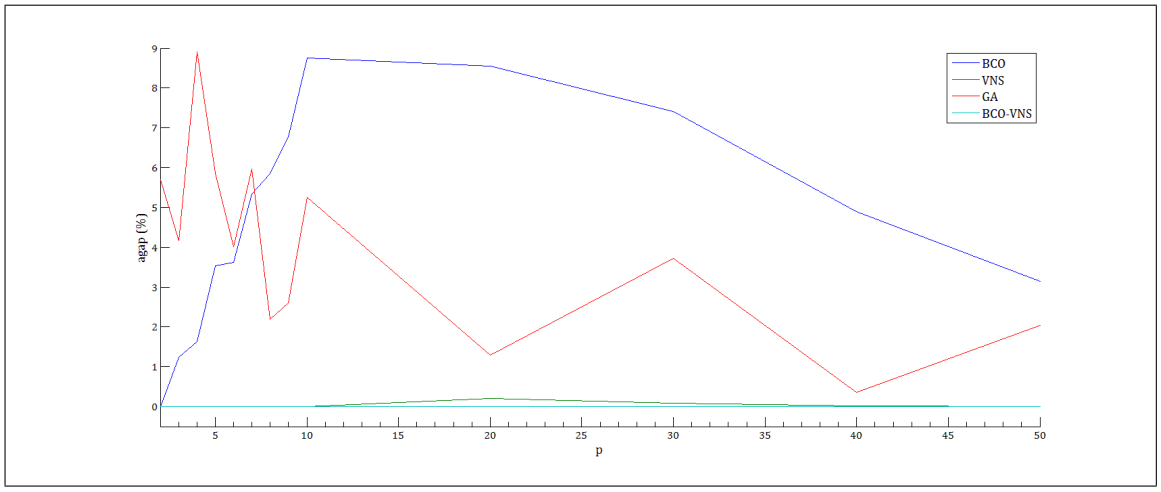
Slika 5.1: Poređenje vremena pronalaska najboljeg rešenja na instancama malih dimenzija

Poređenjem predloženih implemenatacija prema ukupnom utrošenom vremenu, može se zaključiti da su najbolji rezultati postignuti hibridnom i BCO metodom. U proseku, ukupno vreme potrebno za izvršavanje BCO i BCO-VNS metoda je oko 20 sekundi. VNS metoda je nešto sporija, sa prosečnim ukupnim vremenom izvršavanja oko 40 sekundi. BCO, VNS i BCO-VNS metodama je do prvog pronalaska najboljeg rešenja u proseku trebalo oko 8 sekundi. Ukupno vreme izvršavanja GA metode je u proseku 120 sekundi, a za pronalazak najboljeg rešenja 65 sekundi.

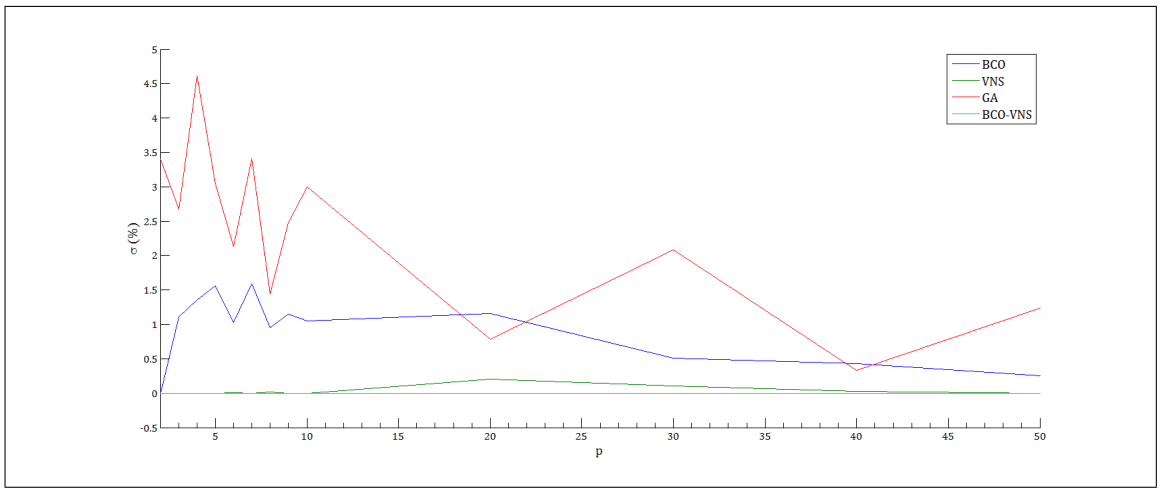


Slika 5.2: Poređenje vremena izvršavanja na instancama malih dimenzija

CPLEX rešavač je pronašao optimalno rešenje za sve instance M grupe. Rešenja koja su dobijena heurističkim metodama se zbog toga mogu uporediti sa optimalnim. Analizom vrednosti *agap* i *sigma* se dolazi do sledećih zaključaka. Najbolji kvalitet rešenja je postignut hibridnom metodom, sa prosečnim odstupanjem od 0 procenata. VNS metode je takođe pronašla optimalno rešenje na svakoj instanci, ali uz prosečno odstupanje od 0.02%. GA i BCO metodama su dobijena rešenja koja u proseku odstupaju od optimalnog oko 4%. Zavisnost *agap* od vrednosti parametra p je predstavljena na grafiku 5.3, dok je zavisnost standardne devijacije od p predstavljena na slici 5.4.



