UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Stefan Z. Mišković

REŠAVANJE KLASE MIN-MAX PROBLEMA ROBUSNE DISKRETNE OPTIMIZACIJE SA PRIMENAMA

doktorska disertacija

Beograd, 2016.

UNIVERSITY OF BELGRADE

FACULTY OF MATHEMATICS

Stefan Z. Mišković

SOLVING THE MIN-MAX ROBUST DISCRETE OPTIMIZATION PROBLEMS WITH APPLICATIONS

Doctoral Dissertation

Belgrade, 2016

Mentor:

dr Zorica Stanimirović
vanredni profesor
Matematički fakultet – Univerzitet u Beogradu

Članovi komisije:

dr Miodrag Živković
redovni profesor
Matematički fakultet – Univerzitet u Beogradu

dr Miroslav Marić
vanredni profesor
Matematički fakultet – Univerzitet u Beogradu

dr Tatjana Davidović
viši naučni saradnik
Matematički institut SANU

Datum odbrane: _____

Zahvalnost

Želeo bih da se zahvalim svom mentoru, prof. dr Zorici Stanimirović, na nesebičnoj podršci koju mi je pružala tokom naše dugogodišnje saradnje. Svojim savetima, znanjem i iskustvom je u velikoj meri doprinela definisanju mojih istraživačkih interesovanja, kao i realizaciji ove disertacije. Takođe, zahvalio bih se članovima komisije, prof. dr Miodragu Živkoviću, prof. dr Miroslavu Mariću i dr Tatjani Davidović na pažljivom čitanju rukopisa i korisnim sugestijama koje su doprinele kvalitetu ovog rada. Posebnu zahvalnost dugujem porodici, prijateljima i svim dragim osobama, koje su na različite načine podržali moje profesionalno usavršavanje.

Naslov disertacije: Rešavanje klase min-max problema robusne diskretne optimizacije sa primenama

Rezime: U ovoj disertaciji razmatrana su tri NP-teška problema diskretne optimizacije sa funkcijom cilja tipa min-max: višeperiodni lokacijski problem za raspoređivanje jedinica za reagovanje u hitnim slučajevima, dinamički lokacijski problem maksimalnog pokrivanja sa više poluprečnika, i problem p-hab centra neograničenih kapaciteta sa višestrukim alokacijama. Kako u situacijama u praksi ulazni podaci (zahtevi korisnika, cena ili vreme transporta) često variraju po nepoznatim raspodelama, u razmatrane determinističke varijante problema je uključena nepouzdanost ulaznih podataka koristeći pristup robusne optimizacije. Razvijeni su matematički modeli za odgovarajuće determinističke i robusne varijante problema, osim u slučaju determinističke varijante problema p-hab centra neograničenih kapaciteta sa višestrukim alokacijama, koja je već razmatrana u literaturi. Dodatno, za lokacijski problem raspoređivanja jedinica za reagovanje u hitnim slučajevima, prvi put u literaturi je dokazano da je NP-težak.

Razmatrani problemi i njihove robusne varijante imaju veliki praktični značaj, imajući u vidu brojne oblasti primene, kao i da razmatrana nepouzdanost ulaznih parametara odgovara čestim situacijama u praksi. Višeperiodni lokacijski problem za raspoređivanje jedinica za reagovanje u hitnim slučajevima nalazi primenu u optimalnom raspoređivanju stanica hitne pomoći, policijskih i vatrogasnih jedinica ili drugih jedinica hitnih službi u okviru određene oblasti. Dinamički lokacijski problem maksimalnog pokrivanja sa više poluprečnika pomaže pri optimalnom odabiru lokacija za izgradnju resursa (fabrika, uslužnih centara, službi hitne pomoći, i sl.), koji bi maksimalno zadovoljili potrebe stanovništva neke oblasti, pri čemu udaljenost resursa do korisnika (odabrani poluprečnik pokrivanja) direktno utiče na efikasnost snabdevanja. Problem p-hab centra neograničenih kapaciteta sa višestrukim alokacijama ima značajne primene pri optimizaciji telekomunikacionih sistema, sistema isporuke poštanskih i drugih pošiljaka, hitnih službi, sistema snabdevanja, itd.

Kako egzaktne metode mogu rešiti samo instance problema manjih dimenzija, razvijeni su hibridni algoritmi za rešavanje determinističkih i robusnih varijanti razmatranih problema. Predloženi algoritmi su dobijeni kombinovanjem optimizacije rojem čestica i dva algoritma zasnovana na lokalnom pretraživanju – klasične lokalne pretrage i metode promenljivih okolina. Za problem dinamičkog lokacijskog problema maksimalnog pokrivanja sa više poluprečnika, izvršena je hibridizacija metaheuristike sa egzaktnom metodom zasnovanom na linearnom programiranju. Svi elementi predloženih hibridnih algoritama su prilagođeni osobinama razmatranih problema. Pri-

menjene su razne strategije za ubrzanje predloženih algoritama, posebno pri računanju funkcije cilja i heuristika lokalnog pretraživanja. Detaljno je analiziran uticaj različitih vrednosti parametara hibridnih algoritama na kvalitet dobijenih rešenja. Adekvatne vrednosti značajnih paramatara su određene analizom varijanse.

Za svaki od razmatranih problema (za deterministička i robusnu varijantu), izvršena su testiranja predloženih hibridnih algoritama na odgovarajućim instancama i eksperimentalna poređenja sa postojećim algoritmima iz literature ili egzaktnim metodama koje su ugrađene u rešavač CPLEX. Dobijeni eksperimentalni rezultati ukazuju na efikasnost predloženih algoritama pri dobijanju visoko kvalitetnih rešenja. Na osnovu rezultata poređenja sa postojećim metodama za rešavanje razmatranih problema, mogu se uočiti prednosti predloženih hibridnih algoritama u smislu kvaliteta rešenja i/ili brzine izvršavanja, što se posebno ogleda u slučaju instanci problema većih dimenzija.

Rezultati prikazani u ovoj disertaciji predstavljaju doprinos oblastima diskretne optimizacije, robusne optimizacije i metaheurističkih metoda.

Ključne reči: diskretna optimizacija, robusna optimizacija, min-max problemi, metaheurističke metode

Naučna oblast: računarstvo

Uža naučna oblast: optimizacija

UDK broj: 519.874(043.3)

Dissertation title: Solving the min-max robust discrete optimization problems with applications

Abstract: In this dissertation, three NP-hard min-max discrete optimization problems are considered. The first considered problem is multi-period emergency service location problem, the second one is dynamic maximal covering location problem with multiple covering radii, and the third one is uncapacitated multiple allocation p-hub center problem. In many practical situations, input parameters (such as user demands, transportation time or cost) often vary with unknown distributions. Therefore, it is necessary to involve these uncertainties in the deterministic variants of the problems by applying robust optimization approach. Mathematical models for the deterministic and non-deterministic variants of all three problems are developed, except for the deterministic uncapacitated multiple allocation p-hub center problem, which has already been addressed in the literature. In addition, for the first time in the literature, it was proven that the emergency service location problem is NP-hard.

The considered problems and their robust variants have numerous applications, due to the fact that in real-life situations input parameters are often subject to uncertainty. Multi-period emergency service location problem may be used when determining optimal locations for police stations, fire brigades, ambulances, and other emergency units in the given region. The dynamic maximal covering location problem with multiple covering radii is useful when choosing the optimal strategy for establishing resources (service centers, suppliers, facilities, etc.) with maximal satisfaction of customer demands in a certain region, by assuming that the service efficiency directly depends on the distance between customer and service center (i.e., the selected coverage radius). The uncapacitated multiple allocation p-hub center problem has significant applications in designing telecommunication and transportation networks, postal delivery systems, emergency systems, supply networks, etc.

Since exact methods provide optimal solutions only for problem instances of small dimensions, hybrid metaheuristic algorithms are developed to solve both deterministic and robust variants of the considered problems. The proposed hybrid algorithms are obtained by combining particle swarm optimization, with local search heuristic – classical local search or variable neighborhood search method. For dynamic maximal covering location problem with multiple covering radii, a hybridization of metaheuristic algorithm with exact method based on linear programming is developed. All elements of the proposed algorithms are adopted to the problems under consideration. Different strategies are implemented for improving the efficiency of propo-

sed algorithms, especially for the calculation of the objective function value and the local search part. The influence of different parameters of hybrid algorithms on the solution quality is analyzed in detail. All parameters are adjusted by using analysis of variance.

For all considered problems (both deterministic and robust variant), the performance of the proposed hybrid algorithms is evaluated on adequate test data sets. The proposed algorithms are compared with existing heuristic from the literature and exact methods incorporated in commercial CPLEX solver. The obtained experimental results indicate the efficiency of proposed algorithms in obtaining high quality solutions for all considered test instances. The presented comparative analysis indicates the advantages of the proposed hybrid algorithms over existing methods in the sense of solution quality and/or required computational time, especially in the case of large problem dimensions.

The results presented in this paper represent a contribution to the field of discrete optimization, robust optimization and metaheuristic methods.

Keywords: discrete optimization, robust optimization, min-max problems, metaheuristic methods

Scientific field: computer science

Scientific sub-field: optimization

UDC number: 519.874(043.3)

Sadržaj

1 Uvod

Za realnu funkciju $f : S \to \mathbb{R}$ definisanu nad skupom S, problem optimizacije se može definisati na sledeći način: odrediti element $x_0 \in S$ takav da je $f(x_0) \leq f(x)$ za sve elemente x iz skupa S (problem minimizacije), odnosno $f(x_0) \geq f(x)$ za sve elemente x iz skupa S (problem maksimizacije). Skup S je obično podskup skupa $\mathbb{R}^n$ i određen je skupom ograničenja. Ograničenja su predstavljena u vidu jednakosti i nejednakosti koje elementi skupa S moraju da zadovoljavaju. Domen S se još naziva i pretraživački prostor, a njegovi elementi dopustiva rešenja. Funkcija f čija se vrednost minimizuje (odnosno maksimizuje) se naziva funkcija cilja. Rešenje x_0, za koje je vrednost funkcije cilja f minimimalna (odnosno maksimalna) predstavlja optimalno rešenje. U opštem slučaju, može postojati više optimalnih rešenja.

Mnogi praktični problemi se mogu predstaviti kao problemi optimizacije. Tako se, na primer, razni problemi u saobraćaju, inženjerstvu, ekonomiji, molekularnoj biologiji, sistemima navodnjavanja i telekomunikacijama mogu posmatrati kao problemi optimizacije i rešiti nekom od njenih metoda.

Problemi optimizacije se mogu podeliti na više načina:

- Prema tipu skupa S, problemi optimizacije mogu biti diskretni (kombinatorni), kontinualni i mešoviti. Kod diskretne (kombinatorne) optimizacije, skup S je konačan ili prebrojivo beskonačan. Kod kontinualne optimizacije, skup S je neprebrojiv i najčešće predstavlja podskup skupa $\mathbb{R}^n$ za $n \geq 1$. Mešoviti problemi imaju osobine i diskretnih i kontinualnih problema.

- Prema kriterijumu postojanja ograničenja, problemi optimizacije se dele na probleme bez i probleme sa ograničenjima. Kod optimizacije bez ograničenja sve promenljive uzimaju vrednosti u okviru gornjih i donjih granica, koje su unapred fiksirane. Kod optimizacije sa ograničenjima, uvodi se skup ograničenja koji sužava prostor pretraživanja. Time skup S ne mora uvek biti realan u najširem smislu.

- Na osnovu broja funkcija cilja, postoje problemi sa jednom ili više funkcija cilja. Kod problema sa više funkcija cilja, može se desiti da je neop-

hodno minimizovati jedne, a maksimizovati druge funkcije cilja koje se razmatraju, te je u tom slučaju neophodna njihova simultana optimizacija. Ovakvi problemi se nazivaju još i problemima višekriterijumske optimizacije.

- Prema tipu funkcije cilja, problemi optimizacije se dele na probleme minimizacije i probleme maksimizacije. Specijalno, ukoliko funkcija cilja f čija se vrednost minimizuje predstavlja funkciju maksimuma po nekom kriterijumu, radi se o min-max problemima.

- Optimizacioni problemi se mogu podeliti i na linearne i nelinearne. Kod linearnih problema, funkcija cilja i sva ograničenja problema su linearni. Ukoliko su bar jedno ograničenje ili funkcija cilja nelinearni, govori se o nelinearnom optimizacionom problemu.

- Optimizacioni problemi se mogu podeliti na determinističke i nedeterminističke. Kod determinističkih problema, za odgovarajući ulaz, jedinstveno je određena vrednost rešenja, što kod nedeterminističkih problema nije slučaj. Za rešavanje nedeterministčkih problema se najčešće koristi stohastičko programiranje i optimizacija sa probabilističkim uslovima. Kod ova dva načina rešavanja problema se pretpostavlja da su odgovarajuće ulazne raspodele poznate. Za razliku od ova dva pristupa, kod robusne optimizacije to nije slučaj. Imajući u vidu da se određene promenljive mogu menjati unutar unapred fiksiranog intervala, kreira se odgovarajući robusni model, kod koga se razmatra najgori scenario, pri čemu funkcija cilja postaje min-max oblika.

Većina problema optimizacije su teški za rešavanje, u smislu da pripadaju klasi složenosti problema za čije rešavanje je potreban algoritam eksponencijalne složenosti. Za problem kažemo da pripada klasi P, odnosno da je polinomske složenosti, ukoliko postoji algoritam polinomske složenosti koji ga rešava. Problem pripada klasi NP ukoliko se u polinomskom vremenu može utvrditi da li je zadata vrednost rešenje tog problema. Problem je NP-težak ukoliko se svaki problem iz klase NP može u polinomskom vremenu svesti na njega, a NP-kompletan ukoliko je NP-težak i pritom pripada klasi NP. Važi da je klasa P podskup klase NP, ali je u opštem slučaju problem jednakosti ovih klasa i dalje otvoren. Stoga se ni za jedan NP-težak problem ne zna da li postoji algoritam polinomske složenosti koji ga rešava. Optimizacioni problemi su u većini slučajeva NP-teški [86].

Predmet ove disertacije je posebna klasa problema diskretne (kombinatorne) optimizacije, kod kojih se, uz odgovarajuće uslove, minimizuje neka

maksimalna vrednost. Ekvivalentno tome, kod ovih problema se može maksimizovati odgovarajuća minimalna vrednost. Takvi problemi se nazivaju min-max (odnosno max-min) problemima. Svi razmatrani optimizacioni problemi su NP-teški.

1.1 Min-max problemi

Min-max problem se može formulisati na sledeći način:

$$\min_{x \in S} F(x), \tag{1.1}$$

pri čemu je

$$F(x) = \max_{i=1,\ldots,m} f_i(x) \tag{1.2}$$

i $f_i : S \to \mathbb{R}$, $i = 1, \ldots, m$, gde je $S \subseteq \mathbb{R}^n$. Ukoliko je skup S konačan ili prebrojiv, radi se o min-max problemu diskretne optimizacije. Skup S se još naziva i dopustiv skup, a tačka $x \in S$ dopustivo rešenje problema (1.1). Dopustivo rešenje x^* koje zadovoljava uslov

$$F(x^*) = \min_{x \in S} F(x) = \min_{x \in S} \max_{i=1,\ldots,m} f_i(x)$$

naziva se optimalnim. U opštem slučaju može postojati više optimalnih rešenja.

Sa druge strane, max-min problem se u opštem slučaju može definisati sa

$$\max_{x \in S} G(x),$$

gde je

$$G(x) = \min_{i=1,\ldots,m} g_i(x).$$

Imajući u vidu da je

$$\max_{x \in S} F(x) = -\min_{x \in S}(-F(x)),$$

jasno je da se max-min problem jednostavno može svesti na min-max problem.

Specijalno, ukoliko je $S \subseteq \mathbb{Z}^n$, radi se o min-max problemu celobrojnog programiranja. Ako je $f_i(x) = c_i^T x$, $i = 1, \ldots, m$ i $S = \{x \in \mathbb{Z}^n : Ax \leq b\}$, min-max problem linearnog celobrojnog programiranja se definiše na sledeći način:

$$\min_{x\in S}\max_{1\le i\le m} c_i^T x,$$

gde je A realna matrica, a c i b vektori odgovarajućih dimenzija. Slično, min-max problem linearnog programiranja se može definisati sa

$$\min_{x\in S}\max_{1\le i\le m} c_i^T x, \quad S=\{x\in\mathbb{R}^n : Ax\le b\},$$

a 0–1 min-max problem linearnog programiranja sa

$$\min_{x\in S}\max_{1\le i\le m} c_i^T x, \quad S=\{x\in\{0,1\}^n : Ax\le b\}.$$

Dodatno, ako je skup $S=\{x : Ax\le b\}$ takav da su neke od koordinata x_i celobrojne, a ostale realne ili pripadaju nekom skupu $x_j\in S_j$, radi se o mešovitom celobrojnom linearnom programiranju.

U [5] su Bandler i Charalambous pokazali da se bilo koji problem nelinearnog programiranja može svesti na min-max problem. Naime, ako je dat opšti oblik problema nelinearnog programiranja sa

$$\min_{x\in S} f(x),$$

$$g_i(x)\ge 0 \quad i=1,\dots,m,$$

on se može transformisati u problem min-max tipa na sledeći način:

$$\min_{x\in S}\max_{0\le i\le m} f_i(x),$$

$$f_0(x)=f(x),$$

$$f_i(x)=f(x)-t_i g_i(x) \quad i=1,\dots,m,$$

$$t_i>0 \quad i=1,\dots,m.$$

U radu [5] je pokazano da se za dovoljno velike t_i, optimalna rešenja ova dva problema poklapaju.

1.1.1 Neki min-max problemi diskretne optimizacije

Min-max problemi se javljaju u različitim oblastima, među kojima su kontrola resursa, aproksimacija funkcija, teorija igara, saobraćaj, telekomunikacije, itd. [163]. U [43], [49] i [163] se mogu pronaći brojni primeri min-max problema.

U slučaju diskretne optimizacije, među min-max problemima su najčešći lokacijski problemi, problemi pretraživanja, triangulacije, raspodele poslova, pakovanja, itd. Pregled raznih min-max problema diskretne optimizacije se može pronaći u [24].

1.1.1.1 Lokacijski problemi

Među problemima diskretne optimizacije, posebnu grupu čine lokacijski problemi, kod kojih se najčešće, pod odgovarajućim uslovima, minimizuje najveće rastojanje između dva podskupa čvorova u mreži, suma transportnih troškova, ukupno vreme transporta, cene uspostavljanja određenih resursa, i slično. Kada je u pitanju funkcija cilja, lokacijski problemi se dele na min-sum probleme, kod kojih se, uz odgovarajuće uslove, minimizuje suma rastojanja, kao i min-max probleme, kod kojih se obično minimizuje maksimalno rastojanje između određenog fiksiranog čvora i uspostavljenog resursa. Rastojanje obično odgovara ceni transporta između dva čvora ili vremenu koje je neophodno za transport između dva čvora. U nastavku će biti navedeno nekoliko primera lokacijskih problema, čija je funkcija cilja min-max tipa.

Kod problema p-centra, za graf $G = (V, E)$, definisanog nad skupom V od n čvorova u mreži ($n = |V|$) i skupom ivica E, $|E| = m$, potrebno je odrediti kojih p, $p \leq n$ čvorova će biti uspostavljeno tako da je maksimalno rastojanje nekog uspostavljenog i neuspostavljenog čvora minimalno. Drugim rečima, potrebno je odrediti

$$\min_{P \in \mathbb{P}(V):|P|=p} \max_{v \in V} d(v, P),$$

gde je P podskup od p fiksiranih čvorova iz V. Oznaka $\mathbb{P}(V)$ predstavlja partitivni skup skupa V. Kod drugih varijanti ovog problema, graf G je particionisan na dva podgrafa, od kojih prvi sadrži čvorove koji se najčešće nazivaju klijentima, a drugi čvorove koji predstavljaju resurse. Čvorovi koji se uspostavljaju se nalaze isključivo među resursima, a potrebno je minimizovati najveće rastojanje između proizvoljnog para klijent-resurs. Nekad se i svakom čvoru dodeljuje određena težina w_v, $v \in V$, nakon čega funkcija cilja postaje

$$\min_{P\in\mathbb{P}(V):|P|=p} \max_{v\in V} w_v d(v,P).$$

Definisani problem je NP-težak, gde složenost zavisi od p [103]. Ako je parametar p fiksiran, varijanta problema koji ne uključuje težine dodeljene čvorovima se može rešiti u $O(m^p n^p \log p)$, a varijanta koja ih uključuje u vremenskoj složenosti $O(m^p n^p \log^2 p)$ [63].

Među problemima koji imaju min-max funkciju cilja iz grupe hab-lokacijskih problema izdvajaju se razne varijante problema p-hab centra. Kod osnovne varijante problema p-hab centra je potrebno od n čvorova u mreži odrediti kojih p čvorova će postati habovi i svakom ne-hab čvoru dodeliti jedan ili više habova preko kojeg će se vršiti transport, tako da maksimalno rastojanje između bilo kog para čvorova bude minimalno. Put između dva čvora u mreži uvek mora proći kroz bar jedan čvor koji je hab. Ukoliko se svaki čvor dodeljuje tačno jednom habu, radi se o problemu p-hab centra neograničenih kapaciteta sa jednostrukim alokacijama (Uncapacitated single allocation p-hub center problem – USApHCP), a ukoliko je dozvoljena dodela više habova, radi se o varijanti problema sa višestrukim alokacijama (Uncapacitated multiple allocation p-hub center problem – UMApHCP). Obe verzije problema imaju svoju varijantu sa ograničenjima, te se tada govori o problemima p-hab centra ograničenih kapaciteta (Capacitated single allocation p-hub center problem – CSApHCP i Capacitated multiple allocation p-hub center problem – CMApHCP). Dodatno, u nekim verzijama problema, može se svakom čvoru dodeliti cena uspostavljanja. Za sve navedene verzije problema je u [57] dokazano da su NP-teške. Ukoliko su habovi fiksirani, pri varijanti problema neograničenih kapaciteta, u [57] je pokazano da potproblem sa jednostrukim alokacijama ostaje NP-težak, dok je varijanta potproblema sa višestrukim alokacijama polinomske složenosti.

U [149] predložen je diskretan lokacijski problem kod koga je, za dati skup klijenata i resursa, potrebno odrediti koje resurse treba uspostaviti tako da je maksimalna opterećenost resursa minimalna. Svaki klijent se dodeljuje najbližem uspostavljenom resursu, a broj resursa koje je potrebno uspostaviti je unapred fiksiran. Pod opterećenošću resursa j se podrazumeva suma svih vrednosti c_{ij} čvorova i koji su mu dodeljeni. Parametar c_{ij} predstavlja vreme procesuiranja podataka dobijenih od čvora i na resursu j [149].

U [89] razmatra se problem optimalne raspodele zdravstvenih centara u Južnoj Koreji, pri čemu se minimizuje maksimalan broj pacijenata u nekom centru. U [147] je predložen za algoritam za rešavanje problema koji predstavlja hibridizaciju evolutivne metode i metode promenljivih okolina, a u [148] efikisan evolutivni algoritam.

1.1.1.2 Problemi triangulacije

Za dati skup P od n tačaka, triangulacija skupa P predstavlja skup duži od kojih svaka spaja dve tačke iz tog skupa, pri čemu između tih duži nema presecanja sem u tačkama iz skupa P. Najpoznatija varijanta ovog problema je Delunejova triangulacija koja maksimizuje minimalni ugao među svim dopustivim triangulacijama tog skupa tačaka [140]. Za dati skup od n tačaka, ovaj problem se može rešiti u vremenu $O(n \log n)$ [130]. U geometriji postoji i generalizacija ovog problema, tzv. Delunejova triangulacija sa ograničenjima, koja podrazumeva da se određene duži moraju uključiti u sam proces triangulacije [94].

Slično Delunejovoj triangulaciji, postoji i varijanta problema koja minimizuje maksimalni ugao među svim triangulacijama skupa P, tj. određuje

$$\min_{\mathbb{T}(P)} \max_{\alpha \in \mathbb{A}(P)} \alpha,$$

pri čemu je $\mathbb{T}(P)$ skup svih triangulacija nad skupom tačaka P, a $\mathbb{A}(P)$ skup svih unutrašnjih uglova trouglova date triangulacije $\mathbb{T}(P)$. Ovaj problem se može rešiti u vremenu $O(n^2 \log n)$ [52].

Još jedna varijanta problema triangulacije minimizuje maksimalnu dužinu među svim triangulacijama $\mathbb{T}(P)$ skupa P, odnosno određuje

$$\min_{\mathbb{T}(P)} \max_{a \in \mathbb{L}(P)} a,$$

gde je $\mathbb{L}(P)$ skup svih stranica trouglova triangulacije $\mathbb{T}(P)$. Ovaj problem se može rešiti u $O(n^2)$ [53].

1.1.1.3 Problemi raspoređivanja i klasterizacije

Među problemima raspoređivanja tačaka, poznati su min-max problemi raspoređivanja tačaka unutar kvadrata ili kruga. Kod prvog problema potrebno je rasporediti n tačaka unutar jediničnog kvadrata tako da je minimalno rastojanje između njih najveće. Druga varijanta problema takođe maksimizuje njihovo minimalno rastojanje unutar jediničnog kruga [36].

Među problemima klasterizacije, značajan je problem particionisanja skupa tačaka P na podskupove $P_1, P_2, \ldots, P_k$, tako da se minimizuje maksimalni dijametar skupova P_i, $i = 1, \ldots, k$. Dijametar skupa tačaka predstavlja maksimalno rastojanje između bilo koje dve tačke. Drugim rečima, potrebno je odrediti

$$\min_{S = S_1 \cup S_2 \cup \cdots \cup S_k} \max_{1 \leq i \leq k} d(S_i),$$

gde $d(S_i) = \max\{d_{AB} : A, B \in S_i\}$ predstavlja maksimalno rastojanje među svim rastojanjima d_{AB} tačaka iz skupa S_i. Za ovaj problem je u [60] dokazano da je NP-težak.

1.1.1.4 Neki grafovski problemi sa min-max funkcijom cilja

Među širokim sprektrom grafovskih problema, biće pomenuti neki tipični, koji su min-max tipa. Neki od poznatih min-max problema nad grafovima su:

- Za dati graf $G = (V, E)$ i skup C od $|E|$ nenegativnih brojeva, kod problema težinskog dijametra potrebno je odrediti optimalno pridruživanje brojeva iz skupa C ivicama iz skupa E tako da je dijametar rezultujućeg težinskog grafa minimalan. Kako dijametar grafa ujedno predstavlja i najveće rastojanje između dva čvora, problem se može predstaviti u min-max obliku sa

$$\min_{f:C\to E} \max_{x,y\in V} d_f(x, y),$$

  pri čemu f predstavlja bijektivno preslikavanje skupa C na skup E, a $d_f(x, y)$ rastojanje čvorova x i y unutar skupa V pri datom preslikavanju f. Za ovaj problem je u [66] pokazano da je NP-težak.

- Za dati graf $G = (V, E)$ i skup T nenegativnih celih brojeva od kojih je tačno jedan broj jednak nuli, kod problema T-bojenja grafa potrebno je odrediti nenegativnu funkciju f_T definisanu nad skupom V, za koju je ispunjen uslov

$$(u, v) \in E \Rightarrow |f_T(u) - f_T(v)| \notin T.$$

  Kod T-raspona T-bojenja grafa G potrebno je odrediti

$$\min_{f_T} \max_{u,v\in G} |f_T(u) - f_T(v)|.$$

  Opisani problem je NP-težak [72].

- Neka je dat graf $G = (V, E)$ i skup razmatranih scenarija S. Svakoj ivici grafa $e \in E$ pridružuje se nenegativna vrednost c_e^s pri scenariju $s \in S$. Ako je sa T označeno odgovarajuće razapinjuće stablo, kod min-max problema razapinjućeg stabla potrebno je odrediti

$$\min_{T} \max_{s \in S} \sum_{e \in T} c_e^s.$$

Ovako definisan problem je NP-težak [161].

1.2 Robusna optimizacija

Jedan od problema prilikom konstrukcije matematičkih modela koji se odnose na realne probleme iz prakse jeste nepouzdanost ulaznih podataka. Nepouzdanost podataka može imati različite uzroke: nepreciznost ulaznih podataka koji se dobijaju kao rezultati merenja, promenljivi zahtevi korisnika, variranja u cenama transporta, promenljivi kapaciteti resursa, promenljivi broj raspoloživih resursa u datom trenutku, itd. U literaturi postoji nekoliko pristupa za uključivanje nepouzdanosti ulaznih podataka u matematičke modele u cilju što realnijeg oslikavanja situacija iz prakse. Među njima se najviše koriste stohastičko programiranje [40] i optimizacija sa probabilističkim uslovima [26]. Međutim, glavni problemi koji se javljaju prilikom primena ova dva pristupa su:

- Raspodele verovatnoća ulaznih podataka su često nepoznate u praksi, dok stohastičko programiranje i optimizacija sa probabilističkim uslovima polaze od pretpostavke da su odgovarajuće raspodele poznate [15], [83];
- Nabrajanje svih scenarija koji bi pokrili sve mogućnosti za variranje ulaznih podataka je teško postići u praksi;
- Dimenzija rezultujućih modela drastično raste sa porastom broja scenarija, što dovodi do teškoća pri rešavanju, imajući u vidu prostornu i vremensku ograničenost računarskih resursa.

Robusna optimizacija [118] je pristup za uključivanje nepouzdanosti ulaznih podataka, koji se pojavio nešto kasnije od stohastičkog programiranja i optimizacije sa probabilističkim uslovima. Ideja je da se, imajući u vidu promenljivost nekih ulaznih podataka, razmatra najgori razmatrani scenario, pri čemu funkcija cilja tada postaje min-max [14]. U [143] predložen je linearni model za konstrukciju rešenja koji uzima u obzir variranje svih ulaznih podataka unutar fiksiranog intervala. Autori u [7], [8] i [9], kao i autori u [54] i [55] razmatraju slučajeve kada ulazni podaci variraju u elipsoidnim skupovima; međutim, rezultujući robusni modeli nisu linearni. Najzad, u [13]

predlažon je odgovarajući linearni robusni model, što omogućava efikasnije pronalaženje optimalnog rešenja.