Slika 5.3: Poređenje prosečnog odstupanja na instancama malih dimenzija

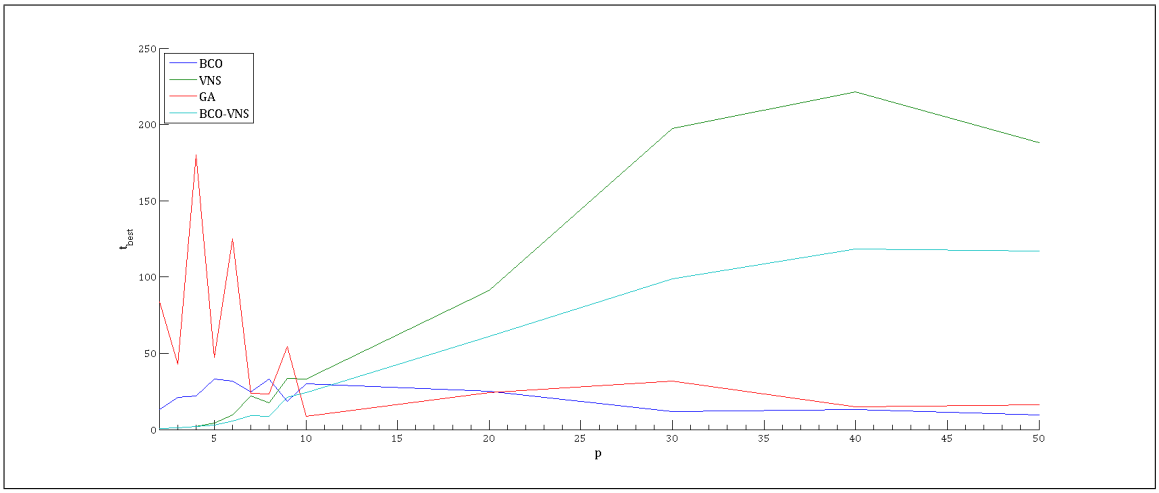


Slika 5.4: Poređenje standardne devijacije na instancama malih dimenzija

5.2 Analiza eksprimentalnih rezultata na instancama srednjih dimenzija

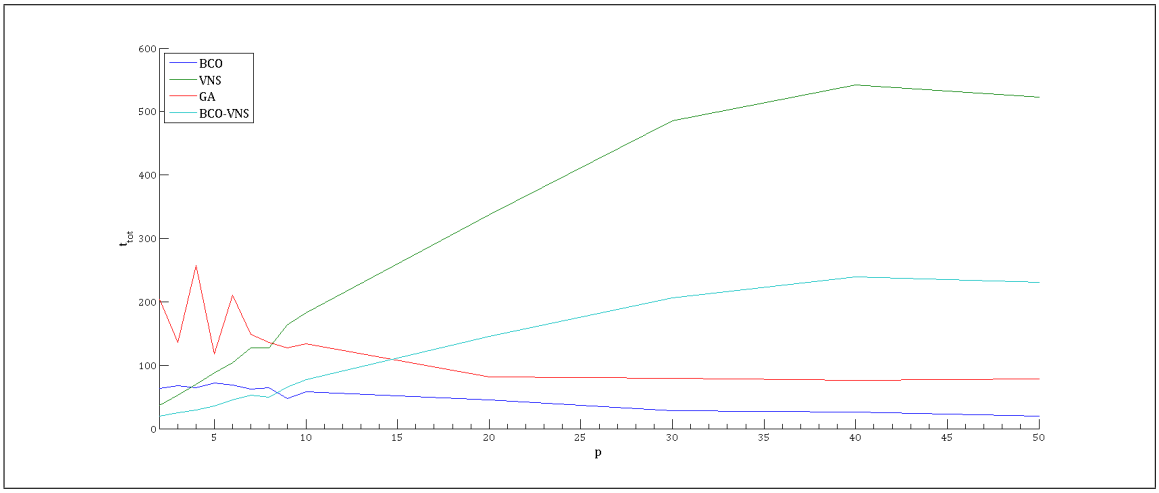
Instance srednjih dimenzija imaju sledeće vrednosti ulaznih parametara: broj korisnika $|I|$ je 3000, broj potencijalnih lokacija $|J|$ je 100, dok broj objekata koje treba uspostaviti p uzima vrednosti iz skupa $\{1, 2, 3, ..., 10, 20, 30, 40, 50\}$.

Na slici 5.5 je predstavljena zavisnost vremena za koje heuristika prvi put pronađe najbolje rešenje od parametra p. Najkraće vreme do pronalaska najboljeg rešenja bilo je u slučaju BCO metode kojoj je proseku trebalo 22 sekunde. Predloženoj hibridnoj metodi je u proseku bilo potrebno 36 sekundi, GA metodi 52 sekunde, a VNS metodi 63 sekunde.



Slika 5.5: Poređenje vremena pronalaska najboljeg rešenja na instancama srednjih dimenzija