U [90] se razmatra problem robusnosti u diskretnoj optimizaciji. Funkcija cilja minimizuje najgori slučaj, imajući u vidu sva variranja ulaznih parametara u funkciji cilja i ograničenjima modela. U [14] je kreiran linearan robusni model za probleme diskretne optimizacije, pri čemu je dokazano da robusna varijanta bilo kog 0–1 problema diskretne optimizacije koji je polinomske složenosti, ostaje polinomske složenosti. Među takvim problemima su problemi određivanja drveta razapinjanja u grafu, uparivanja, najkraćeg puta, itd.

1.2.1 Robusni matematički model za problem mešovitog celobrojnog programiranja

U ovom pododeljku će biti prikazan postupak iz [14] za dobijanje robusnog modela polazeći od determinističke varijante problema mešovitog celobrojnog programiranja. Neka je definisan problem mešovitog celobrojnog programiranja

$$\min \sum_{j=1}^{n} c_j x_j$$

uz sledeća ograničenja:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad \forall i = 1, \ldots, m,$$

$$l_j \leq x_j \leq u_j \quad \forall j = 1, \ldots, n,$$

$$x_j \in \mathbb{Z} \quad \forall j = 1, \ldots, k \quad k \leq n.$$

Problem je definisan nad realnim parametrima c_j, l_j, u_j, b_i, a_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, n$, a jedina pretpostavka za promenljive x_j, $j = 1, \ldots, n$ je da je prvih k od m celobrojno. Pri tom su parametri c_j i a_{ij} podložni variranju. U robusnom modelu ovi parametri postaju promenljive za koje nije poznata raspodela, a jedina dostupna informacija je da variraju u intervalima $[c_j, c_j + \hat{c}_j]$ i $[a_{ij}, a_{ij} + \hat{a}_{ij}]$, $\hat{c}_j, \hat{a}_{ij} \geq 0$, $i = 1, \ldots, n$, $j = 1, \ldots, m$.

Ova pretpostavka odgovara praktičnim situacijama, budući da u praksi često nisu poznate raspodele po kojima variraju ulazni podaci. Takođe, u praksi nisu sve vrednosti c_j i a_{ij} podložne variranju, tj. u većini slučajeva ne važi uvek $\hat{c}_i > 0$ i $\hat{a}_{ij} > 0$ za sve promenljive iz skupa.

Neka je $J_0 = \{j : \hat{c}_j > 0,\ 1 \leq j \leq n\}$ skup svih indeksa j za koje je parametar c_j podložan variranju i neka je celobrojni parametar Γ_0 vrednost iz intervala $[0, |J_0|]$. Slično, definisani su skupovi $J_i = \{j : \hat{a}_{ij} > 0,\ 1 \leq j \leq n\}$, $i = 1, \ldots, m$, koji predstavljaju skupove onih indeksa j za koje je (za fiksiran indeks i) parametar a_{ij} podložan variranju. Sa Γ_i je označen celobrojni parametar iz intervala $[0, |J_i|]$, $i = 1, \ldots, m$. Parametar Γ_0 označava nivo robusnosti u funkciji cilja. Ako je $\Gamma_0 = 0$, tada nijedna od vrednosti c_j ne varira, dok za $\Gamma_0 = |J_0|$ je $\hat{c}_j > 0$ za sve j, odnosno svi odgovarajući parametri podležu robusnosti. Slično, Γ_i definiše nivo robusnosti za ulazne parametre koji se odnose na ograničenje $\sum_{j=1}^{n} a_{ij}x_j = b_i$. Ako je $\Gamma_i = 0$, nijedna vrednost a_{ij}, $i = 1, \ldots, m$ ne varira, dok za $\Gamma_i = n$ sve vrednosti podležu variranju. U nastavku će biti navedena teorema kojom je pokazano na koji način se za polaznu robusnu varijantu determinističkog linearnog modela kreira odgovarajući robusni linarni model.

Teorema 1.1 Neka su $\mathbf{c} = [c_j]_{j\in J}$, $\hat{\mathbf{c}} = [\hat{c}_j]_{j\in J}$, $\mathbf{l} = [l_j]_{j\in J}$ i $\mathbf{u} = [u_j]_{j\in J}$, $J = \{1, \ldots, n\}$ n-dimenzioni vektori, $\mathbf{b} = [b_i]_{i\in I}$ m-dimenzioni vektor, $I = \{1, \ldots, m\}$, a $A = [a_{ij}]_{i\in I, j\in J}$ i $\hat{A} = [\hat{a}_{ij}]_{i\in I, j\in J}$ matrice dimenzija $m \times n$. Neka su pritom definisani skupovi $J_0 = \{j \in J : \hat{c}_j > 0\}$ i $J_i = \{j \in J : \hat{a}_{ij} > 0\}$, $i \in I$ i celobrojni parametri $\Gamma_i \in [0, |J_i|]$, $i \in \{0\} \cup I$. Tada se problem definisan sa

$$\min\left(\sum_{j\in J} c_j|x_j| + \max_{S_0\subseteq J_0:|S_0|\leq\Gamma_0} \sum_{j\in S_0} \hat{c}_j|x_j|\right), \tag{1.3}$$

$$\sum_{j\in J} a_{ij}|x_j| + \max_{S_i\subseteq J_i:|S_i|\leq\Gamma_i} \sum_{j\in S_i} \hat{a}_{ij}|x_j| \quad \forall i \in I, \tag{1.4}$$

$$l_j \leq x_j \leq u_j \quad \forall j \in J, \tag{1.5}$$

$$x_j \in \mathbb{Z} \quad \forall j \in \{1, \ldots, k\} \subseteq J \tag{1.6}$$

može formulisati kao problem mešovitog celobrojnog programiranja na sledeći način:

$$\min\left(\sum_{j\in J} c_j x_j + z_0\Gamma_0 + \sum_{j\in J_0} p_{0j}\right), \tag{1.7}$$

$$\sum_{j\in J} a_{ij}x_j + z_i\Gamma_i + \sum_{j\in J_i} p_{ij} \leq b_i \quad \forall i \in I, \tag{1.8}$$

$$z_0 + p_{0j} \geq \hat{c}_j y_j \quad \forall j \in J_0, \tag{1.9}$$

$$z_i + p_{ij} \geq \hat{a}_{ij} y_j \quad \forall i \in I \quad \forall j \in J_0, \tag{1.10}$$

$$p_{ij} \geq 0 \quad \forall i \in \{0\} \cup I \quad \forall j \in J_i, \tag{1.11}$$

$$y_j \geq 0 \quad \forall j \in J, \tag{1.12}$$

$$z_i \geq 0 \quad \forall i \in \{0\} \cup I, \tag{1.13}$$

$$-y_j \leq x_j \leq y_j \quad \forall j \in J, \tag{1.14}$$

$$l_j \leq x_j \leq u_j \quad \forall j \in J, \tag{1.15}$$

$$x_j \in \mathbb{Z} \quad \forall j \in \{1, \ldots, k\} \subseteq J. \tag{1.16}$$

Dokaz. Ovde će biti dokazano opštije tvrđenje (teorema 1 iz [14]), gde se pretpostavlja da parametri Γ_i, $1 \leq i \leq n$ mogu uzeti realne vrednosti, dok parametar Γ_0 ostaje celobrojan, odakle jednostavno sledi navedeno tvrđenje. U modelu (1.3) – (1.6) se tada uslov (1.4) zamenjuje uslovom

$$\sum_{j \in J} a_{ij}|x_j| + \max_{|S_i \cup t_i| : S_i \subseteq J_i, |S_i| \leq \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i} \left(\sum_{j \in S_i} \hat{a}_{ij}|x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}|x_{t_i}| \right) \quad \forall i \in I. \tag{1.17}$$

Neka je $\mathbf{x}^* = [x_1^*, \ldots, x_m^*]$ i

$$\beta_i(\mathbf{x}^*) = \max_{|S_i \cup t_i| : S_i \subseteq J_i, |S_i| \leq \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i} \left(\sum_{j \in S_i} \hat{a}_{ij}|x_j^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}|x_{t_i}^*| \right). \tag{1.18}$$

Izraz (1.18) se može napisati u sledećem obliku:

$$\max \sum_{j \in J_i} \hat{a}_{ij} |x_j^*| z_{ij}, \tag{1.19}$$

$$\sum_{j \in J_i} z_{ij} \leq \Gamma_i, \tag{1.20}$$

$$0 \leq z_{ij} \leq 1 \quad \forall j \in J_i. \tag{1.21}$$

U optimalnom rešenju problema (1.19) – (1.21) $\lfloor \Gamma_i \rfloor$ promenljivih ima vrednost 1, dok jedna promenljiva ima vrednost $\Gamma_i - \lfloor \Gamma_i \rfloor$. Ovo odgovara izboru podskupa

$$\{|S_i \cup t_i| : S_i \subseteq J_i, |S_i| \leq \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}$$

uz odgovarajuću funkciju cilja

$$\sum_{j \in S_i} \hat{a}_{ij} |x_j^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{it_i} |x_{t_i}^*|.$$

Dualni problem za problem (1.19) – (1.21) je sledećeg oblika:

$$\min \sum_{j \in J_i} p_{ij} + \Gamma_i z_i \tag{1.22}$$

$$z_i + p_{ij} \geq \hat{a}_{ij} |x_j^*| \quad \forall j \in J_i, \tag{1.23}$$

$$p_{ij} \geq 0 \quad \forall j \in J_i, \tag{1.24}$$

$$z_i \geq 0 \quad \forall i \in I. \tag{1.25}$$

Na osnovu teoreme o jakoj dulanosti [157] sledi da se vrednosti funkcija cilja primarnog problema (1.19) – (1.21) i dualnog problema (1.22) – (1.25) poklapaju i iznose $\beta_i(\mathbf{x}^*)$. Dalje, funkcija cilja (1.3) se može napisati u obliku

$$\begin{aligned}
\beta_0(\mathbf{x}^*) &= \max \left\{ \sum_{j \in S_0} \hat{c}_j |x_j^*| : |S_0| \leq \Gamma_0, S_0 \subseteq J_0 \right\} \\
&= \max \left\{ \sum_{j \in J_0} \hat{c}_j |x_j^*| z_{0j} : \sum_{j \in J_0} z_{0j} \leq \Gamma_0, 0 \leq z_{0j} \leq 1, \forall j \in J_0 \right\} \\
&= \min \left\{ \sum_{j \in J_0} p_{0j} + \Gamma_0 z_0 : z_0 + p_{0j} \geq \hat{c}_j |x_j^*|, z_0 \geq 0, p_{0j} \geq 0, \forall j \in J_0 \right\}.
\end{aligned}$$

Zamenom poslednje jednakosti u (1.3), (1.5), (1.6), (1.17), sledi da je problem (1.3), (1.5) – (1.17) ekvivalentan problemu (1.7) – (1.16). □

Robusna varijanta polaznog modela ima, prema teoremi 1.1, svoj linearni ekvivalent. Treba napomenuti da se ovde posmatra slučaj kada su sve vrednosti Γ_i celobrojne, dok se u [14] razmatra slučaj gde parametri Γ_i, $i = 1, \ldots, m$ mogu uzeti realne vrednosti.

1.3 Pojam memetskog algoritma

Memetski algoritam (MA) je heuristika bazirana na populaciji jedinki koja se sastoji od evolutivnog dela i lokalnog pretraživanja, koji je inkorporiran unutar jedne generacije evolutivnog algoritma [78]. On se bazira na filozofskoj teoriji Ričarda Dokinsa, prema kojoj se svo ljudsko saznanje može raščlaniti na jednostavnije delove – meme, koji se kasnije mogu duplirati, modifikovati ili kombinovati sa drugim memima, kako bi se kreirali novi [42]. Ovakav pristup je inspirisao naučnike kasnih osamdesetih godina da definišu pojam memetskog algoritma [116]. Prvi memetski algoritam je predstavljao modifikaciju generskog algoritma u koji je bila inkorporirana lokalna pretraga i bio je primenjen na problem trgovačkog putnika [117].

Kasnije je ovaj koncept MA proširen, pri čemu evolutivni deo algoritma može predstavljati heuristiku zasnovanu na populaciji jedinki (u oznaci P-heuristika – population based heuristic), a lokalno pretraživanje heuristiku zasnovanu na unapređivanju jednog rešenja, nezavisno od vrednosti drugih rešenja (u oznaci S-heuristika – single solution based heuristic). Koncept P-heuristike se sastoji od uzastopne primene operatora selekcije jedinki iz populacije i primene jednog ili više operatora nad njima, a S-heuristike na unapređivanju svake jedinke [120]. Iako su u vreme njihove najranije definicije memetski algoritmi od strane naučne zajednice posmatrani skeptično, krajem 20. veka do danas uočljiva je njihova nagla ekspanzija. Skoriji pregledni radovi o pojmu i primenama memetskog algoritma se mogu pronaći u [120] i [121].

1.3.1 Struktura memetskog algoritma

Memetski algoritam (MA) je baziran na populaciji jedinki, a svaka jedinka odgovara jednom rešenju u pretraživačkom prostoru. Cilj je da se kroz generacije poboljšavanjem jedinki napreduje ka optimalnom rešenju. Osnovna struktura MA je prikazana algoritmom 1.1. Najpre se učitaju ulazni podaci, nakon čega se vrši inicijalizacija populacije. Algoritam se izvršava sve dok

nije ispunjen kriterijum zaustavljanja, generišući u svakoj iteraciji novu generaciju jedinki. U toku jedne generacije, vrši se primena operatora selekcije i rekombinacije, nakon čega se na odabrani skup jedinki primenjuje lokalno pretraživanje. U opštem slučaju se, umesto proste lokalne pretrage, koristi kao lokalno pretraživanje i tabu pretraga, simulirano kaljenje, metoda promenljivih okolina, i sl. U nekim slučajevima se može izvršiti i hibridizacija nekih od tih algoritama. Nakon primene odgovarajućih operatora i lokalnog pretraživanja, vrši se ažuriranje populacije, kojim se odlučuje koje jedinke će učestvovati u populaciji sledeće generacije algoritma. Na kraju se proverava da li je zadovoljen neki od zadatih kriterijuma zaustavljanja. Nakon završene primene memetskog algoritma, ispisuje se dobijeno rešenje.

Algoritmom 1.1 je predstavljen najraniji koncept na kome je bila zasnovana osnovna struktura MA [120]. Kasnije je taj koncept proširen, pri čemu se svaka iteracija MA sastoji iz tri faze: P-heuristike, S-heuristike i ažuriranja populacije. Koncept P-heuristike se sastoji od uzastopne primene operatora selekcije i skupa operatora nad izabranim jedinkama. Pritom je moguća uzastopna primena jednog ili više operatora P-heuristike. Za operatore unutar P-heuristike najčešće se koriste oni koji su zastupljeni u heuristikama zasnovanim nad populacijom jedinki, kao što su operator ukrštanja, mutacije, invertovanja, operatori koji se koriste pri optimizaciji rojeva, itd. Primena skupa operatora P-heuristike naziva se rekombinacijom ili operatorom rekombinacije. Nakon prve faze, primenjuje se S-heuristika, čiji je cilj poboljšanje populacije jedinki ili jednog njenog dela. Lokalno pretraživanje se najčešće koristi kao S-heuristika unutar MA, ali se mogu koristiti i druge S-heuristike koje se zasnivaju na konceptu lokalnog pretraživanja. Lokalno pretraživanje se može shvatiti i kao unarni operator, čime on može predstavljati i deo P-heuristike, ali se, zbog njegovog značaja, najčešće izdvaja u posebnu fazu. U poslednjoj fazi se vrši ažuriranje populacije, gde se najčešće čuvaju dobijena kvalitetna rešenja, a ona manje kvalitetna zamene novim. Nova rešenja se mogu iznova generisati na slučajan ili pseudoslučajan način, ili se mogu dobiti primenom nekih od operatora iz P-heuristike [120].

1.3.1.1 Inicijalizacija

U fazi inicijalizacije je potrebno generisati početnu populaciju algoritma. Veličina populacije je u većini slučajeva od nekoliko desetina do nekoliko stotina jedinki. Inicijalizacija može biti izvršena na slučajan ili pseudoslučajan način. Na primer, ukoliko je jedinka kodirana binarnim kodom dužine n, može se postaviti verovatnoća uspostavljanja jediničnog bita na 0.5. Ukoliko je problem takav da je očekivanje da će manji broj bitova u kodu optimalnog rešenja imati vrednost 1, ta verovatnoća se može smanjiti ili po potrebi

Algoritam 1.1 Osnovna struktura memetskog algoritma

```
1: Učitavanje podataka
2: Inicijalizacija
3: while nije ispunjen kriterijum zaustavljanja do
4:    Selekcija
5:    Rekombinacija
6:    S-heuristika zasnovana na lokalnom pretraživanju
7:    Ažuriranje populacije
8: end while
9: Ispis rešenja
```

povećati, ukoliko se očekuje da će u binarnom kodu optimalnog rešenja veći broj bitova imati vrednost 1. Ako je dodatno nametnut uslov da je potrebno uspostaviti tačno p, $p < n$ jediničnih bitova, najpre se prvi bit može odabrati uniformno, a nakon njega drugi, pri čemu je iz potencijalnog izbora izbačen prvi izabrani bit. Ovaj postupak se ponavlja p puta, gde se u poslednjem koraku između preostalih $n-p+1$ bitova čija je vrednost jednaka nuli bira poslednji koji će biti uspostavljen. Sa druge strane, neretko se može desiti da se inicijalizacija vrši primenom neke jednostavnije heuristike, koja se relativno brzo izvršava, budući da je u nekim slučajevima potrebno proslediti kvalitetnija rešenja memetskom algoritmu, kako bi njegova primena bila uspešnija. Tako se, na primer, može koristiti pohlepni algoritam. Još jedan pristup je da se inicijalizacija izvršava u dva koraka. U prvom koraku se rešenja odabiraju proizvoljno, dok se u drugom koraku vrši primena algoritma lokalnog pretraživanja. Nakon završene inicijalizacije, na dobijene jedinke se primenjuju operatori selekcije, rekombinovanja i lokalnog pretraživanja kroz više generacija, sve dok nije ispunjen kriterijum zaustavljanja.

1.3.1.2 Selekcija

Pod selekcijom se podrazumeva odabir kandidata među populacijom, pomoću kojih se mogu dobiti nova rešenja ili koja mogu opstati u narednim generacijama. Kako bi se dobila što kvalitetnija rešenja u narednim generacijama, selekcija obično uzima u obzir kvalitet rešenja imajući u vidu vrednost funkcije cilja, a drugi kriterijum za selekciju je raznovrsnost. Ukoliko se pri selekciji ne biraju raznovrsna rešenja, može se dogoditi da i rešenje nastalo primenom određenog operatora nad tim rešenjima bude istog ili lošijeg kvaliteta. Ukoliko su rešenja u populaciji dovoljno raznovrsna, selekcija može biti i u potpunosti proizvoljna, pri čemu se uniformno biraju jedinke na koje će se primenjivati određeni operator. Među najčešćim tipovima selekcije koji

se sreću u literaturi su:

- Kod proste ruletske selekcije se svakom rešenju iz populacije dodeljuje verovatnoća odabira na osnovu vrednosti njegove funkcije cilja [68]. Primena ove metode obično dovodi do brzog gubitka genetskog materijala i preuranjene konvergencije.

- Kod selekcije zasnovane na rangiranju kodova jedinki prema njihovoj prilagođenosti, jedinke se biraju na osnovu njihove pozicije u populaciji [68]. Na taj način i lošije jedinke mogu da budu izabrane, čak i u slučaju velikih razlika među vrednostima funkcije cilja. Osnovni nedostatak kod ovog tipa selekcije je što populacija mora biti sortirana pri svakoj primeni selekcije.

- Turnirska selekcija je oblik selekcije pomoću koje se jedinke biraju putem izvođenja turnira [108]. Učesnici turnira se biraju proizvoljno, a pobednik turnira je jedinka na koju će biti primenjeni operatori rekombinacije i lokalnog pretraživanja. Parametar veličine turnira može biti fiksiran ili se može menjati. Ovakav način selekcije je efikasan, jer ne zahteva sortiranje. Takođe, može se i jednostavno paralelizovati, budući da se turniri mogu simultano održavati. Promenom veličine turnira se jednostavno može povećati ili ublažiti efekat primene selekcije. Uopštenje ovog tipa operatora selekcije predstavlja fino gradirana turnirska selekcija [61], kod koje se umesto celobrojnog parametra veličine turnira, uvodi realan parametar koji predstavlja željenu prosečnu veličinu turnira. Fino gradirana turnirska selekcija zadržava sve prednosti osnovne turnirske selekcije, a dodatno omogućava da veličina turnira bude promenljiva tokom izvršavanja algoritma.

- Kod operatora uniformne selekcije se omogućava očuvanje genetske raznovrsnosti. Pritom, iako se time nužno ne povećava prosečna prilagođenost populacije, ova metoda je pokazala dobre rezultate u praksi [95].

1.3.1.3 Rekombinacija

Proces rekombinacije se vezuje za populacijsku heuristiku memetskog algoritma. U ovoj fazi se vrši primena skupa operatora P-heuristike nad jedinkama koje su odabrane u procesu selekcije. Algoritmom 1.2 je prikazan odgovarajući pseudokod operatora rekombinacije. Neka je skup operatora označen sa O, a broj jedinki na koje se primenjuje i-ti operator O_i sa $N(O_i)$. Operatori se uzastopno primenjuju unapred zadati broj puta. Operator rekombinacije se primenjuje na celoj populaciji ili njenom delu, a jedinke koje

su argumenti operatora se biraju procesom selekcije. U opštem slučaju se svakom od operatora može dodeliti verovatnoća njegove primene. Nakon završene primene odgovarajućeg operatora, vrši se ažuriranje populacije.

Algoritam 1.2 Prikaz rekombinacije populacije memetskog algoritma

```
1: for O_i ∈ O do
2:     for j ∈ {1, 2, ..., N(O_i)} do
3:         Izvršiti selekciju j-tog rešenja iz postojeće populacije na koje se primenjuje operator O_i
4:     end for
5:     Nad odabranim rešenjima primeniti operator O_i
6:     Izvršiti ažuriranje populacije
7: end for
```

Na primer, ukoliko genetski algoritam predstavlja evolutivni deo memetskog algoritma, u procesu rekombinacije će najčešće biti primenjivana dva operatora – operator ukrštanja i operator mutacije. Operator ukrštanja je binarni, tj. u procesu selekcije se iz populacije biraju dva rešenja, čiji se geni kasnije kombinuju. Operator mutacije je unaran i time se odabira jedno rešenje čiji određeni bitovi eventualno mutiraju. Još neki operatori koji se mogu primenjivati u ovoj fazi su operator inverzije, operator razmrdavanja, operatori koji se koriste u algoritmima zasnovanim na ponašanju čestica u roju, itd.

1.3.1.4 Primena lokalnog pretraživanja

Dok populacijska heuristika memetskog algoritma služi pre svega da se pažljivo odaberu potencijalno kvalitetniji delovi pretraživačkog prostora, tokom faze S-heuristike se odgovarajući deo prostora pretražuje u cilju pronalaženja lokalnog minimuma (odnosno lokalnog maksimuma, ukoliko je u pitanju problem maksimizacije). U tu svrhu se najčešće koristi heuristika zasnovana na lokalnom pretraživanju. Pritom treba voditi računa koliko puta se lokalna pretraga primenjuje. Ukoliko se primenjuje često, može se dogoditi da se iznova dostiže isti lokalni optimum, čime se i smanjuje raznovrsnost rešenja. Ukoliko se primenjuje retko ili na malom broju jedinki, to može dovesti do toga da se u određenim kvalitetnim delovima pretraživačkog prostora ne dostigne lokalni optimum, a time ni dovoljno kvalitetno krajnje rešenje algoritma. U [77] su izdvojena četiri važna parametra za primenu lokalne pretrage inkorporirane u memetski algoritam:

- Frekvencija – u osnovnoj verziji memetskog algoritma, lokalna pretraga se primenjuje sa fiksnom frekvencijom i na sve jedinke. U drugim ver-

zijama, može se odabrati podskup jedinki prema nekom kriterijumu. Tako se često uzima najkvalitetnija jedinka, određen broj najkvalitetnijih jedinki ili se jedinke odabiraju na taj način da budu raznovrsne, tj. da odgovarajuća rešenja ne budu previše slična. Zatim se lokalna pretraga primenjuje samo nad odabranim podskupom jedinki.

- Verovatnoća primene – u nekim slučajevima se lokalna pretraga može primenjivati sa određenom verovatnoćom p, $0 < p < 1$. Ovaj parametar može biti fiksiran, može varirati ili zavisiti od prirode problema koji se rešava.

- Dubina – vrednost ovog parametra je u uskoj vezi sa vremenom izvršavanja lokalne pretrage. Lokalna pretraga se može primenjivati dok se ne pretraže sva rešenja iz okoline trenutnog ili samo na određen broj proizvoljno odabranih okolnih rešenja.

- Efikasnost – ovaj parametar je u uskoj vezi sa dubinom lokalne pretrage. Lokalna pretraga je efikasnija ukoliko se ne pretražuje celokupna okolina trenutnog rešenja. Sa druge strane, tako se može smanjiti kvalitet rešenja, jer se često može desiti da nije dostignut odgovarajući lokalni optimum. Preciznije, ukoliko se pretražuje celokupna okolina odgovarajućeg rešenja, u svakom koraku lokalne pretrage će se naći rešenje koje predstavlja trenutni optimum, pri čemu se onda lokalni optimum uvek dostiže. Ukoliko se ne razmatraju sva rešenja iz okoline, može se desiti da se preskoči ono koje je minimalno, čime se nekad ne dostiže lokalni optimum. Sa druge strane, ovo ubrzava vreme pretraživanja.

Pri prelasku u naredno rešenje kod lokalnog pretraživanja, ponovno računanje funkcije cilja može biti veoma skupo u pogledu efikasnosti. Zbog toga se, ukoliko je to moguće, često pribegava nalaženju načina da se iskoriste izračunate vrednosti za prethodno rešenje i one ažuriraju, što u velikom broju slučajeva može znatno ubrzati algoritam [110, 149, 150].

1.3.1.5 Ažuriranje populacije

Pri ažuriranju populacije, potrebno je odlučiti koje će jedinke (posmatrajući i stare i novodobijene jedinke) ući u sastav nove populacije. Odluka koju je potrebno doneti se najčešće zasniva na kvalitetu rešenja, ali i raznovrsnosti u celoj populaciji. Ako bi se odluka donosila jedino na osnovu kvaliteta rešenja, tada bi se u narednoj generaciji našli oni koji među svim starim i novodobijenim rešenjima imaju najbolju vrednost funkcije cilja. Međutim, u tom slučaju bi se vrlo brzo izgubila raznovrsnost rešenja i algoritam bi radio

nad istim delovima pretraživačkog prostora, sa veoma malom verovatnoćom da detektuje druge, potencijalno kvalitetnije delove prostora pretrage, pri čemu bi došlo do konvergencije ka lokalnom optimumu. Stoga se politika odabira naredne populacije zasniva na činjenici da se odaberu dovoljno kvalitetna rešenja, ali da se sačuva i raznovrsnost unutar populacije. U pitanju je strategija kojom se ograničava broj duplikata ili jedinki sa istom vrednošću funkcije cilja, a različitim kodovima [110, 150]. Takođe, često se pribegava i elitističkoj strategiji [120], koja podrazumeva da se deo rešenja direktno prebaci u narednu populaciju, a da se operatori primenjuju samo na ostatak populacije. Kako bi se sačuvala kvalitetna rešenja, biraju se obično rešenja sa najboljom vrednošću funkcije cilja, a na ostale se primenjuju operatori. Ovo dodatno ubrzava ceo proces, budući da je samo za deo populacije potrebno vršiti ponovno računanje funkcije cilja.

1.3.1.6 Kriterijum zaustavljanja

Pod kriterijum zaustavljanja se podrazumeva uslov ili skup uslova pod kojim se prekida izvršavanje memetskog algoritma. Za kriterijum zaustavljanja se najčešće uzima jedan od sledećih uslova ili neka njihova kombinacija:

- Dostignut je maksimalan broj iteracija algoritma. Jedna iteracija u ovom slučaju odgovara formiranju jedne populacije.
- Najbolja jedinka se ponovila maksimalan broj puta. Pod najboljom jedinkom se podrazumeva jedinka koja odgovara dopustivom rešenju i koja ima najmanju vrednost funkcije cilja, ukoliko je u pitanju problem minimizacije, odnosno ona koja ima najveću vrednost funkcije cilja, ako se radi o maksimizaciji.
- Dostignuto je rešenje zadovoljavajućeg kvaliteta.
- Dostignuto je maksimalno vreme izvršavanja.
- Dostignuta je maksimalna sličnost jedniki u populaciji.
- Izvršen je prekid od strane korisnika.

1.3.2 Primene memetskog algoritma na probleme diskretne optimizacije

Memetski algoritam je prvi put u literaturi primenjen na problem diskretne optimizacije [116], gde je razmatran problem trgovačkog putnika (Traveling salesman problem – TSP) i predložena hibridizacija genetskog algo-

ritma sa heuristikom lokalnog pretraživanja. U [16] je predložena hibridizacija evolutivnog algoritma i lokalne pretrage, a u [69] je unutar evolutivnog algoritma inkorporiran operator zasnovan na principu ukrštanja jedinki. Oba algoritma su primenjena za rešavanje problema trgovačkog putnika. Sličan pristup je dat u [82]. Tehnika tzv. genetskog ukrštanja ivica, koja se takođe može smatrati ranom verzijom memetskog algoritma, primenjena je na TSP u [101]. U [64] i [65] razmatra se uticaj lokalnog pretraživanja unutar evolutivnog algoritma pri primeni na TSP. U [106] MA je primenjen na rešavanje problema trgovačkog putnika velikih dimenzija.