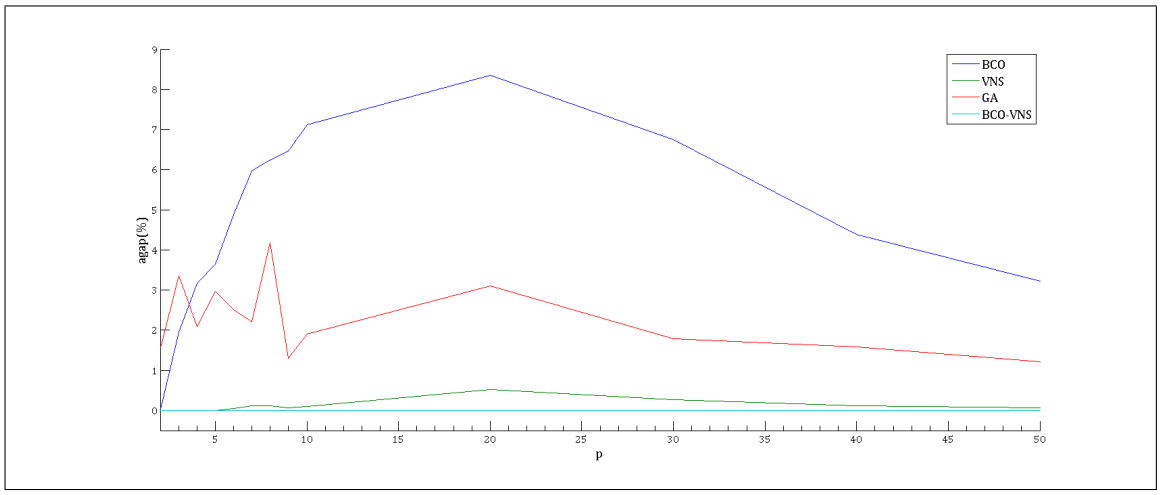
Predložena implementacija BCO metode je takođe najbolja prema kriterijumu ukupnog vremena izvršavanja. Instance iz S grupe je BCO metoda rešila u proseku za 50 sekundi. Hibridnoj BCO-VNS metodi je za iste instance trebalo u proseku 90 sekundi, a GA metodi 137 sekundi. Najviše vremena za izvršavanje, u proseku 210 sekundi, je bilo potrebno VNS metodi. Na 5.6 je predstavljen grafik zavisnosti vremena izvršavanja predloženih metoda od parametra p.



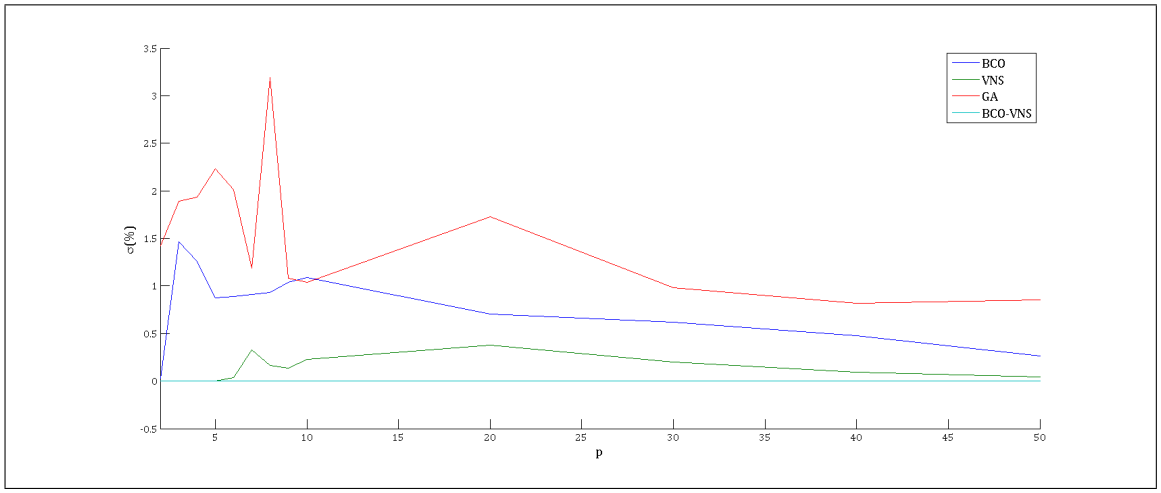
Slika 5.6: Poređenje vremena izvršavanja na instancama srednjih dimenzija

Za sve instance srednjih dimenzija, rešenja koja su dobijena heurističkim metodama se mogu uporediti sa optimalnim rešenjima koja su pronađena egzaktnom metodom. Najmanje odstupanje od optimuma je postignuto predloženom hibridnom metodom, sa prosečnom vrednošću *agap* od $0,001\%$. Rešenja koja je pronašla VNS metoda u proseku odstupaju od optimalnog za 0.02%. Rešenja dobijena GA i

BCO metodama su nešto slabijeg kvaliteta i proseku odstupaju od egzaktnog za 2, odnosno 4 procenta. Zavisnost vrednosti *agap* od parametra p je predstavljena na slici 5.7., dok je zavisnost standardne devijacije od p predstavljena na slici 5.8.



Slika 5.7: Poređenje prosečnog odstupanja na instancama srednjih dimenzija



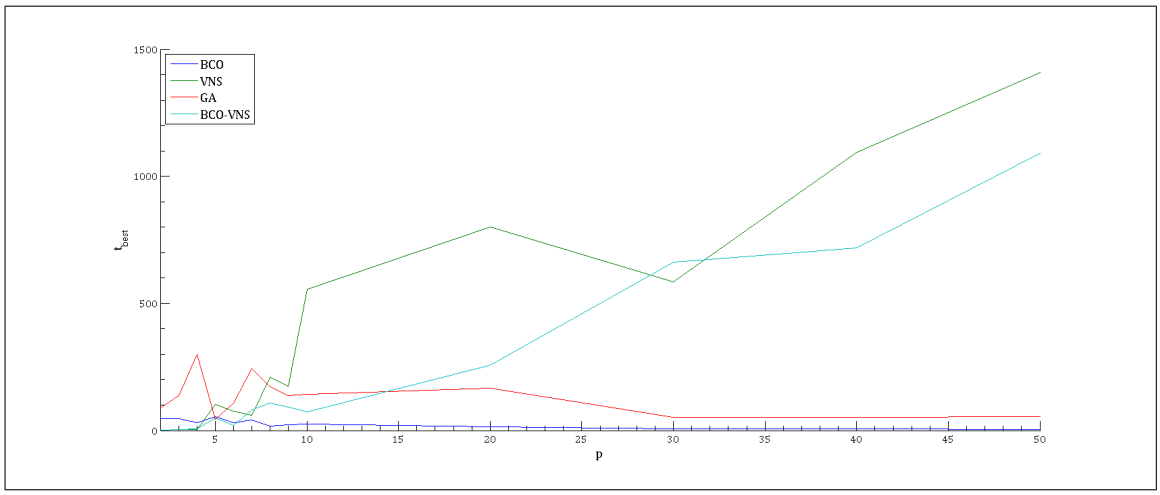
Slika 5.8: Poređenje standardne devijacije na instancama srednjih dimenzija