Kasnije je koncept MA primenjen i na druge optimizacione probleme. U [105] su predstavljeni različiti evolutivni operatori koji su inkorporirani u MA za rešavanje kvadratnog problema pridruživanja. Za problem biparticionisanja grafa je u [104] predložena hibridizacija genetskog algoritma i lokalne pretrage, kojom se efikasno dostižu rešenja koja su blizu optimalnih. U [62] je razvijen memetski algoritam za jednu varijantu problema nalaženja minimalnog drveta razapinjanja u grafu, gde se poseban značaj daje operatoru rekombinacije. Genetski algoritam sa inkorporiranom efikasnom lokalnom pretragom za problem particionisanja grafa je predstavljen u [19]. Pritom je uključena i faza inicijalizacije koja ubrzava algoritam u celini. Za problem nalaženja maksimalnog nezavisnog skupa u [2] se u okviru evolutivnog algoritma operator ukrštanja primenjuje tako da jedan potomak zadržava kvalitet rešenja, a drugi čuva njegovu raznovrsnost. U [35] je za problem bojenja grafa razvijen memetski algoritam gde je unutar evolutivne sheme inkorporiran algoritam opadajućeg gradijenta. Za rešavanje višedimenzionalnog problema ranca je u [6] izvršena hibridizacija genetskog algoritma i lokalne pretrage, koja koristi karakteristike samog problema kako bi poboljšala kvalitet algoritma.

Na hab lokacijski problem neograničenih kapaciteta je u [1] primenjen genetski algoritam koji je hibridizovan sa tabu pretragom. Za lokacijski problem rutiranja ograničenih kapaciteta je u [50] razvijen memetski algoritam, gde je lokalna pretraga inkorporirana unutar operatora ukrštanja. Za rešavanje problema p-hab medijane neograničenih kapaciteta u [160] je predložen hibridni genetski algoritam. Memetski algoritam je primenjen i na dvostepeni lokacijski problem neograničenih kapaciteta [110], zatim lokacijski problem neograničenih kapaciteta sa više nivoa [98], kao i hibridni genetski algoritmi za probleme efikasnog pretraživanja na društvenim mrežama [149], [150]. Kada je reč o problemima robusne diskretne optimizacije, memetski algoritam je primenjen na robusnu varijantu dvostepenog lokacijskog problema ograničenih kapaciteta [109], kao i na problem optimalnog raspoređivanja jedinica za reagovanje u hitnim slučajevima [144].

2 Višeperiodni lokacijski problem za raspoređivanje jedinica za reagovanje u hitnim slučajevima

Primene operacionih istraživanja u oblasti sistema za reagovanje u hitnim slučajevima datiraju još od pedesetih i šezdesetih godina XX veka ([79], [135], [156]). U literaturi se mogu pronaći brojni radovi koji se odnose na primenu lokacijskih modela pri određivanju pozicija za izgradnju stanica hitne pomoći, policijskih i vatrogasnih brigada, kao i optimizaciju njihovog kretanja. Prema [151], najčešća podela lokacijskih modela koji nalaze primenu u ovoj oblasti su modeli tipa medijane i centra, kao i modeli pokrivanja.

Problemi tipa medijane vrše alokaciju resursa sistema za hitne slučajeve, a zatim odgovarajućim resursima pridružuju korisnike tako da prosečno ili ukupno vreme u mreži bude minimalno. Jedan od najčešće korišćenih problema ovog tipa je problem p-medijane kod koga je potrebno alocitari tačno p resursa, kako bi ukupna suma rastojanja korisnika i njima dodeljenih resursa bila minimalna. Formulacija problema p-medijane data je najpre u [71] i [132], da bi nešto složeniji modeli bili razvijeni kasnije (videti npr. [38], [158]). U [158] se vrši alokacija resursa u transportnoj mreži, pri čemu najbliži resurs (u smislu cene, ukupnog vremena ili rastojanja) nije u mogućnosti da u potpunosti zadovolji potražnju korisnika. U [127] predložena je varijanta problema p-medijane kod kojeg je uključena kontrola opterećenja svakog resursa i omogućeno je da se, pri većoj opterećenosti, nekim ili svim korisnicima koje dati resurs opslužuje, dodeli rezervna usluga koja bi zadovoljila njihovu potražnju. U [81] se predlaže uopšteni lokacijski model za reagovanje u hitnim slučajevima. Model se može sagledati kao varijanta modela pokrivanja, problema p-medijane ili problema p-centra, u zavisnosti od konkretne primene u praksi.

Modeli pokrivanja vrše alokaciju resursa tako da bude zadovoljena potražnja korisnika imajući u vidu pozicije resursa i korisnika, kao i maksimalno dozvoljeno rastojanje. Pritom je dovoljno da postoji bar jedan resurs koji je

na manjem rastojanju od maksimalnog, kako bi se korisnik smatrao pokrivenim, odnosno kako bi bilo omogućeno njegovo snabdevanje. Navedeni problem predstavlja osnovnu varijantu problema maksimalnog pokrivanja koja je uvedena u [27].

Verzija ovog modela primenjena je na lociranje policijskih stanica [39] i stanica hitne pomoći [131]. U radu [39] je razmatrana standardna formulacija problema maksimalnog pokrivanja, kao i nova formulacija koja obezbeđuje alternativnu preraspodelu policijskih stanica u Dalasu. U [131] se razmatra model kod koga je potrebno odrediti minimalan broj ambulatnih ustanova koje je potrebno uspostaviti, kao i lokacije za njihovo uspostavljanje. Poseban akcenat je stavljen na vremenske periode kada potražnja korisnika varira u velikoj meri [131]. U radu [44] se razmatra model za reagovanje u hitnim slučajevima, baziran na lokacijskom problemu raspoređivanja resursa i problemu rutiranja vozila za transport robe od resursa do korisnika. U [4] razmatra se problem uspostavljanja resursa i raspoređivanja zaliha robe po resursima, kako bi zahtevi korisnika bili zadovoljeni u što većoj meri. Detaljniji pregled odgovarajućih modela se može pronaći u [18].

Najveći broj lokacijskih problema za reagovanje u hitnim slučajevima se bazira na problemima zadovoljavanja potražnje korisnika pod određenim uslovima. Sa druge strane, u [133] razmatra se lokacijski model koji uzima u obzir odgovarajuće karakteristike resursa. Model predstavlja probabilističku varijantu lokacijskog problema maksimalnog pokrivanja. Među modelima koji razmatraju karakteristike resursa, često se javlja potreba za minimizacijom njihove maksimalne opterećenosti. Tako se u [89] razmatra problem optimalne raspodele zdravstvenih centara u Južnoj Koreji, pri čemu se minimizuje maksimalan broj pacijenata u nekom centru. Slični lokacijski problemi, gde je potrebno minimizovati maksimalan broj korisnika koji su dodeljeni nekom resursu se razmatraju u [148] i [99].

U [144] razmatran je min-max lokacijski problem koji se odnosi na raspoređivanje specijalnih policijskih jedinica na teritoriji Republike Srbije, tako da se obezbedi što efikasnije reagovanje pri pojavi teških krivičnih dela. Potrebno je odabrati lokacije na koje će biti raspoređene policijske jedinice, koje bi bile u mogućnosti da po potrebi intervenišu u gradovima koji su im dodeljeni. Svaki grad se dodeljuje tačno jednoj, i to najbližoj, uspostavljenoj jedinici, a broj lokacija na kojima se one uspostavljaju je unapred fiksiran. Potrebno je odrediti na kojim mestima će policijske jedinice biti locirane, kako bi njihova maksimalna opterećenost po svim resursima koje opslužuju bila minimalna. U [144] je prikazana deterministička i robusna formulacija problema, pri čemu prosečan broj krivičnih dela u gradovima varira. Za rešavanje determinističkog i robusnog modela je predložen algoritam zasnovan na hibridizaciji evolutivne metode i lokalnog pretraživanja. Pritom su

dostignuta sva poznata optimalna rešenja dobijena pomoću CPLEX rešavača. Prikazani su rezultati i za instance većih dimenzija, gde CPLEX nije bio u mogućnosti da dostigne optimalno rešenje za odgovarajući vremenski period. Instance nad kojima je algoritam testiran su dobijene na osnovu geografskog položaja 165 gradova i 234 potencijalne lokacije za specijalne policijske jedinice na teritoriji Republike Srbije. Najveća instanca sadrži sva naseljena mesta i potencijalne lokacije, dok su instance manjih dimenzija dobijene grupisanjem nasaljenih mesta, koja su po geografskom položaju međusobno blizu [144].

Imajući u vidu da u praksi policijske jedinice rade po smenama (obično dve smene od po 12 h ili tri smene od po 8 h), uočena je potreba za uopštavanjem modela iz [144]. Kod uopštenog modela je uveden skup različitih perioda u kojim jedinice mogu reagovati. Pritom u različitim periodima ne mora biti izražena ista potreba za intervencijom određene policijske jedinice. Prednost višeperiodnog modela u odnosu na model sa jednim periodom je u tome što preciznije razmatra promene u potražnji korisnika u različitim vremenskim intervalima. U nastavku je najpre prikazana odgovarajuća matematička formulacija problema, kao i njena robusna varijanta. Kod robusne varijante se razmatra slučaj kada broj teških krivičnig dela varira u određenom intervalu, što odgovara situaciji u praksi. Predložen je i memetski algoritam za njegovo rešavanje i prikazani su eksperimentalni rezultati izvršavanja. U slučaju jednog perioda, rezultati su upoređeni sa rezultatima hibridnog genetskog algoritma predloženog u radu [144].

2.1 Matematička formulacija

Neka je sa I označen skup gradova, sa J skup potencijalnih lokacija na kojima se mogu rasporediti specijalne jedinice policije i sa T skup posmatranih perioda. Kao u radu [144], važi da je $I \subseteq J$, što znači da specijalne policijske jedinice mogu mogu biti locirane u gradovima, ali i van njih. Svakom periodu $t \in T$ i gradu $i \in I$ dodeljen je nenegativni parametar f_{ti}, koji označava prosečan broj krivičnih dela, dobijen na osnovu statističkih podataka u prethodnom periodu. Za svako $i \in I$ i $j \in J$ poznato je rastojanje d_{ij} između odgovarajućeg grada i potencijalne lokacije. U opštem slučaju, ne mora biti ispunjena jednakost $d_{ij} = d_{ji}$, $i, j \in I$, budući da policijska jedinica može koristiti jedan put od lokacije i do lokacije j, a drugi put od lokacije j do lokacije i. Neka je sa c označen poluprečnik kruga unutar koga policijska jedinica uspostavljena u njegovom centru može pravoremeno da reaguje na incidente. Dalje, neka su uvedeni sledeći skupovi:

- $C_{ij} = \{k \in J : d_{ik} \leq d_{ij}\}$, $i \in I$, $j \in J$ – skup potencijalnih lokacija

$k \in J$ za koje je rastojanje od grada i manje ili jednako od rastojanja i i lokacije j;

- $F_{ij} = \{k \in J : d_{ik} > d_{ij}\}$, $i \in I$, $j \in J$ – skup potencijalnih lokacija $k \in J$ za koje je rastojanje od grada i veće od rastojanja i i lokacije j (za sve $i \in I$ i $j \in J$ je $C_{ij} \cup F_{ij} = J$ i $C_{ij} \cap F_{ij} = \emptyset$);
- $S_j = \{i \in I : d_{ij} \leq c\}$, $j \in J$ – skup gradova $i \in I$ koji su na rastojanju ne većem od c od lokacije j;
- $D_j = \{i \in I : d_{ij} > c\}$, $j \in J$ – skup gradova $i \in I$ koji su na rastojanju većem od c od lokacije j (za sve $j \in J$ je $S_j \cup D_j = I$ i $S_j \cap D_j = \emptyset$).

Neka je $k_{t,min}$ celobrojni parametar koji predstavlja donje ograničenje za broj uspostavljenih lokacija u periodu $t \in T$, a k_{max}, $k_{max} \geq \sum_{t \in T} k_{t,min}$ celobrojni parametar koji predstavlja odgovarajuće gornje ograničenje po svim periodima iz T. Za svako $i \in I$ i $j \in J$ definiše se parametar p_{ij}, koji zavisi od rastojanja d_{ij} i konstante c, i računa se na sledeći način:

$$p_{ij} = \min\left\{\frac{|d_{ij} - c|}{c}, 1\right\}.$$

Binarna promenljiva x_{tij} određuje da li je u trenutku t policijska jedinica na lokaciji j u mogućnosti da reaguje u gradu i. Preciznije, promenjiva x_{tij} uzima vrednosti:

$$x_{tij} = \begin{cases} 1, & \text{ako je u trenutku } t \text{ gradu } i \text{ dodeljena lokacija } j, \\ 0, & \text{inače} \end{cases}$$

za sve $t \in T$, $i \in I$ i $j \in J$.

Binarna promenljiva y_{tj} označava da li je specijalna policijska jedinica uspostavljena na lokaciji j u trenutku t i uzima sledeće vrednosti:

$$y_{tj} = \begin{cases} 1, & \text{ako je u trenutku } t \text{ lokacija } j \text{ uspostavljena}, \\ 0, & \text{inače} \end{cases}$$

za sve $t \in T$ i $j \in J$. Nenegativna realna promenljiva z_{max} predstavlja vrednost funkcije cilja.

Cilj je odrediti na kojim lokacijama treba rasporediti specijalne policijske jedinice, tako da maksimalna vrednost izraza

$$\sum_{i \in S_j} f_{ti} x_{tij} + \sum_{i \in D_j} f_{ti}(1 + p_{ij}) x_{tij}$$

po svim $t \in T$ i $j \in J$ bude minimalna.

Pri tome, imajući u vidu prirodu problema, postavlja se uslov da svaki grad treba dodeliti najbližem uspostavljenom resursu. Ukupan broj lokacija na kojima se raspoređuju jedinice po svim periodima je najviše k_{max}. Najmanji broj uspostavljenih jedinica u periodu $t \in T$ je jednak $k_{t,min}$. Parametar p_{ij} povećava vrednost funkcije cilja u slučaju da je lokacija j dodeljena gradu i, gde je udaljenost d_{ij} veća od maksimalnog dozvoljenog rastojanja c.

Koristeći gornju notaciju, problem se matematički može formulisati na sledeći način:

$$\min z_{max} \tag{2.1}$$

uz ograničenja

$$\sum_{j \in J} x_{tij} = 1 \quad \forall t \in T \quad \forall i \in I, \tag{2.2}$$

$$x_{tij} \leq y_{tj} \quad \forall t \in T \quad \forall i \in I \quad \forall j \in J, \tag{2.3}$$

$$y_{tj} \leq \sum_{k \in C_{ij}} x_{tik} \quad \forall t \in T \quad \forall i \in I \quad \forall j \in J, \tag{2.4}$$

$$\sum_{j \in J} y_{tj} \geq k_{t,min} \quad \forall t \in T, \tag{2.5}$$

$$\sum_{t \in T} \sum_{j \in J} y_{tj} \leq k_{max}, \tag{2.6}$$

$$\sum_{i \in S_j} f_{ti} x_{tij} + \sum_{i \in D_j} f_{ti}(1 + p_{ij}) x_{tij} \leq z_{max} \quad \forall t \in T \quad \forall j \in J, \tag{2.7}$$

$$x_{tij} \in \{0, 1\} \quad \forall t \in T \quad \forall i \in I \quad \forall j \in J, \tag{2.8}$$

$$y_{tj} \in \{0, 1\} \quad \forall t \in T \quad \forall j \in J, \tag{2.9}$$

$$z_{max} \geq 0. \tag{2.10}$$

Izraz (2.1) zajedno sa uslovima (2.7) predstavlja funkciju cilja. Ograničenjima (2.2) je omogućeno da se svakom gradu u datom trenutku dodeljuje tačno jedan uspostavljen resurs. Uslovi (2.3) i (2.4) određuju da se gradu i uvek dodeljuje najbliža lokacija j, ukoliko je na toj lokaciji postavljena policijska jedinica. Naime, ako u trenutku t gradu i nije dodeljena lokacija j

niti bilo koja lokacija k za koju je $d_{ik} \leq d_{ij}$, tada sledi da je $\sum_{k \in C_{ij}} x_{tik} = 0$, odakle je i $y_{tj} = 0$, pa tada resurs j ne može biti uspostavljen (videti [28], [58] i [73]). Ograničenje (2.5) označava da broj uspostavljenih lokacija u svakom trenutku t mora biti veći ili jednak parametru $k_{t,min}$, a ograničenje (2.6) da je ukupan broj uspostavljenih lokacija po svim periodima manji ili jednak parametru k_{max}. Uslovi (2.8) – (2.10) se odnose na skupove iz kojih promenljive x_{tij}, y_{tj} i z_{max} uzimaju vrednosti.

Model definisan sa (2.1) – (2.10) je prvi put predložen u ovoj disertaciji i predstavlja uopštenje determinističke matematičke formulacije problema predloženog u [144]. Ako je skup T jednočlan, tada se parametar f_{tj} može zameniti sa f_j, a u promenljivim x_{tij} i y_{tj} indeks t takođe može biti izostavljen. Dodatno, u modelu (2.1) – (2.10) se uvodi donje ograničenje za broj uspostavljenih resursa unutar jednog intervala, $k_{t,min}$. Ovim se obezbeđuje da je za svaki interval t uspostavljena bar jedna policijska jedinica, ukoliko je $k_{t,min} \geq 1$. U praksi je broj uspostavljenih lokacija najčešće jednak k_{max} [144].

2.2 Robusna formulacija

U ovom pododeljku će najpre biti navedeno tvrđenje koje neposredno sledi iz teoreme 1.1, na osnovu koga se može formulisati robusni linearni model za predloženi problem.

Teorema 2.1 Neka je $\mathbf{c} = [c_j]_{j \in J}$ n-dimenzioni vektor, $\mathbf{b} = [b_i]_{i \in I}$ m-dimenzioni vektor, a $A = [a_{ij}]_{i \in I, j \in J}$ i $\hat{A} = [\hat{a}_{ij}]_{i \in I, j \in J}$ matrice dimenzija $m \times n$. Neka su skupovi J_i, $i \in I$ definisani sa $J_i = \{j \in J : \hat{a}_{ij} > 0\}$ i neka su $\Gamma_i \in [0, |J_i|]$, $i \in I$ celobrojni parametri. Tada se problem

$$\min \sum_{j \in J} c_j x_j,$$

$$\sum_{j \in J} a_{ij} x_j + \max_{S_i \subseteq J_i : |S_i| \leq \Gamma_i} \sum_{j \in S_i} \hat{a}_{ij} x_j \leq b_i \quad \forall i \in I,$$

$$x_j \in \{0, 1\} \quad \forall j \in J$$

može formulisati kao problem linearnog programiranja na sledeći način:

$$\min \sum_{j \in J} c_j x_j,$$

$$\sum_{j\in J} a_{ij}x_j + z_i\Gamma_i + \sum_{j\in J_i} p_{ij} \leq b_i \quad \forall i \in I,$$

$$z_i + p_{ij} \geq \hat{a}_{ij}y_j \quad \forall i \in I \quad \forall j \in J_i,$$

$$p_{ij} \geq 0 \quad \forall i \in I \quad \forall j \in J_i,$$

$$z_i \geq 0 \quad \forall i \in I,$$

$$x_j \in \{0,1\} \quad \forall j \in J. \square$$

Neka prosečan broj kriminalnih radnji u gradovima varira u intervalu $[f_{ti}, f_{ti} + \hat{f}_{ti}]$, pri čemu je $\hat{f}_{ti} \geq 0$, $t \in T$, $i \in I$. Dovoljno je posmatrati nesimetričan interval, jer se interval $[f_{ti}, f_{ti} + \hat{f}_{ti}]$ smenom $\bar{f}_{ti} = f_{ti} + \hat{f}_{ti}/2$, $\tilde{f}_{ti} = \hat{f}_{ti}/2$ može svesti na simetričan interval $[\bar{f}_{ti} - \tilde{f}_{ti}, \bar{f}_{ti} + \tilde{f}_{ti}]$. Neka je $G = \{(t,i) \in T \times I : \hat{f}_{ti} > 0\}$ i $\Gamma \in [0, |G|] \cap \mathbb{N}$. Pri istim uslovima, kao kod determinističke varijante problema (2.1) – (2.10), potrebno je odrediti minimum

$$q_{tj} = \sum_{i\in S_j} f_{ti}x_{tij} + \sum_{i\in D_j} f_{ti}(1+p_{ij}) + \max_{P\subset G:|P|\leq\Gamma} \sum_{(t,i)\in P} \hat{f}_{ti}x_{tij}$$

po svim $t \in T$ i $j \in J$. Preciznije, robusna varijanta problema se definiše na sledeći način: odrediti (2.1) uz uslove (2.2) – (2.6), (2.8) – (2.10) i

$$q_{tj} \leq z_{max} \quad \forall t \in T \quad \forall i \in I. \tag{2.11}$$

Imajući prethodno u vidu, na osnovu teoreme 2.1, sledi da se robusna varijanta problema definisana sa (2.1) uz uslove (2.2) – (2.6), (2.8) – (2.11) može formulisati na sledeći način kao problem linearnog programiranja: odrediti (2.1) uz uslove (2.2) – (2.6), (2.8) – (2.10) i

$$\sum_{i\in S_j} f_{ti}x_{tij} + \sum_{i\in D_j} f_{ti}(1+p_{ij})x_{tij} + \Gamma z + \sum_{i\in G} r_i \leq z_{max} \quad \forall t \in T \quad \forall j \in J, \tag{2.12}$$

$$z + r_i \geq \hat{f}_{ti}x_{tij} \quad \forall t \in T \quad \forall i \in G \quad \forall j \in J, \tag{2.13}$$

$$z \geq 0, \tag{2.14}$$

$$r_i \geq 0 \quad \forall i \in G. \tag{2.15}$$

Linearni model definisan sa (2.1) – (2.6), (2.8) – (2.10) i (2.12) – (2.15) predstavlja uopštenje robusne matematičke formulacije problema predloženog u [144]. Ovaj model je prvi put predstavljen u literaturi.

2.3 Složenost problema

U ovom odeljku biće dokazano da je problem NP-težak koristeći redukciju sa problema dominirajućeg skupa.

Problem dominirajućeg skupa (dominating set problem) može biti formulisan na sledeći način: za dati neusmereni graf $G' = (V', E')$, $n = |V|$ i prirodan broj k, $k \leq n$, potrebno je ustanoviti da li postoji skup čvorova $D \subseteq V'$, $|D| = k$ takav da je svaki čvor iz V' ili u D ili je susedan bar jednom čvoru iz D [86]. Problem dominirajućeg skupa je NP-kompletan, što je dokazano u [86].

Teorema 2.2 pokazuje da je specijalan slučaj predloženog problema za koji je $|T| = 1$ NP-kompletan posmatran kao problem odlučivanja. Iz teoreme 2.2 će neposredno slediti da su predloženi deterministički problem, kao i njegova robusna varijanta NP-teški, posmatrani kao optimizacioni problemi.

Teorema 2.2 Neka je za skupove I i J, gde je $I \subseteq J$, definisan težinski neusmereni graf $G = (V, E)$, $V = J$, $E = I \times J$ čije su ivice d_{ij}, $i \in I$, $j \in J$ nenegativni realni brojevi. Svakom čvoru $i \in I$ je pridružena vrednost $f_i \geq 0$. Za parametar $c > 0$, neka je $p_{ij} = \min\{|d_{ij} - c|/c, 1\}$, $i \in I$, $j \in J$. Svaki element $i \in I$ se mora pridružiti tačno onom uspostavljenom čvoru $j \in J$ za koji je vrednost d_{ij} najmanja, pri čemu za broj uspostavljenih čvorova p mora važiti $k_{min} \leq p \leq k_{max}$, gde je $k_{min}, k_{max} \in \mathbb{N}$. Ako je, za sve $j \in J$, P_j skup pridruženih čvorova $i \in I$ čvoru j za koje je $d_{ij} \leq c$, Q_j skup pridruženih čvorova i za koje je $d_{ij} > c$ i $R_j = P_j \cup Q_j$, problem određivanja da li je maksimalna vrednost izraza

$$\sum_{i \in P_j} f_i + \sum_{i \in Q_j} f_i(1 + p_{ij}) \tag{2.16}$$

po svim $j \in J$ jednaka unapred zadatoj vrednosti v je NP-kompletan.

Dokaz. Za proizvoljan ulaz (G', k) problema dominirajućeg skupa, neka je $I = V'$, $J = V' \cup \{n+1, n+2, \ldots, n^2 - 1, n^2\}$, $f_1 = f_2 = \cdots = f_n = 1$,

$c = 1$, $k_{min} = k_{max} = nk$, $v = 1.1$ i za sve $i \in I$ i $j \in J$ neka je parametar d_{ij} definisan sa

$$d_{ij} = \begin{cases} 1.1c, & \text{ako postoji grana izmedju čvorova } i \text{ i } j_0 \text{ u grafu } G', \\ 2c, & \text{inače,} \end{cases}$$

pri čemu je sa j_0, $1 \leq j_0 \leq n$ označen čvor u grafu G' za koji važi $j_0 \equiv j \pmod{n}$.

Ako postoji grana izmedju čvorova i i j_0 u grafu G', biće

$$p_{ij} = \min\left\{\frac{|1.1c - c|}{c}, 1\right\} = \min\left\{\frac{0.1|c|}{c}, 1\right\} = 0.1,$$

a ako ne postoji, važi

$$p_{ij} = \min\left\{\frac{|2c - c|}{c}, 1\right\} = \min\left\{\frac{|c|}{c}, 1\right\} = 1.$$

Kako je $d_{ij} > c$, $i \in I$, $j \in J$, za sve čvorove $j \in J$ važi $P_j = \emptyset$ i $Q_j = R_j$, pa je

$$\sum_{i \in P_j} f_i + \sum_{i \in Q_j} f_i(1 + p_{ij}) = \sum_{i \in R_j} 1 \cdot (1 + p_{ij}) = \sum_{i \in R_j} (1 + p_{ij}). \tag{2.17}$$

Neka je skup čvorova $D = \{d_1, d_2, \ldots, d_k\}$, $1 \leq d_i \leq n$, $1 \leq i \leq k$ dominirajući skup u grafu G', i neka je izvršena particija J na skupove $J_1 = \{1, \ldots, n\}$, $J_2 = \{n+1, \ldots, 2n\}$, $\ldots$, $J_n = \{(n-1)n + 1, \ldots, n^2\}$. Pretpostavlja se da su u svakom od skupova J_l, $1 \leq l \leq n$ uspostavljeni čvorovi $j \in J_l$ za koje postoji $d_i \in D$ takvo da je $d_i \equiv j \pmod{n}$. Broj uspostavljenih čvorova skupa J je tačno nk. Tada, za svako $i \in I$, mora postojati čvor $j \in J_i$ za koji je

$$d_{ij} = 1.1 = \min_{r \in J} d_{ir},$$

odakle sledi da je i $p_{ij} = 0.1$. Neka je izvršena dodela $i \to j$. Svakom $j \in J$ je na ovaj način dodeljen najviše jedan čvor i za koji je $p_{ij} = 0.1$, pa je tada maksimalna vrednost izraza (2.17) jednaka upravo v.

Obratno, ako je skup čvorova $W \subseteq J$, $|W| = nk$ rešenje datog problema za koje je maksimalna vrednost izraza (2.17) jednaka 1.1, tada svaki čvor $i \in I$ mora biti dodeljen onom čvoru $j \in J$ za koji je $d_{ij} = 1.1c$, pa po definiciji d_{ij} mora postojati čvor $j_0 \in V'$ koji je povezan sa čvorom $i \in V'$

i za koji važi $j_0 \equiv j \pmod{n}$. Skup svih takvih čvorova j_0 ima tačno k elemenata i predstavlja dominirajući skup u grafu G'.

Kako se, za fiksirani skup uspostavljenih čvorova u grafu G, može jednostavno u polinomskom vremenu odrediti da li je vrednost izraza (2.16) jednaka zadatoj vrednosti v, sledi da je dati problem NP-kompletan. □

Teoremom 2.2 je dokazano da je specijalan slučaj determinističke varijante predloženog problema za koji je $|T| = 1$ NP-kompletan, ako se on razmatra kao problem odlučivanja. Odatle sledi da je on NP-težak razmatran kao optimizacioni problem. Predloženi problem je NP-težak kao njegovo uopštenje, a takva je i njegova robusna varijanta, budući da važi da, ako je problem NP-težak, odgovarajući robusni problem ostaje takav [14].

2.4 Predloženi memetski algoritam

U ovom odeljku će biti predstavljen memetski algoritam (MA) kao hibridna metaheuristička metoda za rešavanje determinističke i robusne varijante problema. Predloženi MA predstavlja hibridizaciju optimizacije rojem čestica (Particle swarm optimization – PSO) [87] i modifikacije redukovane metode promenljivih okolina (Reduced variable neighbourhood search – RVNS) [111], prilagođene razmatranom problemu. Osnovni koncepti ovih metaheuristika biće opisani u narednim pododeljcima.

Algoritam radi nad populacijom N_r koja se sastoji od $n_r = |N_r|$ rešenja. U svakoj iteraciji algoritma se bira deo manje kvalitetnijih rešenja na koji se primenjuje PSO algoritam, dok se kvalitetnija rešenja propuštaju direktno u narednu fazu. Po završenoj primeni PSO, u cilju dobijanja što kvalitetnijih rešenja, primenjuje se modifikovan RVNS algoritam na svaku jedinku. Ovaj iterativni postupak se ponavlja dok nije zadovoljen kriterijum zaustavljanja. Algoritmom 2.1 prikazan je pseudokod predloženog MA. U nastavku odeljka biće predstavljeni detalji predložene MA implementacije.

2.4.1 Kodiranje rešenja i računanje funkcije cilja

Svako rešenje iz skupa N_r nosi informaciju o tome koje lokacije su uspostavljene i predstavljeno je nizom bitova dužine $|T| \cdot |J|$. Ukoliko je u trenutku $t \in T$ lokacija j uspostavljena, na $((t-1) \cdot |J| + j)$-tom mestu u nizu bitova će se nalaziti bit 1, a inače bit 0. Drugim rečima, niz bitova je podeljen na $|T|$ podnizova dužine $|J|$, a za svaki podniz bit na mestu j označava da li je j-ta lokacija uspostavljena. Pritom, u svakom podnizu koji predstavlja period t, broj bitova koji imaju vrednost 1 ne sme biti manji od $k_{t,min}$. Ukupan broj

Algoritam 2.1 Predloženi memetski algoritam

```
Učitavanje podataka
while nije ispunjen kriterijum zaustavljanja do
    Izdvajanje skupa elitnih rešenja N_e iz skupa N_r koji se direktno prosleđuju u narednu generaciju
    Primena PSO algoritma na rešenja iz skupa N_r \ N_e
    for all r ∈ N_r do
        Primena modifikovanog RVNS metoda na rešenje r
    end for
end while
Ispis rešenja
```

bitova koji imaju vrednost 1 mora biti manji ili jednak k_{max}.