5.3 Analiza eksprimentalnih rezultata na instancama velikih dimenzija

Instance većih dimenzija imaju sledeće vrednosti ulaznih parametara: broj korisnika $|I|$ je 3000, broj potencijalnih lokacija $|J|$ je 300, a broj objekata koje treba otvoriti p uzima vrednosti iz skupa $\{1, 2, 3, ..., 10, 20, 30, 40, 50\}$. CPLEX rešavač je uspešno rešio 8 od 13 instanci grupe V. Predložene heurističke metode su rešile svih 13 instanci ove grupe.

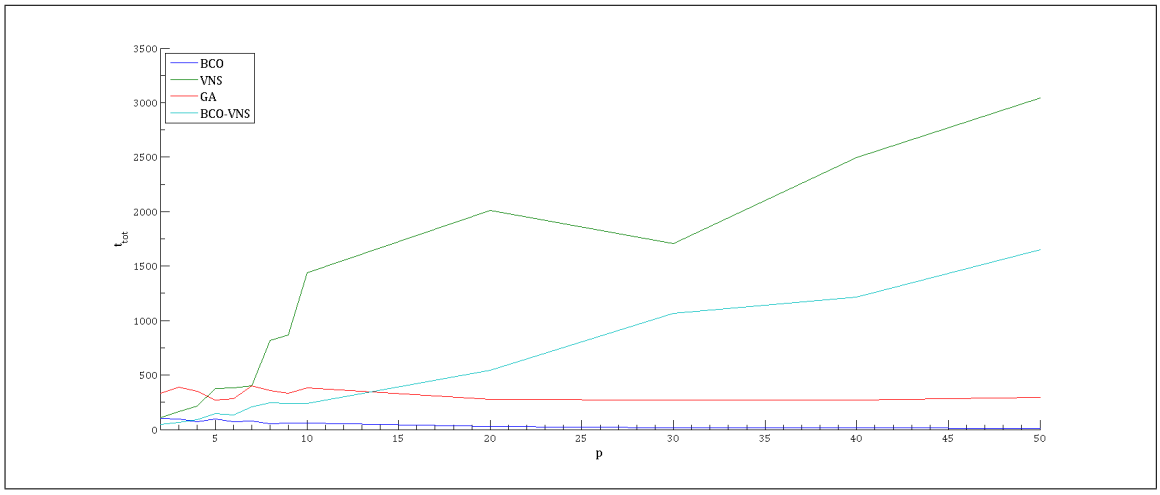
BCO metoda je u proseku nakon 26 sekundi prvi put pronalazila najbolje rešenje. Ostale tri predložene metode su zahtevale više vremena. Posle BCO metode, najmanje vremena je bilo potrebno GA metodi - u proseku 130 sekundi. Predloženoj hibridnoj metodi je u proseku bilo potrebno 243 sekunde, a VNS metodi 390 sekundi.

Na slici 5.9 je prikazano u kojoj meri vreme za koje heuristika prvi put pronađe najbolje rešenje zavisi od parametra p.



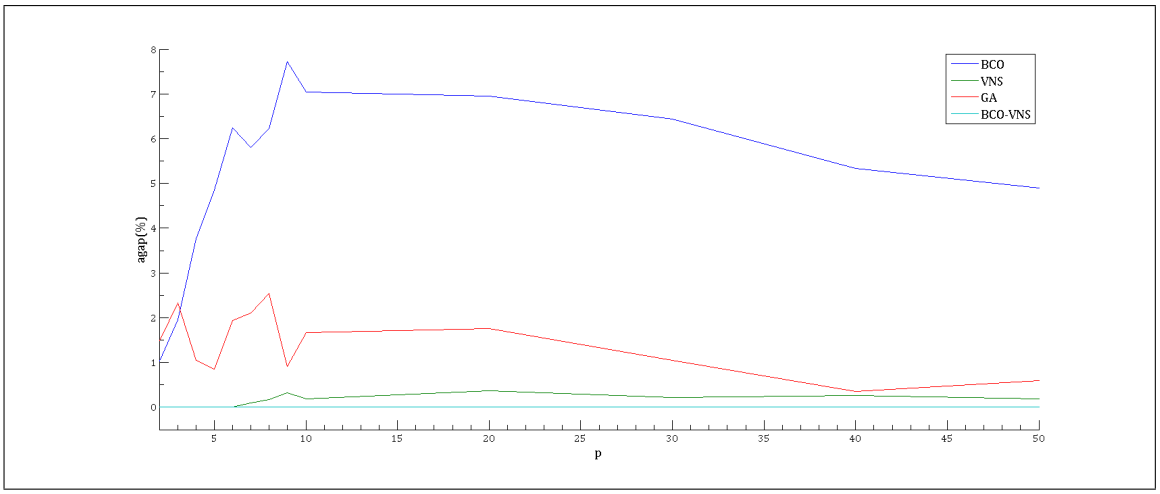
Slika 5.9: Poređenje vremena pronalaska najboljeg rešenja na instancama velikih dimenzija

Grafik zavisnosti ukupnog vremena izvršavanja predloženih metoda od parametra p je predstavljen na slici 5.10. Predložena implementacija BCO metode se na instancama velikih dimenzija izvršavala u proseku 60 sekundi, GA metoda 324 sekunde, hibridna BCO-VNS metoda 453 sekunde, a VNS metoda 1079 sekundi.

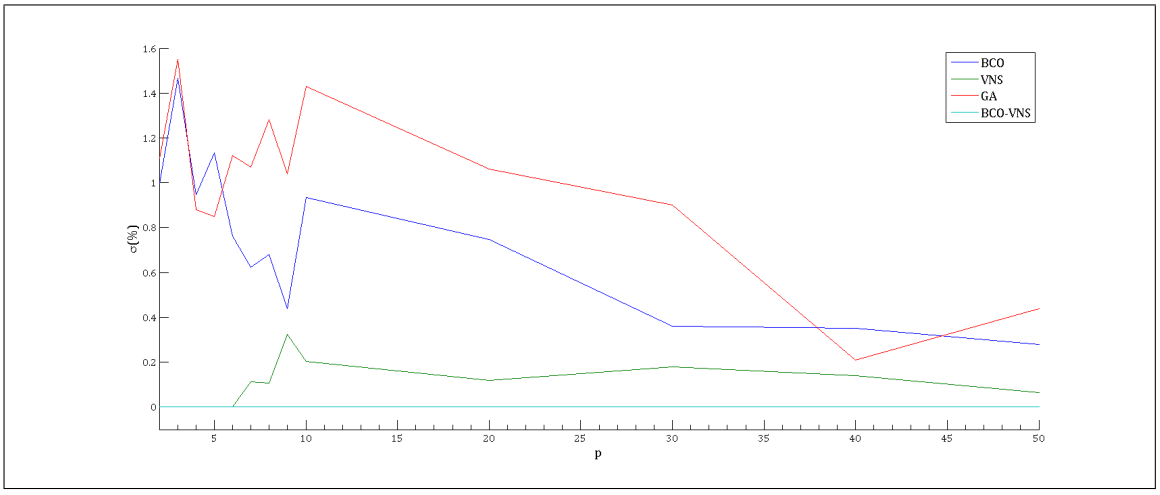


Slika 5.10: Poređenje vremena izvršavanja na instancama velikih dimenzija

Rešenja koja su dobijena heurističkim metodama su upoređena sa najboljim poznatim rešenjem date instance. Za pet instanci koje nisu uspešno rešene egzaktnom metodom se ne može garantovati optimalnost pronađenog rešenja. Međutim, na tim instancama su VNS i predložena hibridna metoda pronašle identična rešenja. Iz tog razloga je velika verovatnoća da je heuristikama pronađeno optimalno rešenje i na ovim MAXCAP instancama. Najmanje odstupanje od najboljeg poznatog rešenja je pokazala predložena BCO-VNS metoda, u proseku $0,001\%$. Prosečno odstupanje rešenja dobijenih VNS metodom je 0.14%, rešenja dobijenih GA metodom 1.4%. Rešenja dobijena BCO metodom su najslabijeg kvaliteta i proseku odstupaju 5% od najboljeg poznatog rešenja. Grafik zavisnosti vrednosti *agap* od parametra p je predstavljena na slici 5.11, a zavisnost standardne devijacije od p na slici 5.12.



Slika 5.11: Poređenje prosečnog odstupanja na instancama velikih dimenzija



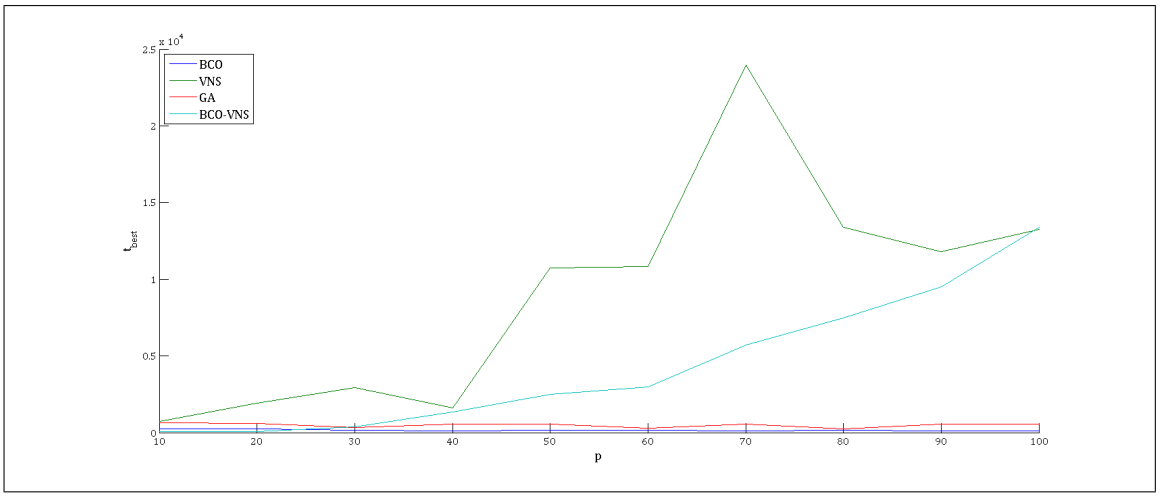
Slika 5.12: Poređenje standardne devijacije na instancama velikih dimenzija

5.4 Analiza eksprimentalnih rezultata na slučajno generisanim instancama

Poslednju grupu čine slučajno generisane instance velikih dimenzija: broj korisnika $|I|$ je 10000, broj potencijalnih lokacija $|J|$ je 1000. Broj objekata koje treba uspostaviti p uzima vrednosti iz skupa $\{10, 20, ..., 90, 100\}$. Nijedna test instanca Rnd grupe nije rešena egzaktnom metodom pod postavljenim vremenskim i memorijskim ograničenjima, dok su predložene heurističke metode uspešno rešene svih deset Rnd instanci.