Primer 2.1 Za $T = \{1, 2, 3\}$, $J = \{1, 2, 3\}$, $k_{1,min} = k_{2,min} = k_{3,min} = 1$ i $k_{max} = 4$, jedno dopustivo rešenje je predstavljeno sa 100010010. Pritom se prva tri bita odnose na prvi period, naredna tri na drugi, a poslednja tri na treći. Tako se, podeljeno po periodima, dato rešenje može napisati u obliku 100|010|010, što označava da je u prvom periodu uspostavljena prva lokacija, a u drugom i trećem druga. Ukupan broj bitova koji imaju vrednost 1 je 3 što nije veće od parametra k_{max}, a njihov broj po periodima je jednak upravo dodnjem ograničenju $k_{t,min} = 1$, $t \in T$, pa je rešenje korektno zapisano.

Pri računanju funkcije cilja se najpre, na osnovu koda rešenja, iščitaju lokacije koje su uspostavljene, a zatim se, za svaki period t, kreira skup uređenih parova $(t, j) \in T \times J$ uspostavljenih lokacija j. Nakon toga je potrebno za svaki grad i i svaki period t proći kroz sve parove (t, j) uspostavljenih lokacija i dodeliti grad i onom paru (t, j) za koji je vrednost d_{ij} najmanja od svih uspostavljenih lokacija iz skupa J u periodu t, nakon čega se na ukupnu opterećenost w_{tj} lokacije j u trenutku t dodaje vrednost f_{ti} ukoliko je $d_{ij} \leq c$, odnosno $f_{ti}(1 + p_{ij})$, ukoliko je $d_{ij} > c$. Ukoliko se desi da je za više lokacija vrednost rastojanja d_{ij} najmanja, bira se bilo koje uniformnom raspodelom. Konačno, vrednost funkcije cilja za dato rešenje se dobija kao maksimum po svim vrednostima w_{tj}, $t \in T$, $j \in J$. Složenost računanja funkcije cilja je $O(|T| \cdot |I| \cdot |J|)$.

2.4.2 Optimizacija rojem čestica

Optimizacija rojem čestica (Particle swarm optimization – PSO) je algoritam koji su prvi put predložili Kennedy i Eberhart 1995. godine u radu

[87]. Ideja algoritma je zasnovana na ponašanju pojedinačnih jedinki unutar određene grupe (jato ptica, roj insekata, itd.). Ukoliko se, vođeno instiktom, jato prica uputi u određenom smeru u potrazi za hranom, očekivanje je da će čitavo jato slediti upravo onu pticu koja je pronašla izvor hrane. Međutim, i svaka ptica ponaosob može biti vođena sopstvenim instiktom i time na trenutak u potrazi za hranom napustiti jato. Tada se verovatno može desiti da, ukoliko pronađe bolji izvor hrane, čitavo jato upravo krene da sledi tu pticu [87].

PSO pripada skupu algoritama koji je zasnivaju na inteligenciji roja (swarm intelligence). Osnove algoritma su najpre prikazane u [87] i [137], a prvi put je upotrebljen u [88] za simulaciju društvenog ponašanja. U [128] i [129] može se pronaći detaljan pregled njegove primene na razne optimizacione probleme.

Algoritam PSO radi nad skupom jedinki, koji se naziva rojem. Elementi ovog skupa se nazivaju česticama. Čestice se na unapred definisan način kreću po prostoru pretraživanja. Njihovo kretanje se usmerava imajući u vidu njihovu trenutnu poziciju, njihovu do sada najbolju poziciju, kao i do sada najbolju poziciju čitavog roja. Pod najboljom pozicijom čitavog roja se podrazumeva do sada najbolja pozicija, uzimajući u obzir sva njegova rešenja. Proces se ponavlja dok ne bude zadovoljen kriterijum zaustavljanja, a u svakoj iteraciji se ažurira najbolja vrednost rešenja za svaku česticu, kao i za roj u celini.

Algoritmom 2.2 dat je prikaz osnovne metode optimizacije rojem čestica [30]. Neka je sa X označen roj i neka su svakoj čestici iz skupa X dodeljeni vektori $\mathbf{x}_i \in \mathbb{R}^n$ i $\mathbf{v}_i \in \mathbb{R}^n$, $i \in X$, koji predstavljaju njene vektore pozicije i brzine. Dodatno, n-dimenzionim vektorom $\mathbf{p}_i$ označena je trenutna najbolja pozicija čestice $i \in X$, a n-dimenzionim vektorom $\mathbf{g}$ trenutna globalna najbolja pozicija. Pri inicijalizaciji čestice i, vrednosti koordinata vektora $\mathbf{x}_i$ se biraju uniformno iz skupa (l, u), a vrednosti koordinata vektora $\mathbf{v}_i$ uniformno iz skupa $(-|u-l|, |u-l|)$. Ovde je sa l i u označeno donje i gornje ograničenje pretraživačkog prostora. Pri inicijalizaciji algoritma se dodeljuju početne vrednosti vektorima $\mathbf{p}_i$ i $\mathbf{g}$.

U svakoj iteraciji se za svaku česticu i, vrši ažuriranje vrednosti vektora brzine sa

$$\mathbf{v}_i \leftarrow c_v \mathbf{v}_i + c_p r_p(\mathbf{p}_i - \mathbf{x}_i) + c_g r_g(\mathbf{g} - \mathbf{x}_i),$$

a zatim i vektora trenutne pozicije sa $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$. Parametri r_g i r_p se u svakoj iteraciji biraju uniformno iz intervala $(0, 1)$, dok su c_v, c_p i c_g unapred definisani parametri koji se mogu eventualno i menjati tokom algoritma. Objavljen je i veliki broj radova na temu ažuriranja ovih parametara (videti

npr. [51] i [155]). U svakoj iteraciji se ažurira i vrednost vektora $\mathbf{p}_i$ i $\mathbf{g}$. Konačno, odgovarajuće rešenje je sadržano u vektoru $\mathbf{g}$. Opisani proces se ponavlja sve dok nije ispunjen kriterijum zaustavljanja.

Algoritam 2.2 Optimizacija rojem čestica

for all $i \in N_r$ **do**
  Izabrati koordinate vektora $\mathbf{x}_i$ uniformno iz intervala (l, u)
  $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  if $f(\mathbf{p}_i) < f(\mathbf{g})$ **then**
    $\mathbf{g} \leftarrow \mathbf{p}_i$
  end if
  Izabrati koordinate vektora $\mathbf{v}_i$ uniformno iz intervala $(-|u-l|, |u-l|)$
end for
while nije ispunjen kriterijum zaustavljanja **do**
  for all $i \in N_r$ **do**
    $r_g, r_p \in U(0, 1)$
    $\mathbf{v}_i \leftarrow c_v\mathbf{v}_i + c_p r_p(\mathbf{p}_i - \mathbf{x}_i) + c_g r_g(\mathbf{g} - \mathbf{x}_i)$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
    if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ **then**
      $\mathbf{p}_i \leftarrow \mathbf{x}_i$
    end if
    if $f(\mathbf{p}_i) < f(\mathbf{g})$ **then**
      $\mathbf{g} \leftarrow \mathbf{p}_i$
    end if
  end for
end while

Postoji više načina da se PSO pristup prilagodi diskretnim problemima. Prvi je preslikavanje diskretnog pretraživačkog prostora u kontinualni, primene algoritma na opisan način, a zatim preslikavanje nazad u diskretan prostor [134]. Drugi način predstavlja drugačije predstavljanje vektora brzine i pozicije, kao i modifikovanje operatora. U osnovnoj verziji algoritma vektor pozicije je n-torka realnih brojeva, a korišćeni operatori su standardni aritmetički. Međutim, koordinate vektora pozicije mogu biti i binarni brojevi [31, 32], a operatori mogu biti definisani kao npr. operatori nad skupovima [25].

2.4.3 Primena optimizacije rojem čestica na predloženi problem

Iako algoritam PSO ima dosta sličnosti sa evolutivnim algoritmima, osnovna razlika se ogleda u tome što se u ovom slučaju ne koristi nikakav operator variranja (poput ukrštanja ili mutacije). Kao populacijski deo MA, ovde je iskorišćena optimizacija rojem čestica umesto evolutivnog algoritma, jer je bazirana na jednostavnom konceptu i jednostavnija za implementaciju, dok, sa druge strane, efikasno obezbeđuje dobru diverzifikaciju rešenja u pretraživačkom prostoru.

U predloženoj varijanti PSO, vektor pozicije $\mathbf{x}_i$ je binaran vektor dužine $|T| \cdot |J|$, pri čemu $\mathbf{x}_i$ prestavlja ujedno i kôd i-tog rešenja iz skupa N_r. Vektor brzine $\mathbf{v}_i$ je realan vektor dužine $|T| \cdot |J|$, čije su koordinate iz intervala $[0, 1]$. Početne vrednosti koordinata tih vektora se dodeljuju uniformno, a zatim se pri svakoj iteraciji vrši njihovo ažuriranje. Pritom treba voditi računa da vektor $\mathbf{x}_i$ zadovoljava pravila o broju jedinica imajući u vidu parametre $k_{t,min}$ i k_{max}.

U svakoj iteraciji, l-ti član vektora $\mathbf{x}_i$, $1 \leq l \leq |T| \cdot |J|$ se određuje na sledeći način:

$$\mathbf{x}_{i,l} \leftarrow \begin{cases} 1, & \text{ako je } r < (1 + e^{-\mathbf{v}_{i,l}})^{-1}, \\ 0, & \text{inače.} \end{cases}$$

Pri računanju vrednosti $\mathbf{x}_{i,l}$ koristi se sigmoidalna funkcija $(1+e^{-\mathbf{v}_{i,l}})^{-1}$ [88]. Parametar r se bira uniformno iz intervala $(0, 1)$, a za sve l, $1 \leq l \leq |T| \cdot |J|$ važi $\mathbf{x}_{i,l} \in \{0, 1\}$ i $\mathbf{v}_{i,l} \in [0, 1]$. Pritom je svaki od $|T| \cdot |J|$ bitova podložan promeni u zavisnosti od r. Pri ovakvom računanju vrednosti koordinata vektora $\mathbf{x}_i$, može se dogoditi da se dobije nekorektno rešenje. U tom slučaju se na slučajan način bira skup bitova koji imaju vrednost 1, a koje je potrebno odbaciti u slučaju da je ukupna suma bitova veća od k_{max}, odnosno skup bitova koji imaju vrednost 0, a koje je potrebno promeniti u 1, da bi minimalan broj uspostavljenih lokacija u trenutku t bio veći ili jednak $k_{t,min}$.

Pri ažuriranju vektora brzine čestice i, najpre se određuju koordinate vektora promene brzine na sledeći način:

$$\Delta\mathbf{v}_{i,l} \leftarrow r_p c_p(\mathbf{p}_i - \mathbf{x}_i) + r_g c_g(\mathbf{g} - \mathbf{x}_i) + r_{g'} c_{g'}(\mathbf{g}' - \mathbf{x}_i),$$

gde je $r_p, r_g, r_{g'} \in U(0, 1)$, $c_p = c_g = 1.5$, $c_{g'} = 5$ i $1 \leq l \leq |T| \cdot |J|$. Ovde je sa $\mathbf{g}'$ označeno drugo najbolje globalno rešenje. Ideja koja uzima u obzir drugo najbolje globalno rešenje je upotrebljena u [138], gde je eksperimentalnim putem pokazano da daje bolje rezultate od osnovne verzije PSO algoritma. Vrednosti parametara $c_p, c_g, c_{g'}$ se poklapaju sa vrednostima iz [138].

Koordinate vektora brzine se određuju na sledeći način:

$$\mathbf{v}_{i,l} \leftarrow \begin{cases} 1, & \text{ako je } \mathbf{v}_{i,l} + \Delta\mathbf{v}_{i,l} > 1, \\ 0, & \text{ako je } \mathbf{v}_{i,l} + \Delta\mathbf{v}_{i,l} < 0, \\ \mathbf{v}_{i,l} + \Delta\mathbf{v}_{i,l}, & \text{inače.} \end{cases}$$

Algoritam PSO se primenjuje samo na ne-elitne jedinke. Ukoliko je u odgovarajućoj iteraciji vrednost funkcije cilja za vektor $\mathbf{x}_i$ bolja od vrednosti funkcije cilja za vektor $\mathbf{p}_i$, vrši se njegovo ažuriranje. Analogno se i vrši ažuriranje vektora $\mathbf{g}$ i $\mathbf{g}'$ u zavisnosti od vrednosti vektora $\mathbf{p}_i$ za sve čestice i iz roja.

2.4.4 Metoda promenljivih okolina

Metoda promenljivih okolina (Variable neighbourhood search – VNS) je prvi put predložena u [111] i predstavlja uopštenje lokalne pretrage. Pritom je neophodno definisati okoline $U_l(x)$, $1 \leq l \leq l_{max}$, po kojima će se sistematski vršiti pretraživanje. Metoda promenljivih okolina se bazira na sledećim principima:

- Lokalni minimum za jedan tip okoline ne mora nužno i da bude lokalni minimum za drugi tip okoline.
- Globalni minimum predstavlja lokalni minimum za sve tipove okolina.
- Za mnoge probleme u praksi, lokalni minimumi više tipova okolina su relativno blizu jedan drugog.

Za detaljan prikaz ovog algoritma, videti [76], [111] i [114]. U literaturi postoji nekoliko varijanti osnovnog algoritma, među kojima su:

- Metoda promenljivog spusta (Variable neighbourhood descent – VND) [111] se sastoji u tome da se izabere l_{max} okolina, odredi početno rešenje x i startuje procedura lokalne pretrage redom u odnosu na svaku od okolina, počev od rešenja x. Za $l_{max} = 1$ reč je o običnoj lokalnoj pretrazi. Osnovni koncept metode promenljivog spusta prikazan je algoritmom 2.3.
- Kod redukovane metode promenljivih okolina (Reduced variable neighbourhood search – RVNS) [111], za razliku od osnovne varijante VNS, izostavljena je lokalna pretraga i umesto lokalnog minimuma bira se proizvoljno rešenje iz odgovarajuće okoline od trenutnog u cilju traženja poboljšanja. Osnovni koncept redukovane metode promenljivih okolina prikazan je algoritmom 2.4.

Algoritam 2.3 Metoda promenljivog spusta

Inicijalizacija. Odabrati tipove okolina U_l, $1 \leq l \leq l_{max}$ i generisati početno rešenje x.
while nije zadovoljen kriterijum zaustavljanja **do**
  $l \leftarrow 1$
  while $l \leq l_{max}$ **do**
    Naći rešenje x' u okolini $U_l(x)$ sa najmanjom vrednošću funkcije cilja
    if rešenje x' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x'$, $l = 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
end while

Algoritam 2.4 Redukovana metoda promenljivih okolina

Inicijalizacija. Odabrati tipove okolina U_l, $1 \leq l \leq l_{max}$ i generisati početno rešenje x.
while nije zadovoljen kriterijum zaustavljanja **do**
  $l \leftarrow 1$
  while $l \leq l_{max}$ **do**
    Izabrati proizvoljno rešenje x' u okolini $U_l(x)$
    if rešenje x' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x'$, $l = 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
end while

- Kod osnovne metoda promenljivih okolina (Basic variable neighbourhood search – BVNS) [111] se slučajno bira početno rešenje x u tekućoj okolini, na koje se zatim primenjuje neki tip lokalne pretrage. Osnovna metoda promenljivih okolina prikazana je algoritmom 2.5.

- Kod metode promenljivih okolina sa dekompozicijom (Variable neighbourhood decomposition search – VNDS) [75] se umesto primene lokalnog pretraživanja u čitavom prostoru potencijalnih rešenja, u svakoj iteraciji posmatra njegov potprostor koji sužava pretragu. Ukoliko se za pretraživanje u tom potprostoru koristi takođe VNS, dobija se tzv. dvostruka VNS šema.

- Kod adaptivnog pretraživanja promenljivom okolinom (Skewed variable neighbourhood search – SVNS) [74] se dopušta prelazak u novo rešenje koje je lošijeg kvaliteta od trenutnog sa nekom verovatnoćom. Time se obezbeđuje izmeštanje pretraživačkog prostora u novu okolinu dalje od trenutno najboljeg rešenja.

- Kod uopštene metode promenljivih okolina (General variable neighbourhood search – GVNS) se koristi metoda promenljivog spusta umesto lokalne pretrage, pri čemu se čemu se tada mogu koristiti dva tipa okolina – jedan za fazu razmrdavanja, a drugi za fazu lokalne pretrage. Osnovna struktura ove metode prikazana je algoritmom 2.6.

Algoritam 2.5 Osnovna metoda promenljivih okolina

Inicijalizacija. Odabrati tipove okolina U_l, $1 \leq l \leq l_{max}$ i generisati početno rešenje x.
while nije zadovoljen kriterijum zaustavljanja **do**
  $l \leftarrow 1$
  while $l \leq l_{max}$ **do**
    Izabrati proizvoljno rešenje x' u okolini $U_l(x)$
    Primenom nekog tipa lokalne pretrage na početno rešenje x' naći rešenje x'' sa najmanjom vrednošću funkcije cilja
    if rešenje x'' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x''$, $l = 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
end while

Algoritam 2.6 Uopštena metoda promenljivih okolina

Inicijalizacija. Odabrati tipove okolina U_l, $1 \leq l \leq l_{max}$ za fazu razmrdavanja i tipove okolina U'_l, $1 \leq l \leq l'_{max}$ za fazu lokalne pretrage. Generisati početno rešenje x i poboljšati ga korišćenjem redukovane metode promenljivih okolina
while nije zadovoljen kriterijum zaustavljanja **do**
  $l \leftarrow 1$
  while $l \leq l_{max}$ **do**
    Izabrati proizvoljno rešenje x' u okolini $U_l(x)$
    $l' \leftarrow 1$
    while $l' \leq l'_{max}$ **do**
      Naći rešenje x'' u okolini $U_{l'}(x')$ sa najmanjom vrednošću funkcije cilja
      if rešenje x'' ima manju vrednost funkcije cilja od rešenja x' **then**
        $x' \leftarrow x''$, $l' = 1$
      else
        $l' \leftarrow l' + 1$
      end if
    end while
    if rešenje x'' ima manju vrednost funkcije cilja od rešenja x **then**
      $x \leftarrow x''$, $l = 1$
    else
      $l \leftarrow l + 1$
    end if
  end while
end while

2.4.5 Primena modifikovanog RVNS na predloženi problem

Redukovana metoda promenljivih okolina je varijanta koja se često koristi u literaturi kada je algoritmu zasnovanom na populaciji jedinki neophodan mali, ali efikasan impuls da se usmeri ka kvalitetnijim rešenjima i izbegne zamku lokalnog optimuma [119] ili pri rešavanju instanci problema velikih dimenzija, gde bi pretraživanje cele okoline nekog rešenja tokom lokalne pretrage znatno povećalo vreme izvršavanja algoritma [67].

Na razmatrani problem se u okviru svake iteracije memetskog algoritma primenjuje modifikovana verzija redukovane metode promenljivih okolina. Preliminarna testiranja su pokazala da se izuzimanjem lokalne pretrage iz metode promenljivih okolina u ovom slučaju ne smanjuje kvalitet dobijenog rešenja. Sa druge strane, znatno je povećana efikasnost memetskog algoritma u celini, jer se tokom izvršavanja RVNS ne pretražuje čitava okolina trenutnog rešenja. Pritom su korišćeni sledeći tipovi okolina $U_l(r)$, $l = 1, 2, 3$:

- $U_1(r)$ – skup svih rešenja koja se dobijaju kada se u rešenju r unutar istog perioda $t \in T$ izaberu dva bita (t, i) i (t, j) $(i, j \in J)$, čije su vrednosti različite, i izvrši njihova razmena. Uređen par (t, j) označava bit koji se nalazi na $((t-1) \cdot |J| + j)$-toj poziciji u rešenju r.

- $U_2(r)$ – skup svih rešenja koja se dobijaju kada se u rešenju r unutar različitih perioda $t_1, t_2 \in T$ izaberu dva bita (t_1, i) i (t_2, j) $(i, j \in J)$, čije su vrednosti različite, i izvrši njihova razmena.

- $U_3(r)$ – skup svih rešenja koja se dobijaju kada se u rešenju r vrši inverzija proizvoljno izabranog bita (t, j), $t \in T$, $i \in J$.

Algoritmom 2.7 je prikazan pseudokod modifikovanog RVNS metoda prilagođenog za rešavanje razmatranog problema. U svakoj iteraciji memetskog algoritma, modifikovan RVNS se primenjuje na svaku jedinku po 10 puta u cilju dobijanja što kvalitetnijeg rešenja. Za svako rešenje r se razmatraju tri tipa okolina, pri čemu se u naredni tip okoline prelazi ukoliko proizvoljno generisano rešenje r' iz date okoline $U_l(r)$ nije bolje od r. Inače, pretraga se nastavlja od prvog tipa okoline. Rešenje r' se smatra boljim od rešenja r ukoliko je vrednost $f(r')$ manja od vrednosti $c_r \cdot f(r)$, gde je sa f označena funkcija cilja. Ukoliko je $l = 3$, svi tipovi okolina su u trenutnoj iteraciji razmotreni i jedna iteracija modifikovanog RVNS algoritma se time prekida. Vrednost parametra c_r iz algoritma je bliska jedinici i biće naknadno utvrđena. Time je omogućeno da se dodatno izbegne kovergencija ka lokalnom minimumu.

Algoritam 2.7 Primena modeifikovanog RVNS metoda tokom jedne iteracije MA

```
1: for all r ∈ N_r do
2:    l ← 1
3:    while l ≤ l_max do
4:       Razmrdavanje: generisati proizvoljno rešenje r' ∈ U_l(r)
5:       if f(r') < c_r · f(r) then
6:          r ← r'
7:          l ← 1
8:       else
9:          l ← l + 1
10:      end if
11:   end while
12: end for
```

Preliminarni eksperimentalni rezultati su pokazali da je dovoljno koristiti gore definisane okoline U_1, U_2 i U_3, da bi modifikovani RVNS dao impuls predloženom MA i omogućila brza konvergencija ka optimalnim, odnosno najboljim poznatim rešenjima. Koncept modifikovanog RVNS metoda se može proširiti tako sto se mogu pretraživati i šire okoline tipa U_1, U_2 i U_3 tekućeg rešenja, ali je u ovom slučaju predložena metoda dala dobre rezultate u smislu kvaliteta rešenja memetskog algoritma i vremena izvršavanja.

Svaki od tipova okolina $U_l(r)$ se sastoji od inverzije bita čija je vrednost 0 u 1, inverzije bita čija je vrednost 1 u 0 ili oba ta koraka. Umesto ponovnog računanja funkcije cilja, u oba slučaja se može iskoristiti njena trenutna vrednost, gde se, pri prelasku u novo rešenje, u obzir uzimaju samo promene koje se dešavaju pri invertovanju jednog bita.

Neka je, pri inverziji bita iz 1 u 0, za rešenje r, bit na poziciji $(t-1)\cdot|J|+j$ za neke $t \in T$ i $j \in J$ promenio vrednost. To znači da je u periodu t na lokaciji j odgovarajući resurs više nije uspostavljen. Ukupna suma f_{ti} po svim gradovima i koji su pridruženi lokaciji j u trenutku t sada postaje 0, a oni se dalje pridružuju najbližoj od preostalih uspostavljenih lokacija i time se ažuriraju nove vrednosti tih lokacija. Budući da se samo posmatra skup J_i, čiji su elementi gradovi pridruženi resursu j, time se ne računa funkcija cilja u celini, čime se složenost računanja smanjuje sa $O(|T| \cdot |I| \cdot |J|)$ na $O(|J_i| \cdot |J|)$.

Pri inverziji bita iz 0 u 1, neka je za rešenje r bit na poziciji $(t-1)\cdot|J|+j$ za neke $t \in T$ i $j \in J$ promenio vrednost. To znači da u periodu t na lokaciji j odgovarajući resurs sada uspostavljen. Za sve gradove $i \in I$ se proverava da li je vrednost d_{ij} manja od udaljenosti od i i njemu trenutno

najbliže uspostavljene lokacije k. Ako jeste, uvećava se vrednost ukupne sume pridruženih elemenata resursu j za f_{ti}, a za isto toliko se smanjuje odgovarajuća suma resursa k. Ovaj korak smanjuje vreme izvršavanja sa $O(|T| \cdot |I| \cdot |J|)$ na $O(|I|)$, budući da nije potrebno računati funkciju cilja u celini.

2.4.6 Ostali aspekti algoritma

Početni skup rešenja se generiše uniformno, pri čemu svako rešenje na početku sadrži tačno k_{max} jedinica. Pri tom odabiru treba jedino voditi računa, pored fiksiranog broja jedinica, da je u svakom periodu t broj bitova koji odgovaraju uspostavljenim lokacijama veći ili jednak $k_{t,min}$. Tokom izvršavanja memetskog algoritma, primenjuje se elitistički pristup, koji podrazumeva da se PSO algoritam primenjuje samo na ne-elitna rešenja, tj. na deo skupa rešenja sa najmanjom vrednošću funkcije cilja. Elitna rešenja (čestice) se direktno prosleđuju u narednu fazu, što obezbeđuje da takva rešenja budu očuvana. Broj čestica u roju $|N_r|$, kao i procenat elitnih jedinki p_{el} su parametri algoritma čija će vrednost biti eksperimentalno određena u cilju postizanja što boljih performansi algoritma. Broj elitnih jedinki p_{el} se pri testiranju uvek zaokružuje na najbliži ceo broj. Algoritam redukovane metode promenljivih okolina se, u cilju dobijanja što kvalitetnijih rešenja, primenjuje na sve elemente skupa N_r. Memetski algoritam se zaustavlja nakon što se najbolje rešenje ponovi rep puta. Parametar rep će takođe biti određen eksperimentalnim putem.

2.4.7 Slučaj $\Gamma > 0$

U ovom podeljku biće formulisana i dokazana teorema 2.3 kojom se obezbeđuje da je u slučaju robusne varijante problema dovoljno posmatrati slučaj kada su svi parametri $f_{ti}, t \in T, i \in I$ fiksirani, odnosno kada je $\Gamma = 0$. Prema teoremi 2.3 vrednost funkcije cilja za $\Gamma > 0$ se može jednostavno odrediti koristeći slučaj $\Gamma = 0$.

Teorema 2.3 Neka su $f_1, f_2, \ldots, f_n, f_{n+1}$ nenegativni realni brojevi takvi da je $f_1 \geq f_2 \geq \cdots \geq f_n \geq f_{n+1} = 0$ i neka je Γ celobrojni parametar iz skupa $\{0, 1, \ldots, n\}$. Ako je za parametre $z, r_1, r_2, \ldots, r_n \geq 0$ ispunjeno $z + r_i \geq f_i$, $i = 1, \ldots, n$, tada izraz

$$\Gamma z + \sum_{i=1}^{n} r_i \tag{2.18}$$

ima najmanju vrednost za $z = f_{\Gamma+1}$.

Dokaz. Neka je najpre, za sve k, $1 \leq k \leq n+1$ ispunjeno $z \neq f_k$. Tada postoji vrednost j takva da je $z = f_j - t$, gde je $t > 0$. Neka je $z' = f_j = z + t$ i $r'_i = r_i - t$, $1 \leq i \leq n$. Važi da je $z' + r'_i = (z' - t) + (r'_i + t) = z + r_i \geq f_i$ i

$$\Gamma z + \sum_{i=1}^{n} r_i = \Gamma(z' - t) + \sum_{i=1}^{n}(r'_i + t) = \Gamma z' + \sum_{i=1}^{n} r'_i + (n - \Gamma)t.$$

Kako je $0 \leq \Gamma \leq n$, to je

$$\Gamma z + \sum_{i=1}^{n} r_i \geq \Gamma z' + \sum_{i=1}^{n} r'_i$$

Iz poslednjeg sledi da je dovoljno posmatrati slučaj kada je $z = f_k$ za $k \in \{1, \ldots, n+1\}$.

Neka je sada $z = f_k$ za neko $k \in \{1, \ldots, n+1\}$. Da bi izraz (2.18) bio minimalan, mora važiti

$$r_i = \begin{cases} f_i - z, & \text{ako je } f_i \geq z, \\ 0, & \text{ako je } f_i < z, \end{cases}$$

$1 \leq i \leq n$, odnosno

$$r_i = \begin{cases} f_i - z, & \text{ako je } i \leq k, \\ 0, & \text{ako je } i > k. \end{cases}$$

Izraz (2.18) se može napisati u obliku

$$\sum_{i=1}^{n} r_i + \Gamma z = \sum_{i=1}^{k}(f_i - f_k) + \Gamma f_k = \sum_{i=1}^{k} f_i + (\Gamma - k) f_k.$$

Razlikuju se tri slučaja:

1° Ako je $k = \Gamma + 1$, tada je

$$\sum_{i=1}^{k} f_i + (\Gamma - k) f_k = \sum_{i=1}^{\Gamma+1} f_i + (\Gamma - \Gamma - 1) f_{\Gamma+1} = \sum_{i=1}^{\Gamma} f_i + f_{\Gamma+1} - f_{\Gamma+1} = \sum_{i=1}^{\Gamma} f_i.$$

2° Ako je $k > \Gamma + 1$, dobija se

$$\sum_{i=1}^{k} f_i + (\Gamma - k) f_k = \sum_{i=1}^{\Gamma} f_i + \sum_{i=\Gamma+1}^{k} f_i + (\Gamma - k) f_k = \sum_{i=1}^{\Gamma} f_i + \sum_{i=\Gamma+1}^{k} (f_i - f_k).$$

Kako je $f_i \geq f_k$ za sve $i \in \{\Gamma + 1, \ldots, k\}$, to je

$$\sum_{i=1}^{\Gamma} f_i + \sum_{i=\Gamma+1}^{k} (f_i - f_k) \geq \sum_{i=1}^{\Gamma} f_i.$$

3° Ako je $k < \Gamma + 1$, sledi

$$\sum_{i=1}^{k} f_i + (\Gamma - k) f_k = \sum_{i=1}^{\Gamma} f_i - \sum_{i=k+1}^{\Gamma} f_i + (\Gamma - k) f_k = \sum_{i=1}^{\Gamma} f_i + \sum_{i=k+1}^{\Gamma} (f_k - f_i).$$

Za sve $i \in \{k + 1, \ldots, \Gamma\}$ je ispunjeno $f_k \geq f_i$, odakle se dobija

$$\sum_{i=1}^{\Gamma} f_i + \sum_{i=k+1}^{\Gamma} (f_k - f_i) \geq \sum_{i=1}^{\Gamma} f_i.$$

Dakle, najmanja vrednost izraza (2.18) se dobija za $z = f_{\Gamma+1}$ i ona iznosi $\sum_{i=1}^{\Gamma} f_i$. □

Na osnovu teoreme 2.3 i uslova (2.12), može se zaključiti da važi

$$F_{min}(\Gamma) = F_{min}(0) + \Gamma z + \sum_{i \in K} r_i = F_{min}(0) + \sum_{k=1}^{\Gamma} \hat{f}_{t_k i_k},$$

pri čemu funkcija $F_{min} : \{0, \ldots, n\} \to \mathbb{R}$ predstavlja vrednost funkcije cilja za fiksirano Γ, a niz $\hat{f}_{t_k i_k}$, $1 \leq k \leq |T| \cdot |I|$ predstavlja permutaciju niza $\hat{f}_{ti}$, $t \in T$, $i \in I$ za koju je

$$\hat{f}_{t_1 i_1} \geq \hat{f}_{t_2 i_2} \geq \cdots \geq \hat{f}_{t_{|T|\cdot|I|} i_{|T|\cdot|I|}}.$$

Kako je predloženi problem uopštenje problema iz [144], analogan zaključak se može primeniti pri računanju funkcije cilja tog problema za $\Gamma > 0$.