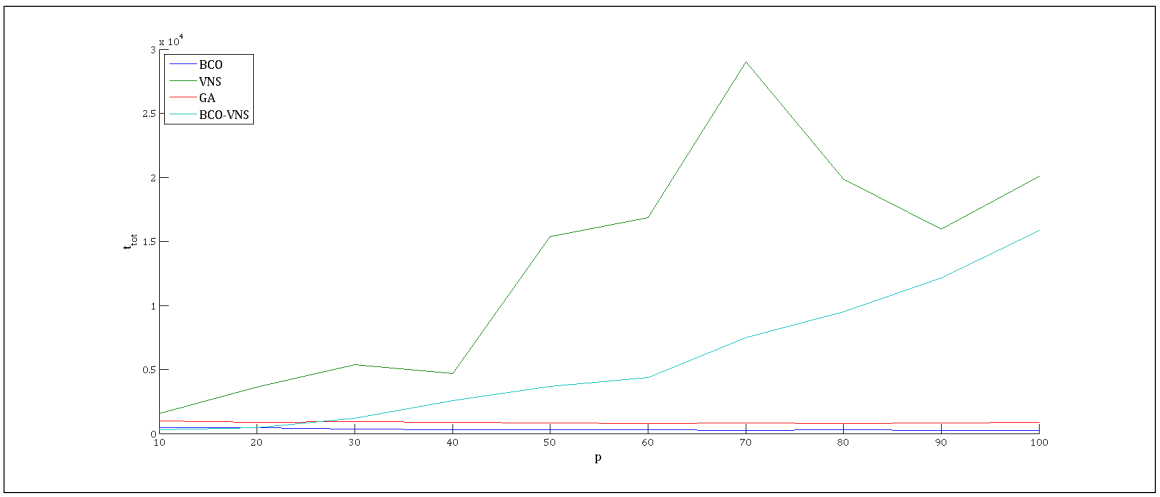
BCO i GA metoda su u proseku zahtevale značajno manje vremena za pronalazak najboljeg rešenja od ostale dve predložene metode. Na slici 5.13 je prikazana zavisnost prosečnog vremena koje je potrebno da heuristika prvi put pronađe najbolje rešenje od parametra p.

Grafik zavisnosti ukupnog vremena izvršavanja predloženih metoda od parametra p je predstavljen na slici 5.14. Predloženoj implementaciji BCO metode je trebalo najmanje vremena, a zatim GA metodi. Dok je za veće vrednosti parametra p, prosečno vreme do pronalaska najboljeg rešenja hibridne BCO-VNS metode i VNS metode slično, ukupno vreme izvršavanja hibridne metode je kraće od VNS metode.



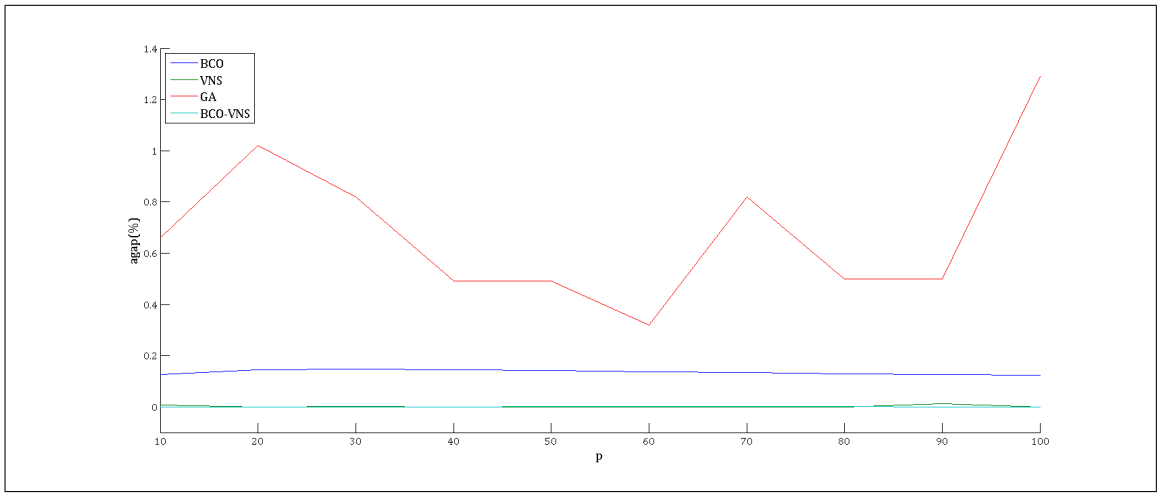
Slika 5.13: Poređenje vremena pronalaska najboljeg rešenja na slučajno generisanim instancama

Na svim instancama Rnd grupe, predložene heuristike su se izvršavale kraće od vremenskog ograničenja rešavača CPLEX koje je postavljeno na 4 sata.