2.5 Eksperimentalni rezultati

U ovom odeljku će biti predstaljveni rezultati memetskog algoritma. Implementacija algoritma je izvršena u programskom jeziku C++, a rešavač CPLEX 12.1 je takođe inkorporiran u C++. Za svaku instancu, MA je pokretan po 15 puta. Sva testiranja su izvršena pod Windows 7 operativnim sistemom sa procesorom Intel i5-2430M od 2.4 GHz i RAM memorijom od 8 GB.

2.5.1 Test instance

Za testiranje memetskog algoritma iskorišćene su realne instance iz [144], dobijene na osnovu geografskog položaja 165 gradova i 234 potencijalne lokacije za specijalne policijske jedinice na teritoriji Republike Srbije. Na osnovu geografskih koordinata su generisana rastojanja između gradova i lokacija. Najveća instanca sadrži sva naseljena mesta i potencijalne lokacije, dok su instance manjih dimenzija dobijene grupisanjem nasaljenih mesta, koja su po geografskom položaju međusobno blizu [144].

Za testiranje MA za $|T| = 1$ korišćene su instance i12, i6_7_8, i1_2_3_4 i i_all iz [144], nad kojima je izvršeno i poređenje rezultata sa EA-LS algoritmom predloženim u [144]. Dodatno, generisano je još osam instanci za slučajeve sa dva i tri perioda. Način generisanja novih instanci polazeći od instanci iz [144] predstavljen je u tabeli 2.1. Broj gradova i potencijalnih lokacija instanci sa dva ili tri perioda odgovara broju gradova i potencijalnih lokacija instance sa jednim periodom od koje je ona generisana, a naziv je dobijen na osnovu naziva instance od koje je generisana i broja perioda. Na primer, naziv i12_t2 se odnosi na instancu sa dva perioda koja je generisana od instance i12 sa jednim periodom, koja je korišćena u [144]. Nazivi kolona čija su zaglavlja $|I|$, $|J|$ i $|T|$ odgovaraju redom broju gradova, potencijalnih lokacija i vremenskih peroda. Vrednosti parametara f_{ti} su generisane od odgovarajućih vrednosti f_i iz [144], koje su u slučaju jednog perioda podeljene sa 12, budući da se raspoređivanje specijalnih jedinica vrši na mesečnoj osnovi. Slično se postupa i za $|T| = 2$ i $|T| = 3$, pri čemu se u slučaju dva perioda vrednosti f_{ti} odnose sa $1 : 2$, a u slučaju tri perioda sa $1 : 1 : 2$. Analogno je dobijena i vrednost $\hat{f}_{ti}$, $t \in T$, $i \in I$ od parametra $d_i = 0.05 f_i$ iz [144]. Za svaku od korišćenih 12 test instanci razmotrene su različite vrednosti parametara k_{max} i Γ.

Tabela 2.1: Instance korišćene za testiranje predloženog MA

Naziv	Opis	$\|I\|$	$\|J\|$	$\|T\|$	f_{ti}
i12	korišćena u [144]	17	21	1	$f_{1i} = f_i/12$
i6_7_8	korišćena u [144]	32	46	1	$f_{1i} = f_i/12$
i1_2_3_4	korišćena u [144]	62	95	1	$f_{1i} = f_i/12$
i_all	korišćena u [144]	165	234	1	$f_{1i} = f_i/12$
i12_t2	gen. na osnovu i12	17	21	2	$f_{1i} = f_i/36$, $f_{2i} = f_i/18$
i6_7_8_t2	gen. na osnovu i6_7_8	32	46	2	$f_{1i} = f_i/36$, $f_{2i} = f_i/18$
i1_2_3_4_t2	gen. na osnovu i1_2_3_4	62	95	2	$f_{1i} = f_i/36$, $f_{2i} = f_i/18$
i_all_t2	gen. na osnovu i_all	165	234	2	$f_{1i} = f_i/36$, $f_{2i} = f_i/18$
i12_t3	gen. na osnovu i12	17	21	3	$f_{1i} = f_{2i} = f_i/48$, $f_{3i} = f_i/24$
i6_7_8_t3	gen. na osnovu i6_7_8	32	46	3	$f_{1i} = f_{2i} = f_i/48$, $f_{3i} = f_i/24$
i1_2_3_4_t3	gen. na osnovu i1_2_3_4	62	95	3	$f_{1i} = f_{2i} = f_i/48$, $f_{3i} = f_i/24$
i_all_t3	gen. na osnovu i_all	165	234	3	$f_{1i} = f_{2i} = f_i/48$, $f_{3i} = f_i/24$

2.5.2 Podešavanje parametara predloženog memetskog algoritma

Iz opisa memetskog algoritma za rešavanje predloženog problema može se uočiti da MA zavisi od nekoliko parametara. U cilju obezbeđivanja što boljih performansi MA, izvršena je analiza i testiranje sledećih parametara:

- $|N_r|$ – broj čestica u roju N_r;
- c_r – koeficijent pri prelasku u naredno rešenje za modifikovanu RVNS metodu;
- rep – maksimalni broj ponavljanja najbolje jedinke, koji predstavlja i kriterijum zaustavljanja;
- p_{el} – procenat elitnih jedinki iz algoritma.

2.5.2.1 Analiza varijanse

Analiza varijanse (analysis of variance – ANOVA) je statistička tehnika koja se koristi za upoređivanje značajnosti razlike nad više grupa ispitanika [113]. Pritom se testira nulta hipoteza da su jednake srednje vrednosti odgovarajućih grupa. Prema broju faktora koji deluju na rezultujuće obeležje, ANOVA može biti jednofaktorska, dvofaktorska i višefaktorska [113].

Neka je, na primer, potrebno odrediti da li promena vrednosti parametra p određenog heurističkog algoritma za neki optimizacioni problem ima uticaja na vrednost dobijenog rešenja. Neka posmatrani parametar p ima k potencijalnih vrednosti $p_1, p_2, \ldots, p_k$ i neka je za svaku fiksiranu vrednost

parametra p heuristika testirana na n različitih ulaznih instanci. Dobijene numeričke vrednosti su predstavljene sledećom matricom:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}.$$

Kolone matrice X odgovaraju vrednostima rešenja za fiksiranu vrednost parametra, a redovi za odgovarajuću ulaznu instancu i razne vrednosti parametra p. Elementi j-te kolone $[x_{1j}, x_{2j}, \dots, x_{nj}]^T$ predstavljaju vrednosti funkcije cilja rešenja za parametar p_j, a elementi i-te vrste $[x_{i1}, x_{i2}, \dots, x_{ik}]$ vrednosti funkcije cilja rešenja za i-tu ulaznu instancu.

U ovoj disertaciji je korišćena jednoparametarska analiza varijanse nad numeričkim podacima. Pritom se testira nulta hipoteza H_0 koja glasi: promena parametra p nema uticaja na promenu vrednosti funkcije cilja rešenja. Parametar p se naziva još i faktor, a njegove vrednosti nivoi. Kolone matrice X se mogu posmatrati kao grupe nad kojima je zabeležena vrednost funkcije cilja rešenja za odgovarajući nivo faktora p. Kod jednoparametarske analize varijanse se najpre računa srednja vrednost za svaku grupu

$$\overline{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij} \quad j = 1, \dots, k,$$

a nakon toga i srednja vrednost po svim grupama $\overline{x} = \frac{1}{k} \sum_{j=1}^{k} \overline{x}_j$. Suma kvadrata rastojanja grupa od srednje vrednosti je jednaka $n \sum_{j=1}^{k} (\overline{x}_j - \overline{x})^2$, a broj stepeni slobode između grupa $k-1$. Odatle sledi da je prosečna vrednost sume kvadrata između grupa jednaka njihovom količniku i iznosi

$$\frac{n}{k-1} \sum_{j=1}^{k} (\overline{x}_j - \overline{x})^2.$$

Slično, suma kvadrata unutar grupe je data sa $\sum_{j=1}^{k} \sum_{i=1}^{n} (\overline{x} - \overline{x}_j)^2$, a broj stepeni slobode unutar grupe iznosi $k(n-1)$. Prosečna vrednost sume kvadrata unutar grupe je jednaka njihovom količniku i iznosi

$$\frac{1}{n(k-1)} \sum_{j=1}^{k} \sum_{i=1}^{n} (\overline{x} - \overline{x}_j)^2.$$

Vrednost F-testa predstavlja količnik prosečne vrednosti sume kvadrata između grupa i prosečne vrednosti sume kvadrata unutar grupe i jednaka je

$$F = n^2 \sum_{j=1}^{k} (\overline{x}_j - \overline{x})^2 \left(\sum_{i=1}^{n} (\overline{x} - \overline{x}_j)^2 \right)^{-1}.$$

Na osnovu vrednosti stepeni slobode između grupa i stepeni slobode unutar grupe određuje se vrednost kritične vrednosti F_k na osnovu Fišerove raspodele za fiksiran prag značajnosti [113]. Ukoliko je $F > F_k$, polazna hipoteza H_0 se odbacuje, a inače se prihvata. Verovatnoća dobijanja rešenja koje je bolje ili jednako onom koje predstavlja rezultat eksperimenta, pod pretpostavkom da važi hipoteza H_0, naziva se p-vrednost.

2.5.2.2 Analiza varijanse za parametre MA

Za parametre memetskog algoritma uzete su sledeće potencijalne vrednosti, koje su se u preliminarnim testiranjima pokazale pogodnim:

- parametar $|N_r|$ ima dva nivoa – 30 i 60;
- parametar c_r ima tri nivoa – 1, 1.001 i 1.005;
- parametar rep ima dva nivoa – 1000 i 2000;
- parametar p_{el} ima dva nivoa – 66.67% i 75%.

Ukupan broj kombinacija iznosi $2 \cdot 3 \cdot 2 \cdot 2 = 24$. U cilju testiranja svih ovih kombinacija, od skupa svih instanci su izabrane četiri reprezentativne: i12, i6_7_8_t2, i1_2_3_4_t3 i i_all. Na taj način je svaka kombinacija parametara za svaku instancu pokretana po 15 puta i pritom je za svaku kombinaciju zabeležen najbolji rezultat. Ukupan broj pokretanja algoritma iznosi $24 \cdot 4 \cdot 15 = 1440$.

Značenja naziva kolona u tabeli 2.2 su sledeća:

- Instanca – naziv instance nad kojom se testiraju parametri;
- Par – parametar koji se testira;
- SK – suma kvadrata između grupa;
- SS – broj stepeni slobode između grupa;
- PK – prosečna vrednost sume kvadrata između grupa;
- F – vrednost F-testa;
- p – p-vrednost.

Tabela 2.2: Rezultati primene analize varijanse na podešavanje parametara

Instanca	Parametar	SK	SS	PK	F	p-vrednost
i12	$\|N_r\|$	0.000	1	0.000	0.0000	1.0000
i12	c_r	0.000	2	0.000	0.0000	1.0000
i12	rep	0.000	1	0.000	0.0000	1.0000
i12	p_{el}	0.000	1	0.000	0.0000	1.0000
i6_7_8_t2	$\|N_r\|$	0.133	1	0.133	1.4268	0.6008
i6_7_8_t2	c_r	1.008	2	0.504	9.0634	0.0015
i6_7_8_t2	rep	0.246	1	0.246	2.1214	0.4879
i6_7_8_t2	p_{el}	0.016	1	0.016	0.1992	0.9254
i1_2_3_4_t3	$\|N_r\|$	0.000	1	0.000	0.0000	1.0000
i1_2_3_4_t3	c_r	0.000	2	0.000	0.0000	1.0000
i1_2_3_4_t3	rep	0.000	1	0.000	0.0000	1.0000
i1_2_3_4_t3	p_{el}	0.000	1	0.000	0.0000	1.0000
i_all	$\|N_r\|$	1.464	1	1.464	2.0260	0.5013
i_all	c_r	0.542	2	0.271	0.3387	0.7165
i_all	rep	0.203	1	0.203	0.2599	0.9041
i_all	p_{el}	1.497	1	1.497	2.0763	0.4942

Kritična vrednost za ovaj test iznosila je 0.05. Iz tabele 2.2 se može uočiti da jedino faktor c_r ima značajnijeg uticaja na vrednost funkcije cilja, što se može zaključiti na osnovu vrednosti p za instancu i6_7_8_t2. Za instance i12 i i1_2_3_4_t3, pri testiranju, svaka kombinacija parametra je dala isti rezultat, pa u tom slučaju se pokazalo da promena parametara nema nikakvog uticaja. Slično, na osnovu rezultata se može uočiti da parametri $|N_r|$, rep i p_{el} nemaju značajnijeg uticaja na promenu vrednosti rezultata. U svim silučajevima vrednost $c_r = 1.001$ se pokazala najboljom, a za ostale parametre pri testiranju svih instanci je odabrano $|N_r| = 30$, $rep = 1000$ i $p_{el} = 75\%$.

2.5.3 Rezultati i poređenja za slučaj $|T| = 1$

Rezultati predloženog MA koji su dobijeni na instancama sa jednim periodom su upoređeni sa rezultatima EA-LS metode iz rada [144]. U tabelama 2.3 – 2.7 su prikazani rezultati testiranja na pomenutim instancama za različite vrednosti maksimalnog broja lokacija na kojima treba rasporediti jedinice i različite nivoe robusnosti. Značenja odgovarajućih zaglavlja su sledeća:

- k_{max} – maksimalni broj uspostavljenih lokacija;

- Γ – nivo robusnosti;
- Rešenje – vrednost funkcije cilja koja odgovara optimalnom ili najboljem poznatom rešenju;
- $t_{CPLEX}[s]$ – vreme izvršavanja CPLEX 12.1 rešavača za robusni model iz [144];
- $t^{napr}_{CPLEX}[s]$ – vreme izvršavanja CPLEX 12.1 rešavača za robusni model iz [144], imajući u vidu teoremu 2.3;
- EA-LS$_{najb}$ – vrednost funkcije cilja najboljeg rešenja EA-LS metode iz [144] za datu instancu (ako je vrednost optimalna, upisana je oznaka *opt*, a inače je korišćena oznaka $najb$);
- MA$_{najb}$ – vrednost funkcije cilja najboljeg rešenja memetskog algoritma (ako je vrednost optimalna, upisana je oznaka *opt*, a inače je korišćena oznaka $najb$);
- $t_{\text{EA-LS}}[s]$ – prosečno vreme za koje je prvi put dostignuto najbolje rešenje tokom izvršavanja EA-LS algoritma iz [144];
- $t_{\text{MA}}[s]$ – prosečno vreme za koje je prvi put dostignuto najbolje rešenje tokom izvršavanja memetskog algoritma;
- Pov[%] – procenat za koji se povećala vrednost koja odgovara optimalnom (ili najboljem poznatom rešenju) za dati nivo robusnosti Γ.

Iz teoreme 2.3 sledi da se za $\Gamma > 0$ rešenje jednostavno može dobiti od rešenja za $\Gamma = 0$. Imajući to u vidu, za dobijanje optimalnih rešenja na instancama manjih dimenzija, pri implementaciji CPLEX-a iskorišćen je deterministički model koji ima $|I||J|+|J|+1$ promenljivih i $2|I||J|+|I|+3|J|+3$ ograničenja, što ubrzava proces dobijanja rešenja. U radu [144] je pri implementaciji CPLEX-a korišćen složeniji robusni model, koji ima $|I||J| + |J| + |G| + 2$ ograničenja i $2|I||J| + |G||J| + |I| + 3|J| + |G| + 4$ promenljivih. Koristeći rezultate teoreme 2.3, skraćeno je i ukupno vreme izvršavanja CPLEX-a, budući da je dovoljno razmotriti samo slučajeve za $\Gamma = 0$. Iz istog razloga skraćeno je i ukupno vreme izvršavanja memetskog algoritma, budući da je dovoljno posmatrati deterministički slučaj. Zbog načina na koji su dobijena rešenja za $\Gamma > 0$, vremena izvršavanja za odgovarajuću instancu i fiksiran parametar k_{max} su jednaka za sve $\Gamma \in \{0, \ldots, |I|\}$.

EA-LS je testiran na računaru sa Intel Core i7-860 procesorom sa 2.8 GHz i 8 GB RAM memorije, a memetski algoritam na računaru sa procesorom

Intel i5-2430M sa 2.4 GHz i RAM memorijom od 8 GB. Time su preformanse računara na kom je testiran EA-LS algoritam nešto jače u odnosu na performanse računara nad kojim je testiran predloženi memetski algoritam. Zbog razlike u platformama nad kojima su testirane EA-LS i MA metoda, izvršena je normalizacija prosečnog vremena izvršavanja za EA-LS, koristeći pristup iz [45] i na osnovu podataka sa http://www.cpubenchmark.net/. Prosečno normalizovano vreme EA-LS algoritma se izračunava na sledeći način:

$$\text{NAT(EA-LS)} = \text{AT(EA-LS)} \cdot \frac{\text{PCPUS(Intel Core i7-860 2.8 GHz)}}{\text{PCPUS(Intel Core i5-2430M 2.4 GHz)}},$$

gde AT(EA-LS) predstavlja prosečno vreme izvršavanja EA-LS algoritma iz [144], a PCPUS predstavlja Passmark CPU skor.

U tabeli 2.3 prikazani su rezultati za instancu i12. Razmotrene su tri različite vrednosti za parametar k_{max}: 3, 4 i 5. Oba algoritma, EA-LS i MA, dostigla su optimalna rešenja, prethodno dobijena pomoću CPLEX-a. U poslednjoj koloni tabele može se uočiti na koji način vrednost funkcije cilja optimalnog rešenja raste sa povećanjem parametra Γ. Za $k_{max} = 3$ i $\Gamma = 17$, najveće povećanje vrednosti funkcije cilja optimalnog rešenja iznosi 10.919%. Za $\Gamma \geq 17$, i $k_{max} \in \{4, 5\}$, vrednosti funkcije cilja su uvećane za 15.232 procenta. Prosečno vreme izvršavanja po svim vrednostima parametara k_{max} i Γ za instancu i12 EA-LS algoritma iznosilo je 0.019 s, dok je odgovarajuće normalizovano vreme iznosilo 0.03 s. Prosečno vreme memetskog algoritma iznosilo 0.92 s. Takođe, kod testiranja memetskog algoritma, dovoljno je izvršiti testiranje instance i12 za fiksirano k_{max}, pri čemu se mogu generisati rešenja za sve vrednosti Γ, što nije bio slučaj kod EA-LS algoritma. Slično važi i za ostale instance.

U tabeli 2.4 dato je poređenje rezultata EA-LS i predloženog MA za instancu i6_7_8, pri čemu su razmatrane tri različite vrednosti za parametar k_{max}: 10, 11 i 12. Na osnovu prikazanih rezultata se može zaključiti da su i EA-LS i MA dostigli optimalna rešenja prethodno dobijena CPLEX-om. U poslednjoj koloni tabele se može videti da se sa povećanjem Γ povećava i vrednost funkcije cilja optimalnog rešenja. Najveći procenat povećanja te vrednosti je prisutan za $\Gamma \geq 32$. Za $k_{max} = 10$ povećanje iznosi 38.043%, za $k_{max} = 11$ iznosi 38.371%, a za $k_{max} = 12$ iznosi 39.012%. Prosečno normalizovano vreme izvršavanja za instancu i6_7_8, po svim vrednostima parametara k_{max} i Γ, EA-LS algoritma iznosilo je 1.9 s, dok je prosečno vreme memetskog algoritma iznosilo 2.1 s.

U tabeli 2.5 prikazani su rezultati poređenja za instancu i1_2_3_4, pri čemu su razmatrane tri različite vrednosti za parametar k_{max}: 21, 22 i 23. Oba algoritma, EA-LS i MA, dostigla su prethodno dobijena optimalna rešenja.

Na osnovu vrednosti iz poslednje kolone tabele se može uočiti da se sa povećanjem Γ povećava i vrednost funkcije cilja optimalnog rešenja, a najveći procenat povećanja je prisutan za $\Gamma \geq 62$. Tada je za sve $k_{max} \in \{21, 22, 23\}$ on jednak 45.372. Prosečno normalizovano vreme izvršavanja po svim vrednostima parametara k_{max} i Γ za instancu i1_2_3_4 EA-LS algoritma iznosilo je 0.9 s, dok je odgovarajuće prosečno vreme memetskog algoritma iznosilo 5.1 s.

U tabelama 2.6 i 2.7 prikazani su rezultati za instancu i_all. Razmotreno je pet različitih vrednosti za parametar k_{max}: 4, 24, 36, 48 i 60. Kako u ovom slučaju, zbog dimenzije problema, CPLEX nije bio u mogućnosti da dostigne optimalno rešenje u roku od 3 h, za oba algoritma prikazana su najbolja dostignuta rešenja, gde je sa *najb* naznačeno ukoliko je dostignuto rešenje ujedno i najbolje poznato. Za $k_{max} = 4$, $k_{max} = 36$, $k_{max} = 48$ i $k_{max} = 60$, oba algoritma su u svim slučajevima dostigla najbolja poznata rešenja. Međutim, za $k_{max} = 24$, memetski algoritam je popravio najbolje poznate vrednosti rešenja po svim vrednostima $\Gamma \in \{0, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165\}$. Za $\Gamma = 0$, algoritam EA-LS je dostigao vrednost rešenja 76.75, dok je memetski algoritam dostigao vrednost 66.25. U tabeli 2.6 se mogu videti dobijena poboljšanja za ostale slučajeve gde je $\Gamma > 0$. Sa povećanjem parametra Γ povećava se i vrednost funkcije cilja najboljeg poznatog rešenja, a najveći procenat povećanja je uočen za $\Gamma \geq 165$. Tada je, za $k_{max} = 4$, on jednak 10.985, za $k_{max} = 24$ iznosi 47.395, a za $k_{max} \in \{36, 48, 60\}$ iznosi 48.354. Prosečno normalizovano vreme izvršavanja po svim vrednostima parametara k_{max} i Γ za instancu i_all EA-LS algoritma iznosilo je 70.18 s, dok je odgovarajuće prosečno vreme memetskog algoritma iznosilo 22.88 s.

Dobijeni rezultati pokazuju da je memetski algoritam za sve instance dostigao visoko kvalitetna rešenja. Za slučajeve kada je $k_{max} = 24$ i $\Gamma \in \{0, 15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165\}$, MA je popravio dosadašnja najbolja poznata rešenja instance i_all, a u ostalim slučajevima su dostignuta rešenja optimalna ili najbolja poznata. Primenom teoreme 2.3 je omogućeno da se ukupno vreme izvršavanja algoritma skrati, pri čemu je dovoljno posmatrati samo determinističku varijantu problema, a iz nje se jednostavno mogu odrediti rešenja za slučajeve kada je $\Gamma > 0$. To s jedne strane omogućava da implementacija CPLEX rešavača ne uzima u obzir robusnu varijantu, već determinističku, koja sadrži manje promenljivih i ograničenja, a s druge da je memetski algoritam dovoljno testirati samo za slučaj kada je $\Gamma = 0$. Što se tiče poređenja prosečnog vremena za koje je dostignuto najbolje rešenje, može se uočiti da se za instance manjih dimenzija (i12, i6_7_8 i i1_2_3_4) EA-LS algoritam pokazao efikasnijim, dok se predloženi MA pokazao efikasnijim za najveću instancu i_all. U proseku, za sve instance manjih dimenzija (po svim

Tabela 2.3: Poređenja rezultata za instancu i12

k_{max}	Γ	Rešenje	$t_{CPLEX}[s]$	$t^{napr}_{CPLEX}[s]$	EA-LS$_{najb}$	MA$_{najb}$	$t_{\text{EA-LS}}[s]$	$t_{\text{MA}}[s]$	Pov[%]
3	0	27.5500	0.37	0.37	*opt*	*opt*	0.009	1.070	0.000
3	5	29.6708	0.76	0.37	*opt*	*opt*	0.006	1.070	7.698
3	10	30.2250	0.91	0.37	*opt*	*opt*	0.008	1.070	9.710
3	15	30.5458	0.48	0.37	*opt*	*opt*	0.009	1.070	10.874
3	17	30.5583	0.50	0.37	*opt*	*opt*	0.006	1.070	10.919
4	0	19.7500	0.34	0.34	*opt*	*opt*	0.042	0.680	0.000
4	5	21.8708	0.49	0.34	*opt*	*opt*	0.036	0.680	10.738
4	10	22.4250	0.51	0.34	*opt*	*opt*	0.041	0.680	13.544
4	15	22.7458	0.68	0.34	*opt*	*opt*	0.041	0.680	15.169
4	17	22.7583	0.77	0.34	*opt*	*opt*	0.036	0.680	15.232
5	0	19.7500	0.16	0.16	*opt*	*opt*	0.012	1.020	0.000
5	5	21.8708	0.44	0.16	*opt*	*opt*	0.011	1.020	10.738
5	10	22.4250	0.42	0.16	*opt*	*opt*	0.011	1.020	13.544
5	15	22.7458	0.50	0.16	*opt*	*opt*	0.011	1.020	15.169
5	17	22.7583	0.59	0.16	*opt*	*opt*	0.011	1.020	15.232
Prosek		24.51	0.53	0.29	*opt*	*opt*	0.02 (0.03*)	0.92	9.90

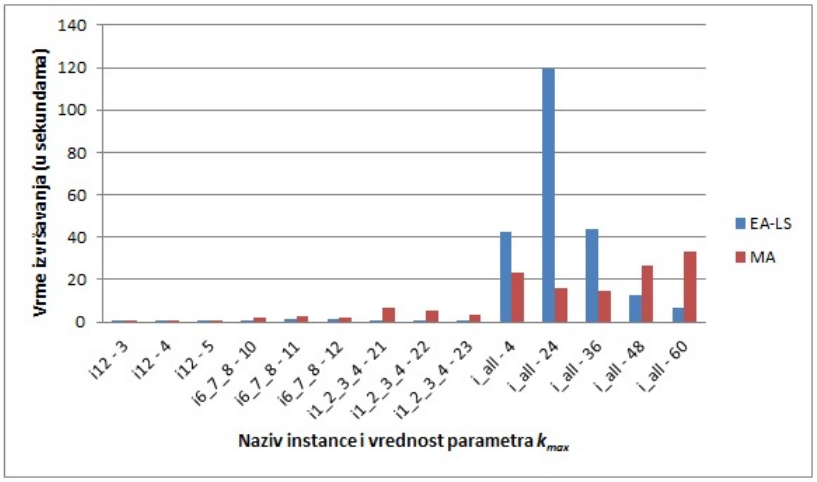
vrednostima k_{max} i Γ), prosečno normalizovano vreme izvršavanja EA-LS algoritma iznosilo je 0.97 s, a prosečno vreme memetskog 2.71 s. Za instance većih dimenzija prosečno normalizovano vreme izvršavanja EA-LS algoritma je bilo 70.18 s, a prosečno vreme memetskog 22.88 s. Može se zaključiti da se EA-LS pokazao nešto efikasnijim nad instancama manjih dimenzija, a MA nad većim. Na slici 2.1 su prikazana prosečna vremena dostizanja najboljeg rešenja EA-LS i MA metode nad svim razmatranim instancama. Pritom za EA-LS algoritam je prikazano odgovarajuće prosečno vreme izvršavanja iz [144].

2.5.4 Rezultati za $|T| > 1$

U ovom pododeljku će biti prikazani rezultati predloženog MA za instance za koje je $|T| = 2$ i $|T| = 3$. Za svaku instancu su posmatrane različite vrednosti parametra k_{max}. Kako se, prema prethodnom, rešenje za robusni slučaj može jednostavno dobiti od rešenja za koje je $\Gamma = 0$, u tabeli 2.8 su prikazani rezultati samo za deterministčku varijantu problema. Vrednosti k_{max} su za svaku instancu dobijene množenjem broja perioda i vrednosti k_{max} za slučajeve sa jednim periodom. Tako, na primer, za instancu i12 vrednosti k_{max} u tabeli 2.3 su iznosile 3, 4 i 5, pa za instancu i12_t2 iznose 6, 8 i 10, a za instancu i12_t3 9, 12 i 15. Oznake zagljavlja tabele se poklapaju sa

Tabela 2.4: Poređenja rezultata za instancu i6_7_8

k_{max}	Γ	Rešenje	$t_{CPLEX}[s]$	$t^{napr}_{CPLEX}[s]$	EA-LS$_{najb}$	MA$_{najb}$	$t_{\text{EA-LS}}[s]$	$t_{\text{MA}}[s]$	Pov[%]
10	0	29.1667	7.1	7.1	*opt*	*opt*	0.875	1.988	0.000
10	10	37.0583	7.1	20.9	*opt*	*opt*	0.777	1.988	27.057
10	20	39.3625	7.1	55.6	*opt*	*opt*	1.031	1.988	34.957
10	25	39.9333	7.1	68.0	*opt*	*opt*	1.074	1.988	36.914
10	32	40.2625	7.1	91.2	*opt*	*opt*	1.000	1.988	38.043
11	0	29.1667	10.3	10.3	*opt*	*opt*	0.831	2.514	0.000
11	10	37.0583	10.3	50.6	*opt*	*opt*	1.325	2.514	27.291
11	20	39.3625	10.3	107.0	*opt*	*opt*	1.141	2.514	35.259
11	25	39.9333	10.3	164.5	*opt*	*opt*	1.127	2.514	37.233
11	32	40.2625	10.3	140.0	*opt*	*opt*	1.464	2.514	38.371
12	0	28.4417	7.0	7.0	*opt*	*opt*	1.577	1.809	0.000
12	10	36.3333	7.0	21.2	*opt*	*opt*	1.168	1.809	27.746
12	20	38.6375	7.0	314.1	*opt*	*opt*	1.670	1.809	35.848
12	25	39.2083	7.0	271.1	*opt*	*opt*	1.770	1.809	37.855
12	32	39.5375	7.0	334.0	*opt*	*opt*	1.558	1.809	39.012
Prosek		36.91	8.1	110.8	*opt*	*opt*	1.23 (1.9*)	2.10	27.71



Slika 2.1: Prosečno vreme dostizanja najboljeg rešenja EA-LS i MA

Tabela 2.5: Poređenja rezultata za instancu i1_2_3_4

k_{max}	Γ	Rešenje	EA-LS$_{najb}$	MA$_{najb}$	$t_{\text{EA-LS}}[s]$	$t_{\text{MA}}[s]$	Pov[%]
21	0	63.7500	*opt*	*opt*	0.764	6.923	0.000
21	15	80.8750	*opt*	*opt*	0.725	6.923	26.863
21	30	87.3667	*opt*	*opt*	0.634	6.923	37.046
21	45	90.8417	*opt*	*opt*	0.673	6.923	42.497
21	55	92.2708	*opt*	*opt*	0.736	6.923	44.738
21	62	92.6750	*opt*	*opt*	0.862	6.923	45.372
22	0	63.7500	*opt*	*opt*	0.502	5.144	0.000
22	15	80.8750	*opt*	*opt*	0.564	5.144	26.863
22	30	87.3667	*opt*	*opt*	0.474	5.144	37.046
22	45	90.8417	*opt*	*opt*	0.624	5.144	42.497
22	55	92.2708	*opt*	*opt*	0.584	5.144	44.738
22	62	92.6750	*opt*	*opt*	0.625	5.144	45.372
23	0	63.7500	*opt*	*opt*	0.502	3.238	0.000
23	15	80.8750	*opt*	*opt*	0.434	3.238	26.863
23	30	87.3667	*opt*	*opt*	0.764	3.238	37.046
23	45	90.8417	*opt*	*opt*	0.622	3.238	42.497
23	55	92.2708	*opt*	*opt*	0.389	3.238	44.738
23	62	92.6750	*opt*	*opt*	0.712	3.238	45.372
Prosek		84.63	*opt*	*opt*	0.62 (0.97*)	5.10	32.75

Tabela 2.6: Poređenja rezultata za instancu i_all i $k_{max} \in \{4, 24\}$

k_{max}	Γ	Rešenje	EA-LS$_{najb}$	MA$_{najb}$	$t_{\text{EA-LS}}[s]$	$t_{\text{MA}}[s]$	Pov[%]
4	0	483.6917	*najb*	*najb*	48.6	23.4	0.00
4	15	504.2542	*najb*	*najb*	52.7	23.4	4.07
4	30	516.1583	*najb*	*najb*	35.3	23.4	6.29
4	45	523.9958	*najb*	*najb*	49.1	23.4	7.69
4	60	528.9667	*najb*	*najb*	36.2	23.4	8.55
4	75	532.7625	*najb*	*najb*	41.7	23.4	9.21
4	90	535.8667	*najb*	*najb*	28.7	23.4	9.73
4	105	538.3542	*najb*	*najb*	48.1	23.4	10.15
4	120	540.4208	*najb*	*najb*	46.9	23.4	10.49
4	135	541.9000	*najb*	*najb*	52.2	23.4	10.74
4	150	542.8958	*najb*	*najb*	34.5	23.4	10.90
4	165	543.3792	*najb*	*najb*	34.9	23.4	10.98
24	0	66.2500	76.7500	*najb*	92.3	16.0	0.00
24	15	86.8125	98.9653	*najb*	78.7	16.0	23.68
24	30	98.7167	109.8166	*najb*	85.0	16.0	32.88
24	45	106.5542	118.1625	*najb*	77.4	16.0	37.82
24	60	111.5250	122.5306	*najb*	83.9	16.0	40.59
24	75	115.3208	127.0700	*najb*	85.3	16.0	42.55
24	90	118.4250	128.8611	*najb*	81.9	16.0	44.05
24	105	120.9125	131.7458	*najb*	99.1	16.0	45.20
24	120	122.9792	134.4950	*najb*	82.6	16.0	46.12
24	135	124.4583	136.0944	*najb*	80.5	16.0	46.76
24	150	125.4542	137.6958	*najb*	509.5	16.0	47.19
24	165	125.9375	137.9986	*najb*	76.5	16.0	47.39
Prosek		318.9997	324.7013	318.9997	80.90 (126.04*)	19.7	23.04

Tabela 2.7: Poređenja rezultata za instancu i_all i $k_{max} \in \{36, 48, 60\}$

k_{max}	Γ	Rešenje	EA-LS$_{najb}$	MA$_{najb}$	$t_{\text{EA-LS}}[s]$	$t_{\text{MA}}[s]$	Pov[%]
36	0	63.7500	*najb*	*najb*	53.7	14.9	0.00
36	15	84.3125	*najb*	*najb*	34.6	14.9	24.38
36	30	96.2167	*najb*	*najb*	43.8	14.9	33.74
36	45	104.0542	*najb*	*najb*	45.4	14.9	38.73
36	60	109.0250	*najb*	*najb*	37.0	14.9	41.52
36	75	112.8208	*najb*	*najb*	44.8	14.9	43.49
36	90	115.9250	*najb*	*najb*	44.4	14.9	45.00
36	105	118.4125	*najb*	*najb*	37.3	14.9	46.16
36	120	120.4792	*najb*	*najb*	48.6	14.9	47.08
36	135	121.9583	*najb*	*najb*	39.2	14.9	47.72
36	150	122.9542	*najb*	*najb*	51.0	14.9	48.15
36	165	123.4375	*najb*	*najb*	46.0	14.9	48.35
48	0	63.7500	*najb*	*najb*	15.0	26.6	0.00
48	15	84.3125	*najb*	*najb*	13.1	26.6	24.38
48	30	96.2167	*najb*	*najb*	11.6	26.6	33.74
48	45	104.0542	*najb*	*najb*	13.7	26.6	38.73
48	60	109.0250	*najb*	*najb*	11.8	26.6	41.52
48	75	112.8208	*najb*	*najb*	13.1	26.6	43.49
48	90	115.9250	*najb*	*najb*	13.0	26.6	45.00
48	105	118.4125	*najb*	*najb*	12.7	26.6	46.16
48	120	120.4792	*najb*	*najb*	13.7	26.6	47.08
48	135	121.9583	*najb*	*najb*	12.4	26.6	47.72
48	150	122.9542	*najb*	*najb*	11.3	26.6	48.15
48	165	123.4375	*najb*	*najb*	12.8	26.6	48.35
60	0	63.7500	*najb*	*najb*	8.2	33.2	0.00
60	15	84.3125	*najb*	*najb*	7.2	33.2	24.38
60	30	96.2167	*najb*	*najb*	5.4	33.2	33.74
60	45	104.0542	*najb*	*najb*	7.2	33.2	38.73
60	60	109.0250	*najb*	*najb*	7.0	33.2	41.52
60	75	112.8208	*najb*	*najb*	6.1	33.2	43.49
60	90	115.9250	*najb*	*najb*	5.2	33.2	45.00
60	105	118.4125	*najb*	*najb*	5.5	33.2	46.16
60	120	120.4792	*najb*	*najb*	6.8	33.2	47.08
60	135	121.9583	*najb*	*najb*	6.2	33.2	47.72
60	150	122.9542	*najb*	*najb*	7.2	33.2	48.15
60	165	123.4375	*najb*	*najb*	5.6	33.2	48.35
Prosek		107.7788	*najb*	*najb*	21.0 (32.8*)	24.9	38.69

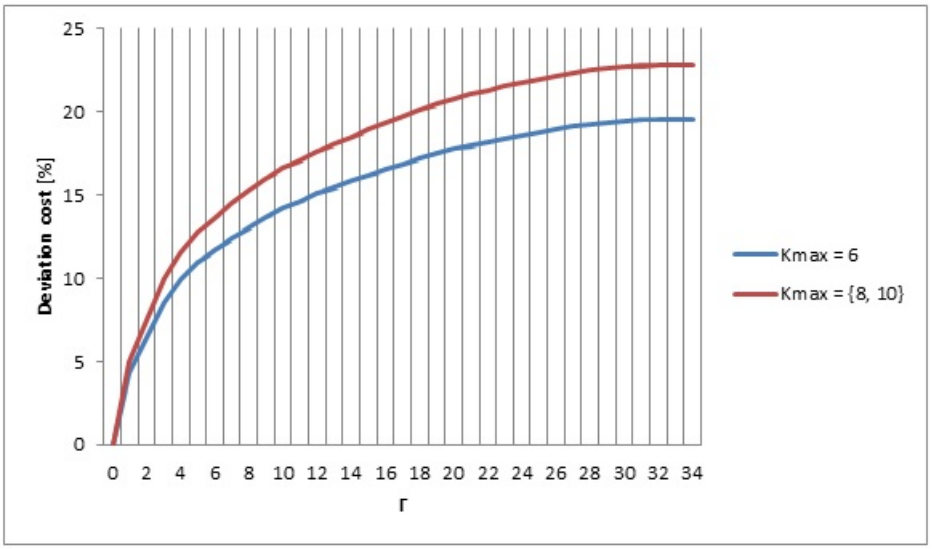
prethodnim, s tim što je sa $t_{\text{CPX}}[s]$ označeno vreme za koje je CPLEX dostigao optimalno rešenje. U slučajevima gde CPLEX nije dostigao optimalno rešenje u roku od 3 h, u odgovarajuće polje upisana je crta [−]. U koloni koja predstavlja vrednost dostignutog rešenja MA data je oznaka *opt* ukoliko je vrednost rešenja optimalna, a inače je prikazana vrednost odgovarajućeg najboljeg rešenja. Za sve instance manjih dimenzija dostignute su optimalne vrednosti rešenja. Prosečno vreme izvršavanja po svim instancama kod kojih je $|T| = 2$ iznosilo je 31.871 s, a za instance kod kojih je $|T| = 3$ bilo je 74.902 s, što dokazuje efikasnost predloženog memetskog algoritma u odnosu na CPLEX rešavač.

Na slici 2.2 prikazan je grafik povećanja vrednosti funkcije cilja optimalnog rešenja za instancu i12_t2 u zavisnosti od parametra Γ. Funkcija je rastuća za $0 \leq \Gamma \leq 34$. Za $\Gamma \geq 34$, vrednost rešenja ostaje nepromenjena. Na grafiku su prikazane vrednosti povećanja u procentima za tri slučaja: $k_{max} = 6$, $k_{max} = 8$ i $k_{max} = 10$. Za $k_{max} = 6$, najveći procenat povećanja iznosi 19.537 i dostignut je za $\Gamma \geq 34$. Dobijene vrednosti se poklapaju za $k_{max} = 8$ i $k_{max} = 10$, pa je najveći procenat povećanja vrednosti rešenja u ta dva slučaja jednak i iznosi 22.848. Plava linija na grafiku razmatra vrednosti za $k_{max} = 6$, a crvena $k_{max} = 8$ i $k_{max} = 10$.

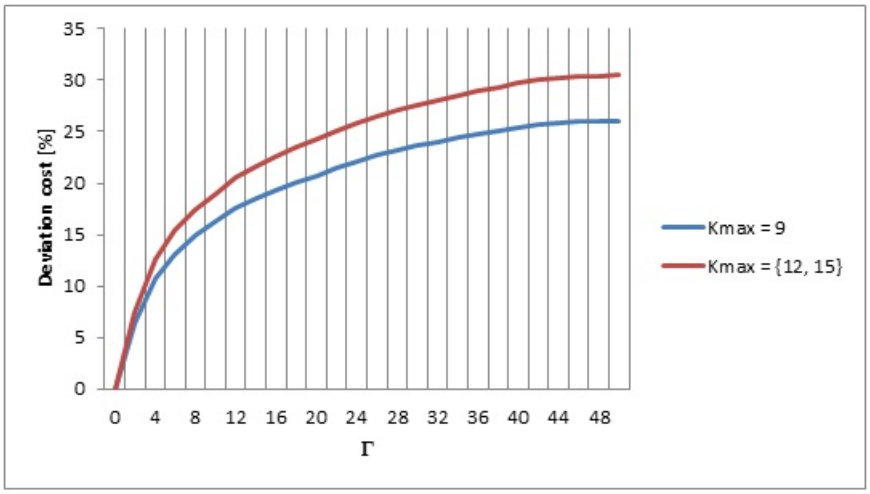
Slično, na slici 2.3 prikazan je grafik povećanja vrednosti funkcije cilja optimalnog rešenja za instancu i12_t3. Funkcija je rastuća za $0 \leq \Gamma \leq 51$. Za $\Gamma \geq 51$, vrednost rešenja ostaje nepromeljena. Na grafiku su prikazane vrednosti povećanja u procentima za tri slučaja: $k_{max} = 9$, $k_{max} = 12$ i $k_{max} = 15$. Za $k_{max} = 9$, najveći procenat povećanja iznosi 26.049 i dostignut je za $\Gamma \geq 51$. Dobijene vrednosti se poklapaju za $k_{max} = 12$ i $k_{max} = 15$, pa je najveći procenat povećanja vrednosti rešenja u ta dva slučaja jednak i iznosi 30.464.

Tabela 2.8: Rezultati predloženog MA za instance sa više perioda

Instanca	k_{max}	Rešenje	$t_{\mathrm{CPX}}[s]$	MA_{najb}	$t_{\mathrm{MA}}[s]$
i12_t2	6	15.3981	6.1	*opt*	2.9
i12_t2	8	13.1667	3.6	*opt*	3.1
i12_t2	10	13.1667	1.7	*opt*	1.8
i6_7_8_t2	20	18.5000	308.5	*opt*	1.3
i6_7_8_t2	22	18.5000	180.4	*opt*	3.4
i6_7_8_t2	24	18.5000	133.0	*opt*	3.4
i1_2_3_4_t2	42	42.5000	1872.2	*opt*	9.7
i1_2_3_4_t2	44	42.5000	3426.2	*opt*	14.9
i1_2_3_4_t2	46	42.5000	2696.2	*opt*	15.3
i_all_t2	8	258.0741	–	*najb*	32.7
i_all_t2	48	42.5000	–	*najb*	73.6
i_all_t2	72	42.5000	–	*najb*	83.9
i_all_t2	96	42.5000	–	*najb*	96.0
i_all_t2	120	42.5000	–	*najb*	103.5
i12_t3	9	11.5486	78.6	*opt*	2.3
i12_t3	12	9.8750	9.8	*opt*	5.0
i12_t3	15	9.8750	6.3	*opt*	4.5
i6_7_8_t3	30	13.8750	1229.5	*opt*	5.5
i6_7_8_t3	33	13.8750	1289.3	*opt*	5.5
i6_7_8_t3	36	13.8750	952.6	*opt*	4.8
i1_2_3_4_t3	63	31.8750	–	*najb*	44.5
i1_2_3_4_t3	66	31.8750	–	*najb*	49.0
i1_2_3_4_t3	69	31.8750	–	*najb*	69.1
i_all_t3	12	165.4549	–	*najb*	100.2
i_all_t3	72	31.8750	–	*najb*	159.1
i_all_t3	108	31.8750	–	*najb*	204.5
i_all_t3	144	31.8750	–	*najb*	206.7
i_all_t3	180	31.8750	–	*najb*	187.2
Prosek		39.7968	–	*najb*	53.3



Slika 2.2: Povećanje vrednosti funkcija cilja optimalnog rešenja u zavisnosti od Γ za instancu i12_t2



Slika 2.3: Povećanje vrednosti funkcija cilja optimalnog rešenja u zavisnosti od Γ za instancu i12_t3

3 Dinamički lokacijski problem maksimalnog pokrivanja sa više poluprečnika

Lokacijski problem maksimalnog pokrivanja (Maximal covering location problem – MCLP) je jedan od najviše rešavanih NP-teških lokacijskih problema, a prvi put je uveden u [27]. MCLP podrazumeva zadati skup lokacija, pri čemu je svakoj lokaciji dodeljen realan parametar, koji predstavlja veličinu njene populacije. Lokacija se smatra pokrivenom ukoliko postoji bar jedna uspostavljena lokacija koja je na rastojanju ne većem od unapred fiksiranog maksimalnog poluprečnika pokrivanja. Broj lokacija koje treba uspostaviti je unapred definisan. Cilj problema je odrediti koje lokacije treba uspostaviti, kako bi vrednost ukupne populacije pokrivenih lokacija bio maksimalan. U opštem slučaju, skupovi potencijalnih lokacija koje je potrebno uspostaviti i lokacija koje je potrebno pokriti mogu biti i disjunktni. U nastavku će se lokacije koje je potrebno uspostaviti nazivati resursima, a lokacije koje je potrebno pokriti gradovima.

Neka je I skup gradova (lokacije kojima je pridružena težina populacije), a J skup potencijalnih lokacija koje je potrebno uspostaviti (skup resursa). Za sve $i \in I$ i $j \in J$ definiše se parametar d_{ij}, koji predstavlja rastojanje između grada i i resursa j. Neka je s maksimalni poluprečnik pokrivanja, a p broj resursa koje je potrebno uspostaviti. Svakom gradu i dodeljen je nenegativan parametar a_i, koji predstavlja veličinu njegove populacije, odnosno broj korisnika u gradu i koje treba opslužiti. Grad $i \in I$ se može pokriti resursom uspostavljenim na lokaciji $j \in J$ ukoliko rastojanje između i i j nije veće od poluprečnika s. Neka je $N_i = \{j \in J : d_{ij} \leq s\}$, $i \in I$ skup resursa koje mogu pokriti grad i. Binarna promenljiva x_j, $j \in J$ je definisana sa

$$x_j = \begin{cases} 1, & \text{ako je resurs na lokaciji } j \text{ uspostavljen,} \\ 0, & \text{inače,} \end{cases}$$

a binarna promenljiva y_i, $i \in I$ sa

$$y_i = \begin{cases} 1, & \text{ako je grad } i \text{ pokriven,} \\ 0, & \text{inače.} \end{cases}$$

Koristeći prethodnu notaciju, lokacijski problem maksimalnog pokrivanja se može formulisati na sledeći način:

$$\max \sum_{i \in I} a_i y_i \tag{3.1}$$

$$y_i \leq \sum_{j \in N_i} x_j \quad \forall i \in I, \tag{3.2}$$

$$\sum_{j \in J} x_j = p, \tag{3.3}$$

$$y_i \in \{0, 1\} \quad \forall i \in I, \tag{3.4}$$

$$x_j \in \{0, 1\} \quad \forall j \in J. \tag{3.5}$$

Izraz (3.1) predstavlja funkciju cilja, tj. odnosi se na maksimizaciju ukupne veličine pokrivene populacije. Ograničenjem (3.2) je određeno da se samo gradovi i, za koje postoji bar jedan uspostavljen resurs j za koji je $d_{ij} \leq s$, mogu smatrati pokrivenim. Broj uspostavljenih resursa je tačno p (uslov 3.3). Uslovi (3.4) i (3.5) se odnose na binarnu prirodu promenljivih y_i i x_j.

Počev od prvog rada vezanog za MCLP [27], u literaturi su razmatrane razne varijante ovog problema. U [159] se razmatra problem kod koga su svi parametri a_i, $i \in I$ jednaki. Predložena heuristika se zasniva na metodi najbržeg spusta. U [29] se razmatra MCLP u ravni, pri čemu je pokazano da se problem može rešiti pomoću skupa tačaka koje predstavaljaju preseke krugova opisanih oko lokacija. Verovatnosna verzija modela predložena je u [133], gde je svakom resursu dodeljena vrednost, koja predstavlja pouzdanost njegovog uspostavljanja. Uopšteni problem maksimalnog pokrivanja (Generalized maximal covering location problem – GMCLP) predložen je u [10]. Kod GMCLP, uvodi se skup koncentričnih krugova poluprečnika $r_1, r_2, \ldots, r_k$, takvih da je $r_1 < r_2 < \cdots < r_k$. Za l takvo da je $r_l \leq d_{ij} < r_{l+1}$, $1 \leq l < k$, uvodi se parameter $s_l \in \{0, 1\}$, gde je $s_l a_i$ deo populacije koji treba pokriti. Autori predlažu heuristiku zasnovanu na pohlepnom algoritmu koja rešava problem. Autori u [33] razmatraju uopštenje MCLP kod koga resursi mogu biti različitog tipa. Broj resursa koji se mogu nalaziti na određenoj lokaciji je ograničen. Za rešavanje problema, autori predlažu varijantu metode

promenljivih okolina. U [46, 152] se razmatra fazi lokacijski problem maksimalnog pokrivanja (fuzzy maximal covering location problem – FMCLP), gde je predložena heuristika zasnovana na optimizaciji rojem čestica za njegovo rešavanje. U [153] se razmatra lokacijski problem minimalnog pokrivanja, koji je rešavan pomoću fazi skupova. Ideja minimalnog lokacijskog problema pokrivanja je da se, uz iste uslove kao kod MCLP, izvrši izbor resursa koje treba uspostaviti tako da ukupna veličina populacije pokrivenih gradova bude minimalna. Imajući to u vidu, lokacijski problem minimalnog pokrivanja se može smatrati suprotnim problemom u odnosu na lokacijski problem maksimalnog pokrivanja.

Problem maksimalnog pokrivanja je dosta proučavan u literaturi upravo zahvaljujući svojoj širokoj primeni u praksi. Najčešća oblast primene MCLP je optimizacija sistema hitnih službi, na primer službe hitne pomoći, policijskih stanica ili vatrogasnih službi [39, 44, 131]. Broj stanica (jedinica) hitnih službi je ograničen unapred zadatom konstantom ili fiksiran na zadatu konstantu, imajući u vidu finansijska i druga ograničenja sistema hitnih službi. Samim tim, postoji realna mogućnost da svi zahtevi neće biti ispunjeni na adekvatan način, tj. u zadatom vremenu reagovanja.

U ovim slučajevima, cilj je odrediti optimalne lokacije na kojima treba uspostaviti jedinice hitne službe, tako da se maksimizuje broj korisnika do kojih služba može pravovremeno doći. Broj korisnika na jednoj lokaciji kojima treba pružiti uslugu je obično broj stanovnika u jednom delu grada, celom gradu ili nekom regionu. Međutim, u praksi se najčešće koriste statistički podaci za određivanje zahteva jedne lokacije, odnosno broja korisnika na jednoj lokaciji kojima treba pružiti pomoć.

U slučajevima kada statistički podaci ukazuju na to da veličina populacije može varirati tokom različitih vremenskih perioda, koristi se dinamička verzija problema (Dynamic maximal covering location problem – DMCLP).

Neka T predstavlja skup razmatranih vremenskih perioda. U odnosu na formulaciju MCLP, parametri a_i su zamenjeni parametrima a_{ti}, $t \in T$, $i \in I$, koji predstavljaju veličinu populacije grada i u periodu t. Binarne promenljive x_j i y_i se zamenjuju binarnim promenljivim x_{tj} i y_{ti}, $t \in T$, $j \in J$, $i \in I$, pri čemu je

$$x_{tj} = \begin{cases} 1, & \text{ako je resurs } j \text{ uspostavljen u trenutku } t, \\ 0, & \text{inače} \end{cases}$$

i

$$y_{ti} = \begin{cases} 1, & \text{ako se grad } i \text{ može pokriti u trenutku } t, \\ 0, & \text{inače.} \end{cases}$$

Dinamički lokacijski problem maksimalnog pokrivanja može se formulisati kao problem linearnog programiranja na sledeći način [162]:

$$\max \sum_{t\in T}\sum_{i\in I} a_{ti}y_{ti}, \tag{3.6}$$

uz ograničenja

$$y_{ti} \leq \sum_{j\in N_i} x_{tj} \quad \forall t \in T \quad \forall i \in I, \tag{3.7}$$

$$\sum_{t\in T}\sum_{j\in J} x_{tj} = p, \tag{3.8}$$

$$y_{ti} \in \{0,1\} \quad \forall t \in T \quad \forall i \in I, \tag{3.9}$$

$$x_{tj} \in \{0,1\} \quad \forall t \in T \quad \forall j \in J. \tag{3.10}$$

Uslovi (3.7) – (3.10) su analogni uslovima (3.2) – (3.5). Dodatno, autori predlažu algoritam koji je zasnovan na simuliranom kaljenju za rešavanje DMCLP. Budući da formulacija (3.6) – (3.10) sadrži $|T||I| + |T||J|$ promenljivih i $2|T||I| + |T||J| + 1$ ograničenje, CPLEX rešavač je u mogućnosti da za većinu instanci iz [162], za koje je $|T| \leq 5$, $|I| \leq 2500$ i $|J| \leq 200$, pronađe optimalno rešenje. Sa druge strane, algoritam simuliranog kaljenja ni u jednom slučaju da nije dostigao optimalno rešenje, a najveće odstupanje iznosilo je oko 1%. Ipak, imajući u vidu vreme izvršavanja, SA algoritam se pokazao efikasnijim od CPLEX rešavača.

Kod problema dinamičkog maksimalnog pokrivanja, poluprečnik pokrivanja r je fiksiran. Međutim, u praksi ta pretpostavka može dovesti do neželjenih situacija. Na primer, ukoliko se dva grada nalaze blizu, pri čemu je jedan na rastojanju nešto manjem od r u odnosu na uspostavljeni resurs, a drugi na nešto većem do r, populacija prvog će u potpunosti biti pokrivena, a drugog neće. U slučaju da je odgovarajući resurs uspostavljen, za neku malu vrednost pozitivnog parametra ε, grad na rastojanju $r - \varepsilon$ će biti pokriven, dok onaj na rastojanju $r + \varepsilon$ neće, iako se nalaze jedan blizu drugoga.

Ideja sa različitim poluprečnicima pokrivanja je korišćena u radu [10]. Autori uvode sistem od k koncentričnih krugova za svaku lokaciju grada, pri čemu se populacija svakog grada sa manjom ili većom efikasnošću može opslužiti od strane resursa koji su locirani unutar razlicitih krugova pokrivanja. Za razliku od [10], u ovoj disertaciji je predložena varijanta dinamičkog modela maksimalnog pokrivanja kod koga se izbor maksimalnog poluprečnika pokrivanja vrši imajući u vidu veličinu populacije u odgovarajućim vremenskim intervalima i pouzdanost pokrivanja od strane resursa.

U uopštenju DMCLP predloženom u disertaciji, umesto parametra maksimalnog rastojanja s, uvodi se niz dozvoljenih maksimalnih rastojanja s_k, $k \in K$, pri čemu je $s_1 < s_2 < \cdots < s_{|K|}$. Za svaki krug poluprečnika s_k sa centrom u resursu j, svi gradovi i za koje je $d_{ij} \leq s_k$ se mogu pokriti. Pritom se uvodi parametar b_k, $k \in K$ ($b_1 > b_2 > \cdots > b_{|K|}$) kojim se kontroliše mogućnost pokrivanja svih gradova unutar kruga sa poluprečnikom s_k. Sa porastom poluprečnika raste broj pokrivenih gradova, a vrednost parametra b_k opada, čime najveći poluprečnik ne mora uvek da bude nužno najbolji odabir. Potrebno je odrediti minimalnu vrednost skupa

$$\left\{ \max \sum_{t \in T} \sum_{i \in I} b_k a_{ti} y_{ti} : k \in K \right\}.$$

Svaki element skupa predstavlja rešenje dinamičkog lokacijskog problema maksimalnog pokrivanja sa maksimalnim poluprečnikom s_k.

Dinamički lokacijski problem maksimalnog pokrivanja je prvi put predložen u [162] i predstavlja uopštenje MCLP, kod koga se posmatra skup vremenskih intervala T, pri čemu veličina populacije može imati različite vrednosti u različitim vremenskim intervalima. Ukoliko se na početku vremenskog perioda uspostavi resurs na određenoj lokaciji, na kraju se može zatvoriti i uspostaviti novi. Pritom je maksimalni poluprečnik pokrivanja po svim vremenskim intervalima fiksiran.

U praksi je korisno razmatrati potencijal raspoloživih resursa imajući u vidu različite poluprečnike pokrivanja. Na primer, policijske jedinice smeštene na određenoj lokaciji nisu u mogućnosti da reaguju podjednako ukoliko je incident izazvan neposredno pored stanice ili nekoliko desetina kilometara dalje. Zbog toga je značajno uzeti u obzir različite vrednosti za poluprečnike pokrivanja i izabrati najpogodniji, u zavisnosti od konkretne situacije. Sličan princip se može primeniti i prilikom određivanja optimalnih lokacija za službu hitne pomoći. U praksi se može izabrati nešto veći poluprečnik i time vozila hitne pomoći mogu stići do većeg broja pacijenata. Naravno, treba imati u vidu da se tada mogu povećati ukupni troškovi ili ljudski resursi, ali se predloženim modelom nudi mogućnost što boljeg odabira.

Za razne vrste kompanija koje mogu pružati korisnicima usluge, uvođenje sistema poluprečnika može u praksi dovesti do povećanja profita. Na primer, pri dostavi robe, određene lokacije se mogu nalaziti jedna blizu druge, a ako je dalja lokacija na nešto većem rastojanju od poluprečnika pokrivanja, određena kompanija može izgubiti deo klijenata. Sa druge strane, ukoliko je pogodnije uzeti veći poluprečnik, određenoj kompaniji može trebati više goriva ili ljudskih resursa, ali dobija veći broj korisnika i time može povećati profit. Pri telekomonikacionim sistemima, može se razmotriti broj stanov-

nika u odgovarajućim mestima i na osnovu toga odrediti optimalan raspored prijemnika signala. Ukoliko se razmatraju različite vrednosti poluprečnika njihovog dometa, ukupan broj prijemnika se može smanjiti i time uštedeti pri njihovom uspostavljanju.