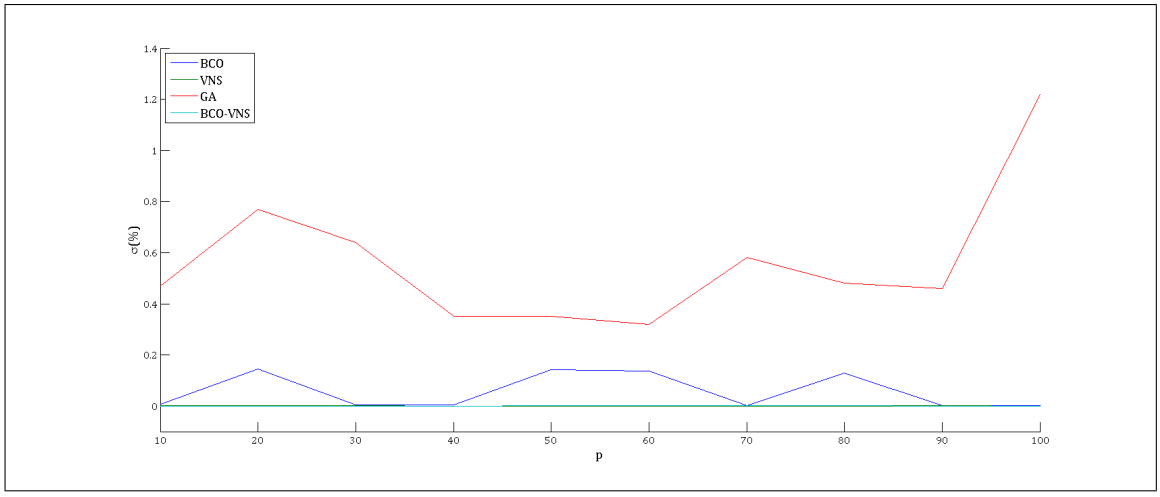


Slika 5.14: Poređenje vremena izvršavanja na slučajno generisanim instancama

Rešenja koja su dobijena heurističkim metodama su upoređena sa najboljim poznatim rešenjem, ali čija optimalnost nije potvrđena egzaktnom metodom. Na velikom broju instanci, VNS i predložena BCO-VNS metoda pronašle su ista rešenja, pa postoji šansa da je u pitanju optimalno rešenje iako se to ne može garantovati. Najbolje rešenje je uglavnom dobijeno predloženom hibridnom metodom koja je pokazala i najmanje odstupanje od najboljeg poznatog rešenja, u proseku $0,001\%$. Prosečno odstupanje rešenja dobijenih VNS metodom je 0.002%, dok su rešenja dobijena BCO i GA metodama lošijeg kvaliteta. Grafik zavisnosti vrednosti *agap* od parametra p je predstavljena na slici 5.15, a zavisnost standardne devijacije od parametra p na slici 5.16.



Slika 5.15: Poređenje prosečnog odstupanja na slučajno generisanim instancama



Slika 5.16: Poređenje standardne devijacije na slučajno generisanim instancama

6
Zaključak

Predmet istraživanja ovog rada je problem maksimalnog zauzimanja tržišta. MAXCAP predstavlja izuzetno koristan model koji je od velikog praktičnog značaja u brojnim oblastima. Izvršenim testiranjima na instancama problema različitih dimenzija je pokazano da egzaktne metode uglavnom nisu adekvatne za rešavanje MAXCAP instanci velikih dimenzija. U radu su predložene četiri heurističke metode koje omogućavaju rešavanje instanci problema velikih dimenzija: optimizacija kolonijom pčela, metoda promenljivih okolina i genetski algoritam i hibridizacija metode promenljivih okolina i optimizacije kolonijom pčela. Hibridizacija te dve metaheurističke metode je prvi put predložena u ovom radu. Elementi svake od četiri predložene metode su prilagođene karakteristikama razmatranog MAXCAP problema. Analiza performansi heurističkih metoda je pokazala da je predložena BCO-VNS metoda uspešnija od preostale tri metahurističke metode implementirane u ovom radu u smislu kvaliteta rešenja, dok je brzina izvršavanja zadovoljavajuća. Buduća istraživanja bi se mogla koncentrisati na dalje poboljšanje efikasnosti BCO-VNS metode i na njeno prilagođavanje rešavanju drugih varijanti problema MAXCAP.

Literatura

[1] Alp, O., Erkut, E. & Drezner, Z. "An efficient genetic algorithm for the p-median problem". *Annals of Operations research* 122.1 (2003), 21–42.

[2] Alp, O., Erkut, E. & Drezner, Z. "Solving the uncapacitated hub location problem using genetic algorithms". *Computers & Operations Research* 32.4 (2005), 967–984.

[3] Beasley, J.E. "Obtaining test problems via internet". *Journal of Global Optimization* 8.4 (1996), 429–433.

[4] Correa, E.S., Steiner, M.T.A., Freitas, A.A. & Carnieri, C. "A genetic algorithm for solving a capacitated p-median problem". *Numerical Algorithms* 35.2 (2004), 373–388.

[5] Darwin, C. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, 1859.

[6] Davidović, T., D., Ramljak, Šelmić, M. & Teodorović, D. "Bee colony optimization for the p-center problem". *Computers and Operations Research* 38.10 (2011), 1367–1376.

[7] Đenić, A., Marić, M., Stanimirović, Z. & Stanojević, P. "A variable neighbourhood search method for solving the long-term care facility location problem". *Journal of Management Mathematics* 206 (2016).

[8] Drezner, Z. & Hamacher, H. W., eds. *Facility location: applications and theory*. Springer Science & Business Media, 2001.

[9] Drezner, Z., Klamroth, K., Schöbel, A. & Wesolowsky, G. O. "The Weber Problem". In: *Facility Location: Applications and Theory*. Ed. by Drezner, Z. & Hamacher, H.W. Springer, 2001, pp. 1–36.

[10] Drezner, Z. & Suzuki, A. "The Big Triangle Small Triangle Method for the Solution of Nonconvex Facility Location Problems". *Operations Research* 52.1 (2004), 128–135.

[11] Farahani, R. Z., Abedian, M. & S., Sharahi. "Dynamic facility location problem". In: *Facility Location*. Ed. by Z., Farahani R. & H., Masoud. Physica-Verlag HD, 2009, pp. 347–372.

[12] Fogel, L.J., A.J., Ovens & Walsh, M. J. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, 1966.

[13] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[14] Hakimi, S.L. "On locating new facilities in a competitive environment". *European Journal of Operational Research* 12.1 (1983), 29–35.

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

[16] Hansen, P. & Mladenović, N. "Variable neighborhood search". In: *Search methodologies*. Ed. by E., Burke & G., Kendall. Springer, 2014, pp. 313–337.

[17] Hansen, P. & Mladenović, N. "Variable neighborhood search for the p-median". *Location Science* 5.4 (1997), 207–226.

[18] Hansen, P. & Mladenović, N. "Variable neighborhood search: Principles and applications". *European journal of Operational Research* 130.3 (2001), 449–467.

[19] Hansen, P., Mladenović, N. & Pérez, J.A.M. "Variable neighbourhood search: methods and applications". *Annals of Operations Research* 175.1 (2010), 367–407.

[20] Holland, J. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[21] Hotelling, H. "Stability in Competition". *Economic Journal* 39.153 (1929), 41–57.

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

[23] Karakitsiou, A. *Modeling Discrete Competitive Facility Location*. Springer, 2015.

[24] Karakitsiou, A. & Migdalas, A. "Locating facilities in a competitive environment". *Optimization Letters* (2016).

[25] Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[26] Lučić, P. & Teodorović, D. "Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence". *TRISTAN IV Triennial Symposium on Transportation Analysis* (2001), 441–445.

[27] Marić, M., Stanimirović, Z., Đenić, A. & Stanojević, P. "Memetic Algorithm for Solving the Multilevel Uncapacitated Facility Location Problem". *Informatica* 25.3 (2014), 439–466.

[28] Marić, M., Stanimirović, Z. & Stanojević, P. "An efficient memetic algorithm for the uncapacitated single allocation hub location problem". *Soft Computing* 17.3 (2013), 445–466.

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

[30] Mladenović, N. & Hansen, P. "Variable neighborhood search". *Computers and Operations Research* 24.11 (1997), 1097–1100.

[31] Mladenović, N., Labbe, M. & Hansen, P. "Solving the p-Center problem with Tabu Search and Variable Neighborhood Search". *Networks* 42.1 (2003), 48–64.

[32] Mladenović, N., Todosijević, R. & Urošević, D. "Two level General variable neighborhood search for Attractive traveling salesman problem". *Computers & Operations Research* 206.52 (2014), 341–348.

[33] Peeters, D., Hansen, P., Thisse, J. & D., Richard. "The Minisum and Minimax Location Problems Revisited". *Operations Research* 33.6 (1985), 1251–1265.

[34] Plastria, F. "Continuous Covering Location Problems". In: *Facility Location: Applications and Theory*. Ed. by Drezner, Z. & Hamacher, H.W. Springer, 2001, pp. 37–80.

[35] Rechenberg, I. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, 1973.

[36] Reeves, C.R. *Genetic Algorithms, Modern Heuristic Techniques for Combinatorial Problems*. John Willy and Sons, 1993.

[37] ReVelle, C. "The maximum capture or sphere of influence problem: hotelling revised on a network". *Journal of Regional Science* 26.2 (1986), 343–357.

[38] ReVelle, C. & Serra, D. "The maximum capture problem including reallocation". *Information Systems and Operational Research* 29.2 (1991), 130–138.

[39] Rodríguez, C. C., Moreno-Pérez, J.A. & Santos-Penate, D. R. "Particle swarm optimization with two swarms for the discrete $(r|p)$-centroid problem". In: *Computer Aided Systems Theory – EUROCAST 2011*. Ed. by Moreno-Díaz, R., Pichker, F. & Quesada-Arencibia, A. Springer, 2011, pp. 432–439.

[40] Schwefel, H.P. *Numerische Optimierung von Computer-modellen mittels der Evolutionsstrategie*. Birkhäuser Verlag, 1977.

[41] Šelmić, M., Teodorović, D. & Davidović, T. "Bee Colony Optimization part I: The algorithm overview". *Yugoslav Journal of Operations Research* 25.1 (2015), 33–56.

[42] Serra, D. & Colome, R. "Consumer choice and optimal locations models: Formulations and heuristics". *Papers in Regional Science* 80.4 (2001), 439–464.

[43] Serra, D. & ReVelle, C. "Competitive location and pricing on networks". *Geographical Analysis* 31.2 (1995), 109–129.

[44] Serra, D. & ReVelle, C. "Competitive location in discrete space". In: *Facility Location: A Survey of Applications and Methods*. Ed. by Z., Drezner. Springer, 1995.

[45] Sörensen, K., Sevaux, M. & Glover, F. "A History of Metaheuristics". In: *Handbook of Heuristics*. Ed. by R., Marti, P., Pardalos & M., Resende. (to be published). Springer, 2016.

[46] Stackelberg, H. von. *Marktform und Gleichgewicht (Market Structure and Equilibrium),* trans. by Bazin, D., Hill, R. & Urch, L. Springer, 2011.

[47] Stanimirović, Z., Kratica, J., Filipović, V. & Tošić, D. *Evolutivni pristup za rešavanje hab lokacijskih problema*. Zavod za udžbenike i nastavna sredstva, 2011.

[48] Stanimirović, Z., Marić, M., Božović, S. & Stanojević, P. "An Efficient Evolutionary Algorithm for Locating Long-Term Care Facilities". *Information Technology and Control* 41.1 (2012), 77–89.

[49] Suárez-Vega, R., Santos-Penate, D. R. & Dorta-González, P. "Discretization and resolution of the $(r|X_p)$-medianoid problem involving quality criteria". *Top* 12.1 (2004), 111–133.

[50] Talbi, E. *Metaheuristics from design to implementation*. John Wiley & Sons, Inc., 2009.

[51] Tobin, R. & Friesz, T.L. "Spatial competition facility location models: definitions, formulations and solution approach". *Annals of Operations Research* 6.3 (1986), 49–74.