U odeljcima 3.1 i 3.2 biće prikazana matematička formulacija problema, kao i odgovarajuća robusna varijanta. U odeljku 3.3 biće prikazan algoritam koji rešava problem, tako što ga raščlani na odgovarajuće potprobleme, koje se uspešno rešavaju hibridizacijom memetskog algoritma i algoritma zasnovanog na linearnom programiranju. Konačno, u poslednjem delu biće prikazani dobijeni eksperimentalni rezultati.

3.1 Matematička formulacija

Kod razmatranog problema, umesto skupova N_i, $i \in I$, uvode se skupovi $N_{ik} = \{j \in J : d_{ij} \leq s_k\}$, $i \in I$, $k \in K$. Za svaki grad $i \in I$, N_{ik} predstavlja skup resursa $j \in J$ za koje rastojanje d_{ij} nije veće od maksimalnog poluprečnika pokrivanja s_k. Svaki potencijalni resurs $j \in J$ predstavlja centar za tačno $|K|$ krugova poluprečnika s_k, $k \in K$. Ako je sa M_{jk} označen skup gradova koji se mogu pokriti krugom poluprečnika s_k gde je j-ti resurs uspostavljen, važi da je $M_{j1} \subseteq M_{j2} \subseteq \cdots \subseteq M_{j|K|}$. Cilj problema je izabrati p od $|J|$ resursa za uspostavljanje, kako bi se maksimizovala ukupna veličina populacije pokrivenih gradova.

Neka su binarne promenljive x_{tjk}, $t \in T$, $j \in J$, $k \in K$ i y_{tik}, $t \in T$, $i \in I$, $k \in K$ definisane sa

$$x_{tjk} = \begin{cases} 1, & \text{ako je resurs } j \text{ uspostavljen u trenutku } t \text{ za poluprečnik } s_k, \\ 0, & \text{inače} \end{cases}$$

i

$$y_{tik} = \begin{cases} 1, & \text{ako se grad } i \text{ može pokriti u trenutku } t \text{ za poluprečnik } s_k, \\ 0, & \text{inače.} \end{cases}$$

Predloženi problem se može predstaviti kao problem mešovitog celobrojnog linearnog programiranja na sledeći način:

$$\max z \tag{3.11}$$

uz ograničenja

$$z \leq \sum_{t \in T} \sum_{i \in I} b_k a_{ti} y_{tik} \quad \forall k \in K, \tag{3.12}$$

$$y_{tik} \leq \sum_{j \in N_{ik}} x_{tjk} \quad \forall t \in T \quad \forall i \in I \quad \forall k \in K, \tag{3.13}$$

$$\sum_{t \in T} \sum_{j \in J} x_{tjk} = p \quad \forall k \in K, \tag{3.14}$$

$$y_{tik} \in \{0, 1\} \quad \forall t \in T \quad \forall i \in I \quad \forall k \in K, \tag{3.15}$$

$$x_{tjk} \in \{0, 1\} \quad \forall t \in T \quad \forall j \in J \quad \forall k \in K, \tag{3.16}$$

$$z \geq 0. \tag{3.17}$$

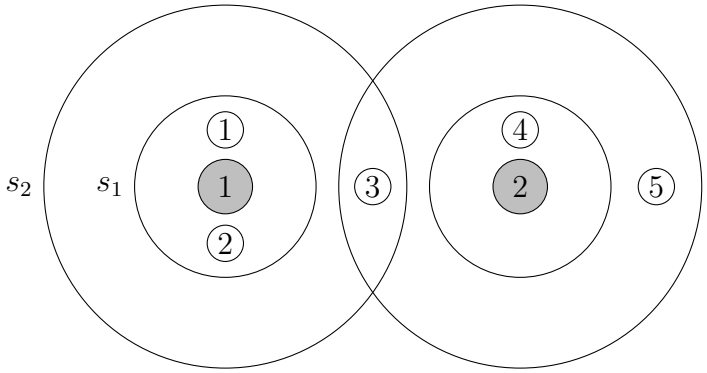
Izraz (3.11) zajedno sa uslovom (3.12) određuje vrednost funkcije cilja. Iz (3.11) i (3.12) sledi da je promenljiva z jednaka

$$\min \left\{ \max \sum_{t \in T} \sum_{i \in I} b_k a_{ti} y_{tik} : k \in K \right\},$$

gde je svaka suma iz skupa vrednost optimalnog rešenja za dinamički lokacijski problem maksimalnog pokrivanja sa poluprečnikom s_k. Ograničenjem (3.13) je omogućeno da se samo gradovi i za koje postoji bar jedna uspostavljen resurs j u trenutku t za koju je $d_{ij} \leq s_k$ mogu smatrati pokrivenim. Uslov (3.14) ograničava broj uspostavljenih resursa sumirano po svim periodima iz T za fiksirano $k \in K$ na tačno p. Ograničenja (3.15) i (3.16) se odnose binarnu prirodu promenljivih y_{tik} i x_{tjk}, a ograničenje (3.17) na nenegativnost promenljive z.

Primer 3.1 Neka je $|T| = 1$, $|I| = 5$, $|J| = 2$, $|K| = 2$, $p = 1$, $b_1 = 0.75$, $b_2 = 0.4$ i neka su sve veličine populacije u gradovima jednake 1. Resursom 1 mogu se pokriti prva dva grada za maksimalni poluprečnik s_1, odnosno prva tri grada za poluprečnik s_2. Četvrti grad se može pokriti resursom 2 za poluprečnik s_1, dok se za najveći poluprečnik drugim resursom mogu pokriti poslednja tri grada (slika 3.1). Razmatraju se sledeća dva scenarija:

- Maksimalni poluprečnik pokrivanja je s_1. Kako je $p = 1$, samo jedan resurs može biti uspostavljen. Ako je uspostavljen prvi resurs, vrednost optimalnog rešenja je $2b_1 = 1.5$, a pokriveni su gradovi 1 i 2. Ako je drugi resurs uspostavljen, optimalna vrednost rešenja iznosi 0.75 i tada se jedino četvrti grad može pokriti. Dakle, u ovom slučaju vrednost optimalnog rešenja je 1.5.



Slika 3.1: Pozicije gradova i resursa iz primera 1

- Maksimalni poluprečnik pokrivanja je s_2. Ako je prvi resurs uspostavljen, gradovi 1, 2 i 3 se mogu pokriti. Ako je uspostavljen drugi resurs, poslednja tri grada su pokrivena. Optimalno rešenje iznosi $3b_2 = 1.2$, nezavisno od izbora resursa koji je uspostavljen.

U optimalnom rešenju, uspostavljen je prvi resurs, a gradovi 1 i 2 su pokriveni, pri čemu je maksimalni poluprečnik pokrivanja s_1. Vrednost optimalnog rešenja iznosi 1.5.

3.2 Robusna formulacija i složenost

U većini praktičnih situacija, veličina populacije u gradovima može varirati u određenom vremenskom periodu. Na primer, broj incidenata se može smanjiti ili povećati na mesečnom ili godišnjem nivou, ili intenzitet prirodnih katastrofa za određenu regiju može biti povećan usled globalnog zagrevanja ili drugog prirodnog fenomena. Zbog toga je neophodno razmatrati slučaj kada parametri a_{ti} nisu fiksirani. U predloženom modelu ti parametri variraju i predstavljaju promenljive koje uzimaju vrednosti iz intervala $[a_{ti}, a_{ti} + \hat{a}_{ti}]$, gde je $\hat{a}_{ti} \geq 0$, $t \in T$ i $i \in I$.

Neka je $G_t = \{i \in I : \hat{a}_{ti} > 0\}$ skup gradova čija veličina populacije (broj korisnika, broj incidenata koje treba pokriti itd.) varira u posmatranom vremenskom periodu $t \in T$. Dalje, neka celobrojni parametri Γ_t, $0 \leq \Gamma_t \leq |G_t|$ predstavljaju maksimalan broj gradova, kod kojih veličina populacije varira. Parametri Γ_t kontrolišu nivo robusnosti funkcije cilja. Ako je $\Gamma_t = 0$, njihov uticaj se u potpunosti ignoriše, dok se za $\Gamma_t = |G_t|$ u obzir uzimaju sva variranja veličine populacije u vremenskom periodu t.

Za robusnu varijantu modela, definisan je novi skup promenljivih u_{tik}, $t \in T$, $i \in I$, $k \in K$ sa

$$u_{tik} = \begin{cases} 1, & \text{ako potažnja grada } i \text{ varira u periodu } t \text{ za poluprečnik } s_k, \\ 0, & \text{inače.} \end{cases}$$

Robusna matematička formulacija je definisana sa (3.11), (3.13) – (3.17) i sledećim uslovima:

$$z \leq \sum_{t \in T} \sum_{i \in K} b_k (a_{ti} y_{tik} + \hat{a}_{ti} u_{tik}) \quad \forall k \in K, \tag{3.18}$$

$$u_{tik} \leq y_{tik} \quad \forall t \in T \quad \forall i \in I \quad \forall k \in K, \tag{3.19}$$

$$\sum_{i \in I} u_{tik} \leq \Gamma_t \quad \forall t \in T \quad \forall k \in K, \tag{3.20}$$

$$u_{tik} \in \{0, 1\} \quad \forall t \in T \quad \forall i \in I \quad \forall k \in K. \tag{3.21}$$

Uslov (3.18) je analogan uslovu (3.12), s tim što se ovde u obzir uzimaju i odgovarajuće vrednosti $\hat{a}_{ti}$. Uslovom (3.19) obezbeđeno je da veličina populacije može varirati samo u pokrivenim gradovima. Broj gradova koji varira u trenutku t za svako fiksirano $k \in K$ ne sme biti veći od Γ_t, što je određeno ograničenjem (3.20). Uslov (3.21) se odnosi na binarnu prirodu promenljivih u_{tik}.

Predloženi problem je uopštenje dinamičkog lokacijskog problema maksimalnog pokrivanja. Sa druge strane, DMCLP predložen u [162] je NP-težak kao uopštenje MCLP za koji je pokazano da je NP-težak u [102]. Jasno je i da je odgovarajuća robusna varijanta problema takođe NP-teška, budući da svaki NP-težak problem ostaje takav u njegovoj robusnoj varijanti [14].

3.3 Predloženi hibridni algoritam

U ovom odeljku je opisana hibridizacija memetskog algoritma (MA) i tehnike zasnovane na linearnom programiranju, u oznaci MA-LP. Predložena hibridizacija rastavlja početni problem na $|K|$ potproblema, od kojih je svaki ekvivalentan dinamičkom lokacijskom problemu maksimalnog pokrivanja. Za svako $k \in K$, ulazni parametri za potproblem koji se rešava su I, J, T, maksimalni poluprečnik pokrivanja s_k, broj raspoloživih resursa p i veličine populacije u gradovima $b_k a_{ti}$, $t \in T$, $i \in I$. U robusnoj varijanti problema veličine populacije variraju u intervalu $[b_k a_{ti}, b_k a_{ti} + b_k \hat{a}_{ti}]$. Ako DMCLP_k, $k \in K$ predstavlja optimalnu vrednost rešenja DMCLP sa poluprečnikom pokrivanja s_k, vrednost optimalnog rešenja polaznog problema

iznosi $\min_{k \in K} \text{DMCLP}_k$. Svaki DMCL potproblem je dalje podeljen na potprobleme ekvivalentne lokacijskom problemu maksimalnog pokrivanja. Svaki od MCL potproblema se rešava hibridizacijom MA pristupa i tehnike zasnovane na linearnom programiranju. Dobijena rešenja se zatim kombinuju kako bi se konstruisalo rešenje za DMCLP. Osnovna struktura MA-LP je prikazana algoritmom 3.1. U narednim odeljcima će svaki korak iz algoritma detaljno biti objašnjen.

Algoritam 3.1 Osnovna struktura MA-LP pristupa

```
1: sol = ∞
2: for all k ∈ K do
3:    for all t ∈ T do
4:       for j ← p' − Δ to p' + Δ do
5:          Primena memetskog algoritma na MCLP(k, t, j)
6:          Primena tehnike zasnovane na linearnom programiranju na
            MCLP(k, t, j)
7:       end for
8:    end for
9:    Kombinovanje rešenja MCLP(k, t, j) za određivanje DMCLP_k
10:   sol = min{sol, DMCLP_k}
11: end for
12: return sol
```

3.3.1 Rešavanje potproblema maksimalnog pokrivanja

Neka $\text{MCLP}(k, t, j)$ označava potproblem ekvivalentan lokacijskom problemu maksimalnog pokrivanja, pri čemu je $k \in K$, $t \in T$ i $1 \leq j \leq |J|$. Za MCLP se svaki vremenski period $t \in T$ razmatra nezavisno od ostalih. Ulazni parametri za potproblem su skup gradova i potencijalnih resursa I i J, veličine populacije $b_k a_{ti}$, $i \in I$, maksimalni poluprečnik pokrivanja s_k i broj raspoloživih resursa j. U robusnom slučaju veličine populacije variraju u intervalu $[b_k a_{ti}, b_k a_{ti} + b_k \hat{a}_{ti}]$, $i \in I$.

Osnovna struktura MA za rešavanje lokacijskog problema maksimalnog pokrivanja je prikazana algoritmom 3.2. Početne vrednosti bitova rešenja se generišu proizvoljno. Jedino ograničenje prilikom inicijalizacije je da broj jediničnih bitova mora biti jednak j. Kriterijum zaustavljanja predstavlja maksimalan broj iteracija. Vrednost ovog parametra će biti eksperimentalno određena u pododeljku 3.4.2. Memetskim algoritmom se rešava jedino deterministička varijanta problema, dok se pri primeni tehnike zasnovanoj na

linearnom programiranju razmatraju deterministička i robusna varijanta problema, pri čemu se kao polazne vrednosti koriste dobijena rešenja iz MA.

Algoritam 3.2 Osnovna struktura MA za rešavanje MCLP(k, t, j)

```
1: Učitavanje podataka
2: while nije ispunjen kriterijum zaustavljanja do
3:    Izdvajanje skupa elitnih rešenja N_e iz skupa N_r koji se direktno
      prosleđuju u narednu generaciju
4:    Primena PSO algoritma na rešenja iz skupa N_r \ N_e
5:    for all r ∈ N_r do
6:       Primena lokalne pretrage na rešenje r
7:    end for
8: end while
9: Ispis rešenja
```

3.3.2 Reprezantacija rešenja

Populacija predloženog MA predstavlja skup N_r, koji se sastoji od $|N_r|$ binarno kodiranih rešenja. Svako rešenje sadrži tačno $|J|$ bitova. Ako se na l-tom bitu nalazi vrednost 1, resurs l, $1 \leq l \leq |J|$ se smatra uspostavljenim. Inače, ako l-ti bit ima vrednost 0, resurs na lokaciji l nije uspostavljen. Suma svih bitova iznosi tačno j. Na primer, ako je $J = \{1, \ldots, 9\}$ i $j = 3$, jedno dopustivo rešenje je oblika 100010010, što znači da su resursi na lokacijama 1, 5 i 8 uspostavljeni.

Početne vrednosti bitova rešenja se generišu proizvoljno. Jedino ograničenje prilikom inicijalizacije je da broj jediničnih bitova mora biti jednak j. Svaka iteracija memetskog algoritma se sastoji od primene optimizacije rojem čestica (PSO) i heuristike zasnovane na lokalnom pretraživanju (LS). Za razliku od evolutivnog dela MA, lokalna pretraga se primenjuje na sva rešenja iz skupa N_r. Predložena LS heuristika se primenjuje po 10 puta u svakoj iteraciji. Kriterijum zaustavljanja predstavlja maksimalan broj iteracija. Vrednost ovog parametra će biti eksperimentalno određena u pododeljku 3.4.2. Memetskim algoritmom se rešava jedino deterministička varijanta problema, dok se pri primeni tehnike zasnovanoj na linearnom programiranju razmatraju deterministička i robusna varijanta problema, pri čemu se kao polazne vrednosti koriste dobijena rešenja iz MA.

3.3.3 Računanje funkcije cilja

Neka je sa Q_l označen skup gradova koji se mogu pokriti uspostavljanjem resursa na lokaciji l. Kako bi se povećala efikasnost funkcije cilja, skupovi Q_l su inicijalno određeni.

Neka celobrojni parametri δ_i, $i \in I$ predstavljaju broj uspostavljenih resursa na rastojanju manjem ili jednakom s_k od grada i. Inicijalno su sve vrednosti δ_i jednake 0. Niz uspostavljenih resursa se najpre generiše na osnovu vrednosti koda rešenja. Za svaki uspostavljen resurs l, u obzir se uzimaju svi gradovi i iz Q_l, pri čemu se ažurira vrednost parametra δ_i. Ažuriranjem se njegova vrednost uvećava za 1. Ako je $\delta_i = 1$, to znači da postoji bar jedan resurs čijim je uspostavljanjem pokriven grad i u periodu t. Tada se vrednost funkcije cilja uvećava za $b_k a_{ti}$.

3.3.4 Evolutivni deo memetskog algoritma

Evolutivni deo MA je zasnovan na optimizaciji rojem čestica (PSO), pri čemu čestice odgovaraju rešenjima iz skupa N_r. Svakom rešenju $i \in N_r$ dodeljuje se binarni vektor pozicije $\mathbf{x}_i$ i realni vektor brzine $\mathbf{v}_i$, koji su dužine $|J|$. Binarni vektor $\mathbf{x}_i$ odgovara kodu rešenja i. Preciznije, ako je l-ti bit vektora $\mathbf{x}_i$ jednak 1, resurs na lokaciji l, $1 \leq l \leq |J|$ je uspostavljen. Ako je l-ti bit vektora $\mathbf{x}_i$ jednak 0, resurs na lokaciji l nije uspostavljen. Algoritam PSO, kao evolutivni deo MA, se primenjuje samo na ne-elitna rešenja iz skupa N_r, dok se kvalitetnija rešenja direktno prosleđuju u narednu generaciju.

U svakoj iteraciji l-ti bit vektora $\mathbf{x}_i$ se računa na sledeći način:

$$\mathbf{x}_{i,l} \leftarrow \begin{cases} 1, & \text{ako je } r < (1 + e^{-\mathbf{v}_{i,l}})^{-1}, \\ 0, & \text{inače.} \end{cases}$$

Parametar r se bira uniformnom raspodelom iz intervala $(0, 1)$. Nakon računanja vektora pozicije, broj jediničnih bitova može biti različit od j (j je broj resursa koji treba da bude uspostavljen). Ako je suma svih bitova s_j veća od j, na slučajan način bira se $s_j - j$ bitova sa vrednošću 1 i ovi bitovi se invertuju. Ukoliko je suma svih bitova manja od j, tada se $j - s_j$ bitova čija je vrednost 0 proizvoljno biraju i invertuju. Ukoliko je vrednost funkcije cilja novodobijenog rešenja i veća od trenutno najbolje, ažurira se vektor $\mathbf{p}_i$ i odgovarajuća vrednost funkcije cilja. Ako je vrednost funkcije cilja za vektor $\mathbf{p}_i$ veća od vrednosti rešenja za vektor $\mathbf{g}$, analogno se ažurira i vrednost vektora $\mathbf{g}$.

Nakon ažuriranja vektora pozicije, ažurira se odgovarajući vektor brzine. Prilikom ažuriranja vektora brzine $\mathbf{v}_i$ rešenja i, najpre se određuje vektor promene brzine sa

$$\Delta \mathbf{v}_{i,l} \leftarrow r_p(\mathbf{p}_i - \mathbf{x}_i) + r_g(\mathbf{g} - \mathbf{x}_i),$$

gde je $r_p, r_g \in U(0,1)$ i $1 \leq l \leq |J|$. Zatim se vektor brzine $\mathbf{v}_i$ se određuje na sledeći način:

$$\mathbf{v}_i \leftarrow \begin{cases} 1, & \text{ako je } \mathbf{v}_{i,l} + \Delta \mathbf{v}_{i,l} > 1, \\ 0, & \text{ako je } \mathbf{v}_{i,l} + \Delta \mathbf{v}_{i,l} < 0, \\ \mathbf{v}_{i,l} + \Delta \mathbf{v}_{i,l}, & \text{inače.} \end{cases}$$

3.3.5 Primena lokalne pretrage

Lokalno pretraživanje se primenjuje po 10 puta u svakoj iteraciji MA na sva rešenja iz skupa N_r. U svakom koraku se na proizvoljan način biraju dva bita na pozicijama l_1 i l_2, $1 \leq l_1, l_2 \leq |J|$, koji imaju različitu vrednost koda (jedan od resursa na lokacijama l_1 i l_2 je uspostavljen, a drugi nije), a zatim se izvrši njihova zamena. Ukoliko dobijeno rešenje ima bolju vrednost funkcije cilja od najboljeg, novo rešenje se postavlja za najbolje. Inače, vrši se restauracija zamene bitova. Ovaj proces se ponavlja dok god postoji poboljšanje vrednosti rešenja najbolje jedinke.

Prelazak u novo rešenje koje pripada okolini tekućeg rešenja realizuje se inverzijom dva slučajno izabrana bita, sa različitim vrednostima. Bit koji je imao vrednost 0 dobija vrednost 1, što odgovara otvaranju nove lokacije snabdevača. Bit koji je imao vrednost 1 dobija vrednost 0, što odgovara zatvaranju lokacije snabdevača koja je bila otvorena. Umesto ponovnog računanja funkcije cilja ovako dobijenog novog rešenja, može se iskoristiti vrednost funkcije cilja tekućeg rešenja, gde se u obzir samo uzimaju promene koje se dešavaju pri invertovanju izabranog bita. Ukoliko bit na poziciji l menja vrednost iz 0 u 1, dovoljno je samo uvećati vrednosti parametara δ_i za sve gradove i iz skupa Q_l. Ako tada grad i postaje pokriven, vrednost $b_k a_{ti}$ se dodaje na trenutnu vrednost funkcije cilja. Slično, ukoliko bit na poziciji l menja vrednost iz 1 u 0, dovoljno je samo umanjiti vrednosti parametara δ_i gradova i iz skupa Q_l. Ako tada neki grad i više nije pokriven, vrednost $b_k a_{ti}$ se oduzima od trenutne vrednosti funkcije cilja. Složenost obe opisane inverzije je $O(|Q_l|)$, gde je Q_l skup gradova koji se mogu pokriti uspostavljanjem resursa na lokaciji l.

3.3.6 Primena tehnike zasnovane na linearnom programiranju

Nakon završene primene memetskog algoritma, koristi se pristup zasnovan na linearnom programiranju u cilju dobijanja optimalnog rešenja za MCLP. Pritom se najbolje rešenje MA koristi kao početno rešenje CPLEX rešavača. Tokom preliminarnog testiranja je utvrđeno da su ova početna rešenja optimalna ili blizu optimalnih, što redukuje ukupno vreme izvršavanja CPLEX-a.

Sva početna rešenja CPLEX rešavača su dobijena primenom MA na determinističku varijantu problema. Kod determinističke varijante problema se u ovoj fazi koristi formulacija (3.1) – (3.5) lokacijskog problema maksimalnog pokrivanja, pri čemu a_i odgovara parametru a_{ti}, a maksimalna vrednost odstojanja je s_k. Ulaz u CPLEX rešavač predstavlja rešenje kod koga su određene vrednosti promenljivih x_j i y_i, što se jednostavno može odrediti na osnovu koda svakog rešenja.

Linearni model robusne varijante MCLP može se formulisati na sledeći način:

$$\max \sum_{i \in I} (a_{ti} y_i + \hat{a}_{ti} u_i) \tag{3.22}$$

uz ograničenja (3.2) – (3.5) i

$$u_i \leq y_i \quad \forall i \in I, \tag{3.23}$$

$$\sum_{i \in I} u_i \leq \Gamma_t, \tag{3.24}$$

$$u_i \in \{0, 1\}. \tag{3.25}$$

Funkcija cilja (3.22) maksimizuje ukupnu veličinu populacije pokrivenih gradova. Ograničenjima (3.23) i (3.24) se određuje gornja granica za broj gradova u kojima varira vrednost populacije. Vrednost populacije može varirati samo u gradovima koji su pokriveni. Uslov (3.25) se odnosi na binarnu prirodu promenljivih u_i. Vrednosti promenljivih x_j i y_i su čitaju iz koda najboljeg rešenja dobijenog primenom MA, a sve vrednosti promenljivih u_i su inicijalno jednake nuli. Tokom preliminarnog testiranja je uočeno da su rešenja konstruisana na ovaj način u većini slučajeva bliska optimalnom.

3.3.7 Računanje vrednosti DMCLP$_k$

Neka M_{ktj} predstavlja rešenje za MCLP(k, t, j), $k \in K$, $t \in T$, $j \in P_\Delta$. Neka je $p' = \lfloor p/|T| \rfloor$ i neka za celobrojni parametar Δ i skup P_Δ važi

$0 \leq \Delta \leq p$ i $P_\Delta = \{p' - \Delta, \ldots, p' + \Delta\}$. Potproblem MCLP$(k, t, j)$ je rešen hibridizacijom MA i tehnike zasnovane na linearnom programiranju. Dobijena rešenja se kombinuju u cilju dobijanja rešenja DMCLP na sledeći način:

$$\text{DMCLP}_k = \max \left\{ \sum_{t \in T} M_{ktj_t} : \sum_{t \in T} j_t = p \ \wedge \ j_t \in P_\Delta \right\}, \quad k \in K. \tag{3.26}$$

Preciznije, vrednost rešenja DMCLP$_k$ je jednaka maksimumu među svim sumama $M_{k1j_1} + M_{k2j_2} + \cdots + M_{k|T|j_{|T|}}$ za koje je $j_1 + \cdots + j_{|T|} = p$, gde j_t predstavlja broj raspoloživih resursa za vremenski period t. Ako je $P_\Delta \supseteq \{0, 1, \ldots, p\}$, vrednost rešenja DMCLP$_k$ je jednaka optimalnom. Međutim, cilj je odrediti minimalnu vrednost parametra Δ, tako da je sačuvan kvalitet rešenja. Na taj način se povećava efikasnost algoritma.

Neka su binarne promenljive z_{tj}, $t \in T$, $j \in P_\Delta$ definisane sa

$$z_{tj} = \begin{cases} 1, & \text{ako je } j \text{ resursa uspostavljeno u vremenskom periodu } t, \\ 0, & \text{inače.} \end{cases}$$

Za računanje vrednosti (3.26), koristi se sledeći linearni model:

$$\max \sum_{t \in T} \sum_{j \in P_\Delta} M_{ktj} z_{tj}, \tag{3.27}$$

$$\sum_{j \in P_\Delta} z_{tj} = 1 \quad \forall t \in T, \tag{3.28}$$

$$\sum_{t \in T} \sum_{j \in P_\Delta} j z_{tj} = p, \tag{3.29}$$

$$z_{tj} \in \{0, 1\} \quad \forall t \in T \quad \forall j \in P_\Delta. \tag{3.30}$$

Izraz (3.27) predstavlja funkciju cilja i služi za računanje (3.26). Prema (3.28), broj raspoloživih resursa tokom jednog perioda je jedinstven. Preciznije, za fiksirano $t \in T$, nikoje dve promenljive z_{tj_1} i z_{tj_2}, $j_1 \neq j_2$ ne mogu istovremeno imati vrednost 1. Broj uspostavljenih resursa po svim periodima $t \in T$ je p (3.29). Ograničenja (3.30) se odnose na binarnu prirodu promenljivih z_{tj}. Broj promenljivih i broj ograničenja iz formulacije (3.27) – (3.30) je jednak $O(|T|\Delta)$, što dopusta korišćenje CPLEX rešavača za efikasno dobijanje rešenja linearnog modela.

3.4 Eksperimentalni rezultati

U ovom odeljku će biti predstaljveni rezultati hibridnog algoritma MA. Implementacija algoritma je izvršena u programskom jeziku C++, a rešavač CPLEX 12.1 je takođe inkorporiran u C++. Za svaku instancu, MA-LP algoritam je pokretan po 15 puta. Sva testiranja su izvršena pod Windows 7 operativnim sistemom sa procesorom Intel i5-2430M od 2.4 GHz i RAM memorijom od 8 GB.

3.4.1 Test instance

Budući da instance iz [162] nisu javno dostupne, za testiranje predloženog algoritma su generisane dve grupe instanci. Prva grupa od 45 instanci je generisana na način opisan u [162]. Prva grupa instanci se sastoji od tri podgrupe: instance manjih dimenzija (DMCLP-M), za koje je $|I| = 1800$ i $|J| = 100$, instance srednjih dimenzija (DMCLP-S), za koje je $|I| = 2000$ i $|J| = 150$ i instance velikih dimenzija (DMCLP-V), za koje je $|I| = 2500$ i $|J| = 200$.

Svaka podgrupa sastoji se od 15 instanci. Za sve DMCLP instance skup perioda T ima tačno 5 elemenata. Svaka podgrupa sadrži sve parove vrednosti p i s za koje je $p \in \{65, 70, 75, 80, 85\}$ i $s \in \{4, 4.5, 5\}$. Vrednosti x i y koordinata gradova generisane su uniformno iz intervala $[0, 30]$, dok su vrednosti a_{ti}, $t \in T$, $i \in I$ takođe birane proizvoljno iz intervala $[0, 100]$ (videti [162]).

Druga grupa instanci je generisana za robusnu varijantu problema. U ovom slučaju su generisane tri podrgupe instanci. Prva podgrupa, u oznaci DMCLP-R1, generisana je na sličan način kao što je to urađeno za DMCLP instance. Vrednosti parametara su $|T| = 5$, $|I| = 1800$, $|J| = 100$, $|K| = 3$ i $p = 85$, a poluprečnici pokrivanja iznose $s_1 = 4$, $s_2 = 4.5$ i $s_3 = 5$. Uzeto je i $b_1 = 1.05$, $b_2 = 1$, $b_3 = 0.95$, $x, y \in U[0, 30]$, $a_{ti} \in U[0, 100]$, $\hat{a}_{ti} \in U[0.05a_{ti}, 0.1a_{ti}]$, $t \in T$, $i \in I$. Veličine populacije u gradovima su uvećane za 5 do 10 procenata. Parametar Γ uzima svih 19 vrednosti iz skupa $\{100\gamma : 0 \leq \gamma \leq 18 \wedge \gamma \in \mathbb{N}_0\}$.

Za preostale dve podgrupe, u oznaci DMCLP-R2 i DMCLP-R3, skupovi gradova i resursa se poklapaju, pri čemu je $|I| = |J| = 165$. Rastojanja d_{ij} su određena na osnovu geografskih koordinata odgovarajućih mesta, a parametri a_{ti} su bazirani na prosečnom broju ozbiljnih krivičnih dela dobijenih na osnovu statističkih podataka u prethodnih nekoliko godina. Vrednosti ostalih parametara iznose: $|K| = 3$, $(s_1, s_2, s_3) = (25, 30, 35)$, $(b_1, b_2, b_3) = (1.05, 1, 0.95)$ i $p = 120$. Veličine populacije u gradovima su

uvećane za 5 do 10 procenata. Za podgrupu DMCLP-R2 je $|T| = 2$ i $0 \leq \Gamma \leq 330$, a za podgrupu DMCLP-R3 je $|T| = 3$ i $0 \leq \Gamma \leq 495$.

3.4.2 Podešavanje parametara MA-LP algoritma

U MA-LP algoritmu postoji nekoliko parametara čije je vrednosti neophodno preciznije odrediti kako bi algoritam konvergirao ka što kvalitetnijim rešenjima. Na osnovu preliminarnog testiranja pretpostavljeno je da sledeća četiri parametra MA-LP mogu imati uticaja na rešenje:

- $|N_r|$ – broj rešenja iz skupa N_r;
- $iter$ – maksimalni broj iteracija MA;
- p_{el} – procenat elitnih jedinki iz algoritma;
- Δ – veličina skupa P_Δ.

Za određivanje adekvatnih vrednosti prva tri parametra, korišćena je analiza varijanse [113]. Korišćeni su sledeći nivoi:

- parametar $|N_r|$ ima dva nivoa – 15 i 20;
- parametar $iter$ ima dva nivoa – 2000 i 5000;
- parametar p_{el} ima dva nivoa – 66.67% i 75%.

Ukupan broj kombinacija iznosi $2 \cdot 2 \cdot 2 = 8$. Po jedna reprezentativna instanca je uzeta iz skupova DMCLP-M, DMCLP-S i DMCLP-V, pri čemu je $p = 70$, $s = 5$ i $(I, J) \in \{(1800, 100), (2000, 150), (2500, 200)\}$.

Za svaku kombinaciju parametara memetski algoritam je puštan po 15 puta za svaku instancu. Zabeleženo je najbolje dostignuto MA rešenje za svaku instancu. Treba napomenuti da se pri testiranju samo uzima u obzir MA deo algoritma za MCLP, tj. tehnika zasnovana na linearnom programiranju je izostavljena, jer na nju razmatrani parametri nemaju uticaja.

Kritična vrednost za analizu varijansi iznosila je 0.05. Iz tabele 3.1 se može uočiti da za 95%-ni interval poverenja nijedan faktor nije imao značajnijeg uticaja na vrednost funkcije cilja. Parametri $|N_r|$ i p_{el} imaju najmanji uticaj, budući da su njihove p-vrednosti u tabeli uglavnom blizu 1. Broj iteracija algoritma ima najveći uticaj na vrednost funkcije cilja. U sva tri slučaja se najbolje vrednosti funkcije cilja dobijaju za $|N_r| = 20$ i $iter = 5000$. Vrednost $p_{el} = 75\%$ za prvu i treću instancu daje najbolje

Tabela 3.1: Rezultati primene analize varijanse na podešavanje parametara

$\|I\|$	$\|J\|$	Parametar	SK	SS	PK	F	p
1800	100	$\|N_r\|$	315 654.407	1	315 654.407	0.2048	0.9234
1800	100	$iter$	23 997 200.082	1	23 997 200.082	51.5840	0.1736
1800	100	p_{el}	1 039 501.127	1	1 039 501.127	0.6890	0.7714
2000	150	$\|N_r\|$	3 775 235.404	1	3 775 235.404	0.9613	0.7010
2000	150	$iter$	64 685 278.384	1	64 685 278.384	55.8269	0.1736
2000	150	p_{el}	17 490.600	1	17 490.600	0.0043	0.9983
2500	200	$\|N_r\|$	1 746 577.307	1	1 746 577.307	0.2068	0.9227
2500	200	$iter$	151 545 962.940	1	151 545 962.940	92.4973	0.1736
2500	200	p_{el}	2 379 258.482	1	2 379 258.482	0.2826	0.8963

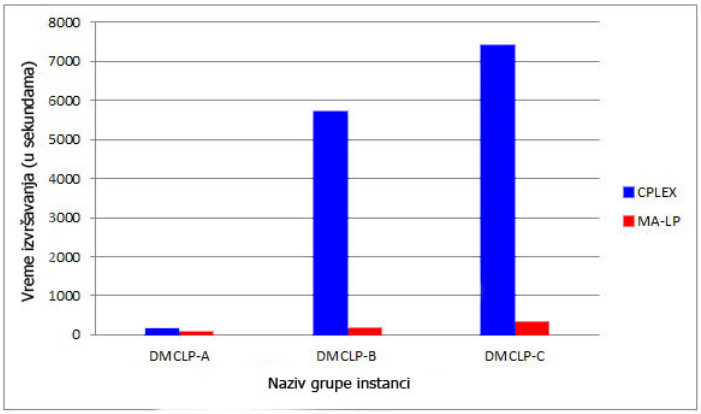
rešenje, a za drugu vrednost $p_{el} = 66.67\%$. Imajući u vidu dobijene rezultate, za vrednost parametra p_{el} uzeto je 75%.

Parametar Δ, koji je srazmeran veličini skupa P_Δ, ima važnu ulogu pri dobijanju kvalitetnih rešenja i efikasnosti algoritma. Ako je $P_\Delta \supseteq \{1, 2, \ldots, p\}$, algoritam će dostići optimalno rešenje za DMCLP, budući da su sva rešenja za MCLP optimalna. Međutim, potrebno je podesiti vrednost parametra Δ tako da se sačuva kvalitet rešenja, a da algoritam bude efikasan. Za izabrane vrednosti parametara $|N_r|$, $iter$ i p_{el}, testirane su četiri različite vrednosti parametra Δ: 1, 2, 5 i 10. Za $\Delta \in \{2, 5, 10\}$, algoritam uvek dostiže optimalno ili najbolje poznato rešenje. Kako se efikasnost algoritma povećava sa smanjivanjem vrednosti ovog parametra, u predloženoj implemantaciji uzeta je vrednost $\Delta = 2$.

3.4.3 Rezultati za DMCLP

Ako je skup K jednočlan, predloženi problem se svodi na dinamički lokacijski problem maksimalnog pokrivanja. U tabelama 3.2–3.4 su prikazani rezultati CPLEX-a i MA-LP za DMCLP instance. Značenja naziva kolona u tabelama su sledeća:

- s – vrednost maksimalnog poluprečnika za odgovarajuću instancu;
- p – maksimalan broj raspoloživih resursa u svim vremenskim intervalima;
- Rešenje – optimalno ili najbolje poznato rešenje;
- CPLEX – rešenje dobijeno pomoću CPLEX-a (u slučaju da CPLEX nije dao optimalno rešenje, navedena je donja granica dobijena CPLEX rešavačem);



Slika 3.2: Poređenje vremena izvršavanja rešavača CPLEX 12.1 i predloženog MA-LP za DMCLP instance

- MA-LP – najbolje rešenje dobijeno pomoću MA-LP algoritma.

Iz rezultata u tabelama 3.2 – 3.4 može se uočiti da CPLEX nije bio u mogućnosti da u roku od 3 sata (gornje vremensko ograničenje koje je korišćeno u [162]) dostigne optimalno rešenje za sve test instance. Preciznije, CPLEX je dostigao donju granicu za jednu instancu iz DMCLP-S grupe i tri instance iz DMCLP-V grupe. Sa druge strane, MA-LP je dostigao sva poznata optimalna rešenja. Za jednu DMCLP-S instancu i tri DMCLP-V instance, za koje optimalno rešenje nije poznato, popravljene su donje granice dobijene CPLEX rešavačem (videti tabele 3.3 i 3.4).

Poredeći vremena izvršavanja CPLEX-a i MA-LP, može se zaključiti da je MA-LP oko 2.3 puta brži za DMCLP-M instance, oko 38.01 put brži za DMCLP-S instance, odnosno oko 22.2 puta brži za DMCLP-V instance u odnosu na CPLEX rešavač (videti sliku 3.2). Može se zaključiti da se MA-LP pokazao superiornijim od CPLEX rešavača, u pogledu kvaliteta dobijenih rešenja i brzine izvršavanja. Kako za fiksiranu vrednost parametra Δ, MA-LP pristup uvek dostiže isto rešenje, vrednost standardne devijacije za sve instance iznosi 0. Za male vrednosti Δ se može desiti da se optimalno rešenje nikad i ne dostigne, dok se za $\Delta \geq n_0$, gde je $n_0 \in \mathbb{N}_0$, ne poboljšava kvalitet rešenja, ali se sa druge strane smanjuje efikasnost algoritma, tj. povećava se vreme izvršavanja, kako se vrednost Δ povećava.

Prema rezultatima predstavljenim u [162], algoritam predložen u tom radu nije bio u mogućnosti da dostigne optimalno rešenje ni za jednu od testiranih instanci. Odsupanja svih rešenja dobijenih SA pristupom su do 1% u odnosu na optimalna rešenja za testirane instance. Na osnovu rezultata iz tabela 3–5 iz [162], prosečno odstupanje SA iznosilo je 0.55% za instance

Tabela 3.2: Rezultati za DMCLP-M instance

s	p	Rešenje	CPLEX	t_{CPLEX}[s]	MA-LP	$t_{\text{MA-LP}}$[s]
4.0	65	322760.8	*opt*	196.47	*opt*	46.02
4.0	70	335986.1	*opt*	145.38	*opt*	42.52
4.0	75	355999.9	*opt*	176.59	*opt*	58.39
4.0	80	369873.2	*opt*	159.14	*opt*	81.06
4.0	85	381437.8	*opt*	144.43	*opt*	86.47
4.5	65	371732.5	*opt*	166.87	*opt*	72.84
4.5	70	390677.4	*opt*	147.23	*opt*	90.71
4.5	75	402872.0	*opt*	181.49	*opt*	83.35
4.5	80	419897.1	*opt*	199.67	*opt*	76.02
4.5	85	420556.5	*opt*	170.71	*opt*	63.93
5.0	65	419413.7	*opt*	152.16	*opt*	78.66
5.0	70	427120.5	*opt*	144.30	*opt*	70.48
5.0	75	434030.2	*opt*	184.96	*opt*	74.35
5.0	80	441649.3	*opt*	153.41	*opt*	91.42
5.0	85	447793.6	*opt*	222.55	*opt*	90.46
Prosek		396120.0	*opt*	169.69	*opt*	73.78

Tabela 3.3: Rezultati za DMCLP-S instance

s	p	Rešenje	CPLEX	t_{CPLEX}[s]	MA-LP	$t_{\text{MA-LP}}$[s]
4.0	65	365831.8	*opt*	5294.20	*opt*	109.17
4.0	70	387118.5	*opt*	5811.98	*opt*	82.63
4.0	75	406595.8	*opt*	4731.91	*opt*	111.88
4.0	80	415483.3	*opt*	6298.30	*opt*	184.49
4.0	85	425809.4	*opt*	5297.07	*opt*	120.78
4.5	65	426030.8	*opt*	5114.21	*opt*	129.78
4.5	70	434135.7	*opt*	5122.63	*opt*	101.13
4.5	75	458713.3	*opt*	4287.73	*opt*	160.52
4.5	80	468110.4	*opt*	6067.87	*opt*	136.17
4.5	85	**469415.5**	469414.7	10800.00	*najb*	195.11
5.0	65	464067.1	*opt*	5816.02	*opt*	242.41
5.0	70	470459.3	*opt*	5294.75	*opt*	152.90
5.0	75	485528.7	*opt*	4582.49	*opt*	204.69
5.0	80	494346.2	*opt*	4945.19	*opt*	144.74
5.0	85	497421.2	*opt*	5992.57	*opt*	171.88
Prosek		444604.5	444604.4	5697.13	444604.5	149.88

Tabela 3.4: Rezultati za DMCLP-V instance

s	p	Rešenje	CPLEX	t_{CPLEX}[s]	MA-LP	$t_{\text{MA-LP}}$[s]
4.0	65	460983.5	*opt*	6789.84	*opt*	115.28
4.0	70	483078.7	*opt*	7238.02	*opt*	129.31
4.0	75	507397.2	*opt*	5593.97	*opt*	186.45
4.0	80	530731.8	*opt*	6298.51	*opt*	283.13
4.0	85	545124.2	*opt*	5972.47	*opt*	241.79
4.5	65	531247.8	*opt*	5156.59	*opt*	772.26
4.5	70	**549621.8**	549573.0	10800.00	*najb*	266.52
4.5	75	564916.5	*opt*	6256.52	*opt*	365.45
4.5	80	585465.9	*opt*	5619.58	*opt*	446.89
4.5	85	**598528.6**	598473.6	10800.00	*najb*	421.92
5.0	65	590262.8	*opt*	6373.86	*opt*	279.99
5.0	70	598285.7	*opt*	5463.11	*opt*	525.51
5.0	75	**609544.0**	609495.5	10800.00	*najb*	376.41
5.0	80	612671.7	*opt*	7510.06	*opt*	253.11
5.0	85	623633.0	*opt*	10378.30	*opt*	338.27
Prosek		559432.9	554139.8	7403.39	559432.9	333.49

manjih dimenzija, 0.73% za instance srednjih i 0.64% za instance većih dimenzija. Sa druge strane, za sve DMCLP instance, koje su generisane na isti način kao u [162], MA-LP je dostigao optimalna ili, u slučaju gde optimalna rešenja nisu poznata, popravio donje granice dobijene CPLEX rešavačem. Algoritam simuliranog kaljenja je bio oko 24.74 puta brži od CPLEX rešavača korišćenog u [162], dok je za DMCLP instance MA-LP bio oko 20.84 puta brži od CPLEX 12.1 rešavača. Kako u [162] nisu navedene performanse računara nad kojim je testiran algoritam simuliranog kaljenja, u ovom slučaju nije moguće izvršiti normalizaciju vremena SA i njegovo direktno poređenje sa MA-LP.

3.4.4 Rezultati za robusni slučaj

U tabelama 3.5–3.7 su prikazani rezultati izvršavanja CPLEX-a i MA-LP za robusne instance DMCLP-R1, DMCLP-R2 i DMCLP-R3, respektivno. Značenja naziva kolona u tabelama su sledeća:

- Γ – vrednost parametra Γ_t (za sve $t \in T$ je uzeto $\Gamma_t = \Gamma$);
- CPLEX – optimalno rešenje dobijeno pomoću CPLEX-a;
- t_{CPLEX}[s] – vreme izvršavanja CPLEX-a u sekundama;

- MA-LP – rešenje dobijeno pomoću MA-LP algoritma;
- t_{MA}[s] – prosečno vreme izvršavanja algoritma u sekundama;
- Pov[%] – procentualna vrednost povećanja funkcije cilja u odnosu na slučaj $\Gamma = 0$.

Na osnovu dobijenih rezultata za instance DMCLP-R1, DMCLP-R2 i DMCLP-R3, može se uočiti da preloženi MA-LP algoritam na svim primerima dostiže optimalna rešenja. Za instance manjih dimenzija, DMCLP-R2 i DMCLP-R3, CPLEX dostiže optimalna rešenja za kratko vreme izvršavanja (za manje od 0.3 sekunde). Za instancu DMCLP-R2, prosečno vreme izvršavanja CPLEX-a po svim Γ iznosilo je 0.094, a MA-LP 1.567 sekundi. Za instancu DMCLP-R3, odgovarajuća prosečna vremena izvršavanja su 0.217 i 6.849 sekundi. Prednost MA-LP pristupa u odnosu na CPLEX rešavač dolazi do izražaja na instancama većih dimezija. Na osnovu rezultata dobijenih za instancu DMCLP-R1 prikazanih u tabeli 3.5, vidi se da je MA-LP superiorniji u odnosu na CPLEX u pogledu vremena izvršavanja. Za $\Gamma = 1600$, CPLEX-u je bilo potrebno oko sat vremena da dobije optimalno rešenje, dok je MA-LP sva optimalna rešenja dostigao za manje od 170 sekundi. Prosečno vreme izvršavanja CPLEX-a po svim Γ iznosilo je 1142.715, a MA-LP algoritma 135.933 sekunde, što je oko 8.4 puta brže. Slično kao kod rezultata testiranja MA-LP za determinističu varijantu problema, vrednost standardne devijacije iznosi 0 za sve instance.

Poslednja kolona predstavlja povećanje vrednosti funkcije cilje u procentima u zavisnosti od Γ. Jasno je da sa povećanjem vrednosti Γ dolazi do povećanja vrednosti funkcije cilja. Analizirajući rezultate predstavljene u koloni Pov[%], može se zaključiti da se sa povećanjem Γ vrednost funkcije cilja povećava do određene vrednosti, nakon koje ostaje nepromenjena, iako vrednost parametra Γ i dalje raste. Na primer, vrednost funkcije cilja u tabeli 3.5 se ne menja za $\Gamma \geq 1600$. Iz toga se može zaključiti da je maksimalan broj gradova po svim periodima koji se u optimalnom rešenju može pokriti manji ili jednak 1600, a veći od 1500, budući da se za $\Gamma = 1500$ dobija drugačije rešenje. Slično, za instance DMCLP-R2 i DMCLP-R3 se vrednost funkcije cilja ne menja za $\Gamma \geq 165$. Za sve razmatrane instance najveći procenat povećanja vrednosti funkcije cilja iznosi oko 7.5%.

Tabela 3.5: Rezultati za instancu DMCLP-R1

Γ	CPLEX	t_{CPLEX}[s]	MA-LP	$t_{\text{MA-LP}}$[s]	Pov[%]
0	409348.590	104.7	*opt*	80.8	0.00
100	413742.525	296.9	*opt*	111.1	1.07
200	417467.295	459.7	*opt*	112.3	1.98
300	420789.705	833.5	*opt*	115.1	2.79
400	423789.660	900.5	*opt*	117.4	3.52
500	426494.040	413.7	*opt*	126.2	4.18
600	428960.385	568.1	*opt*	126.0	4.79
700	431180.295	514.4	*opt*	136.9	5.33
800	433153.875	511.0	*opt*	145.8	5.81
900	434890.785	891.2	*opt*	143.5	6.24
1000	436389.030	1921.4	*opt*	152.0	6.60
1100	437638.740	1757.2	*opt*	154.4	6.91
1200	438642.435	1119.8	*opt*	156.8	7.15
1300	439382.790	2094.8	*opt*	156.7	7.33
1400	439844.475	1837.5	*opt*	160.1	7.45
1500	440050.275	1825.7	*opt*	166.0	7.50
1600	440059.830	3163.1	*opt*	163.6	7.50
1700	440059.830	2137.1	*opt*	162.3	7.50
1800	440059.830	360.1	*opt*	95.0	7.50
Prosek	431154.968	1142.7	*opt*	135.9	5.32

Tabela 3.6: Rezultati za instancu DMCLP-R2

Γ	CPLEX	t_{CPLEX}[s]	MA-LP	$t_{\text{MA-LP}}$[s]	Pov[%]
0	1134.069	0.07	*opt*	1.37	0.00
15	1165.355	0.09	*opt*	1.46	2.75
30	1181.486	0.12	*opt*	1.68	4.18
45	1191.943	0.12	*opt*	1.71	5.10
60	1198.969	0.10	*opt*	1.43	5.72
75	1204.458	0.10	*opt*	1.46	6.20
90	1208.716	0.12	*opt*	1.45	6.58
105	1212.428	0.12	*opt*	1.43	6.91
120	1215.131	0.09	*opt*	1.40	7.14
135	1217.127	0.10	*opt*	1.41	7.32
150	1218.537	0.09	*opt*	1.40	7.44
165	1219.201	0.07	*opt*	1.37	7.50
180	1219.201	0.09	*opt*	1.41	7.50
195	1219.201	0.07	*opt*	1.66	7.50
210	1219.201	0.07	*opt*	1.66	7.50
225	1219.201	0.09	*opt*	1.66	7.50
240	1219.201	0.07	*opt*	1.93	7.50
255	1219.201	0.07	*opt*	1.66	7.50
270	1219.201	0.09	*opt*	1.68	7.50
285	1219.201	0.07	*opt*	1.66	7.50
300	1219.201	0.07	*opt*	1.68	7.50
315	1219.201	0.07	*opt*	1.66	7.50
330	1219.201	0.07	*opt*	1.66	7.50
Prosek	1207.767	0.09	*opt*	1.56	6.49

Tabela 3.7: Rezultati za instancu DMCLP-R3

Γ	CPLEX	t_{CPLEX}[s]	MA-LP	$t_{\text{MA-LP}}$[s]	Pov[%]
0	840.833	0.18	*opt*	3.13	0.00
15	863.703	0.28	*opt*	3.32	2.72
30	876.081	0.28	*opt*	3.37	4.19
45	884.201	0.31	*opt*	3.37	5.15
60	889.551	0.28	*opt*	3.29	5.79
75	893.679	0.28	*opt*	3.51	6.28
90	896.938	0.29	*opt*	3.47	6.67
105	899.546	0.28	*opt*	4.14	6.98
120	901.605	0.28	*opt*	4.36	7.22
135	903.079	0.31	*opt*	4.32	7.40
150	903.984	0.32	*opt*	4.19	7.51
165	904.049	0.18	*opt*	3.07	7.51
180	904.049	0.17	*opt*	3.07	7.51
195	904.049	0.18	*opt*	3.05	7.51
210	904.049	0.18	*opt*	3.08	7.51
225	904.049	0.18	*opt*	3.07	7.51
240	904.049	0.18	*opt*	3.08	7.51
255	904.049	0.18	*opt*	3.08	7.51
270	904.049	0.18	*opt*	3.08	7.51
285	904.049	0.18	*opt*	3.08	7.51
300	904.049	0.18	*opt*	3.05	7.51
315	904.049	0.17	*opt*	3.08	7.51
330	904.049	0.20	*opt*	3.10	7.51
345	904.049	0.17	*opt*	3.12	7.51
360	904.049	0.18	*opt*	3.08	7.51
375	904.049	0.18	*opt*	3.07	7.51
390	904.049	0.18	*opt*	3.05	7.51
405	904.049	0.18	*opt*	3.05	7.51
420	904.049	0.18	*opt*	3.05	7.51
435	904.049	0.18	*opt*	3.07	7.51
450	904.049	0.17	*opt*	3.08	7.51
465	904.049	0.18	*opt*	3.10	7.51
480	904.049	0.18	*opt*	3.08	7.51
495	904.049	0.18	*opt*	3.07	7.51
Prosek	898.421	0.21	*opt*	3.27	6.84

4 Problem p-hab centra neograničenih kapaciteta sa višestrukim alokacijama

Habovi predstavljaju posebnu vrstu čvorova u mreži u kojima se vrši kolekcija i konsolidacija protoka između dve izabrane lokacije. Umesto direktnog snabdevanja korisnika od strane određenog snabdevača, u takvim mrežama je dozvoljen transport preko habova. Svaki čvor u mreži se može pridružiti jednom ili više habova, pa se umesto direktnog transporta od snabdevača do korisnika, snabdevanje vrši od snabdevača do odgovarajućeg haba, zatim između habova, i na kraju od odgovarajućeg haba do korisnika. Obično je cena transporta između habova niža od cene transporta između ne-hab čvorova i cene transporta između ne-hab čvora i haba, pa se njihovim korišćenjem najčešće smanjuju ukupni troškovi.

Hab lokacijski problemi su široko proučavana oblast kombinatorne optimizacije, kako sa teorijskog, tako i sa praktičnog aspekta. Kod njih je obično potrebno identifikovati hab čvorove u mrežama i pridružiti ostale čvorove habovima, kako bi se uspostavio transport između svih parova snabdevača i korisnika. Neki od kriterijuma klasifikacije ove grupe problema su:

- Domen rešenja hab lokacijskog problema može biti diskretan ili kontinualan. Kod prvog skup potencijalnih habova se sastoji od konačnog broja čvorova, dok u drugom slučaju domen nije prebrojiv skup i najčešće je predstavljen pomoću ravni ili sfere.
- Prema funkciji cilja, izdvajaju se problemi hab medijane i problemi hab centra. Kod problema hab medijane funkcija cilja je oblika min-sum, gde se vrši minimizacija ukupne sume troškova transporta svakog para korisnik-snabdevač preko odgovarajućih habova. Kod problema hab centra se minimizuje maksimalna cena transporta između bilo koja dva čvora u mreži. U ovom slučaju funkcija cilja je min-max tipa.

- Prema broju hab čvorova, postoje problemi sa jednim habom i problemi sa više habova.
- Broj habova može biti unapred fiksiran ili se može određivati zajedno sa rešenjem.
- Prema kapacitetu habova, podela je izvršena na probleme neograničenih i ograničenih kapaciteta.
- U ukupnu cenu transporta može biti uključena i cena uspostavljanja haba. Cena u opštem slučaju ne mora biti uključena, može biti fiksna ili promenljiva.
- Prema načinu alokacije ne-hab čvora habu, postoji jednostruki i višestruki tip alokacije. Kod jednostruke alokacije svaki korisnik i svaki snabdevač je pridružen tačno jednom habu, tako da se transport između snabdevača i korisnika obavlja isključivo preko unapred utvrđene putanje, uzimajući u obzir pridruživanja habova čvorovima, koja su jedinstveno određena. Kod višestruke alokacije je svakom snabdevaču i svakom korisniku pridružuje više habova. Transport između snabdevača i korisnika se u tom slučaju realizuje preko onih habova koji obezbeđuju najnižu cenu transporta.

Najstariji oblik primene koncepta hab mreža pojavio se u telekomunikacionoj industriji, da bi nešto kasnije našao primenu pri logističkim sistemima, rečnom i avio-saobraćaju, transportu tereta, javnom prevozu, isporuci pošiljaka, itd. Tako pri letu avionom od početnog do završnog odredišta, određeni putnik može izabrati više međustanica, kako bi mu let bio što povoljniji. U tom slučaju se koristi šema višestruke alokacije. Slično, pri isporuci pošiljaka od pošiljaoca do primaoca, pošta može da se distribuira preko različitih terminala. Sa druge strane, u sistemima brze poštanske isporuke, najčešće se koristi jednostruka alokacija. U takvim sistemima koncept problema hab centra dolazi do izražaja, budući da je potrebno minimizovati maksimalno vreme isporuke za svaki par korisnik-snabdevač. Sa druge strane, pri odabiru međustanica za let od početne do završne stanice, značajni su problemi tipa hab medijane, jer je potrebno minimizovati ukupan trošak leta.

Prvi izvor iz literature koji je baziran na konceptima hab lokacijskih problema objavljen je još 1964. godine [70]. Oko dve decenije kasnije u [154] razmatra se primena mreže sa habovima u avio-saobraćaju. Najzad je u [122] i [123] razvijena prva matematička formulacija hab lokacijskog problema i metode za njegovo rešavanje. Među najvažnijim ranim radovima iz ove oblasti su [124] i [126], gde je razvijena kvadratna formulacija za neke hab lokacijske

probleme. U [20] i [21] su kasnije razmotrene razne varijante hab lokacijskih modela, čija funkcija cilja odgovara funkciji cilja nekoliko standardnih lokacijskih problema. U međuvremenu je, tokom zadnje tri decenije, objavljen veliki broj radova iz ove oblasti. Pregled hab lokacijskih modela i metoda za njihovo rešavanje može se pronaći u [3], [22] i [59].

U ovom poglavlju biće razmatran problem p-hab centra neograničenih kapaciteta sa višestrukim alokacijama. U odeljku 4.1 će biti dat opis problema i pregled relevantne literature. U odeljku 4.2 će biti prikazana njegova matematička formulacija, a u delu 4.3 i robusna, kod koje parametar koji predstavlja količinu transporta može biti podožan variranju, što odgovara realnoj situaciji. Biće predstavljen i memetski algoritam koji rešava problem, kao i eksperimentalni rezultati, koji će u slučaju determinističke varijante problema biti upoređeni sa rezultatima genetskog algoritma predloženog u [92].

4.1 Opis problema i pregled relevantne literature

Kod problema p-hab centra neograničenih kapaciteta sa višestrukim alokacijama (Uncapacitated multiple allocation p-hub center problem – UMApHCP) potrebno je iz skupa zadatih lokacija izabrati tačno p na kojima će biti uspostavljeni habovi, tako da je maksimalna cena transporta između bilo koja dva čvora minimalna. Transport se vrši od čvora snabdevača, preko najviše dva haba, do čvora korisnika. U nekim situacijama je moguće da jedan hab bude uključen u transport, a nekad se tranport vrši između direktno između snabdevača i korisnika. Kod višestruke alokacije, za razliku od jednostruke, svakom čvoru se može dodeliti više potencijalnih habova, zbog čega je pri transportu od čvora snabdevača do čvora korisnika potrebno izabrati one habove za koje je cena transporta između ta dva čvora minimalna.

Ovako definisan problem primenu može naći u avio-industriji, telekomunikacijama, i sl. Na primer, za isporučivanje poštanskih pošiljki od početne (pošiljaoca) do krajnje destinacije (primaoca), pošiljka može proći kroz nekoliko međustanica (habova).

Problem p-hab centra neograničenih kapaciteta sa višestrukim alokacijama su prvi put definisali O'Kelly i Miller 1991. godine u [125], gde je ilustrovana njegova primena u avio-saobraćaju između američkih gradova. Campbell je zatim problem formulisao kao problem kvadratnog programiranja u [20], gde je predložena i njegova reformulacija u obliku linearnog programa. Pritom je razvijena LP-relaksacija koja uspešno rešava instance

manjih dimenzija [20]. U [84] predloženo je nekoliko linearizacija kvadratne formulacije problema.

Ernst i sar. su u [57] razvili dve formulacije za UMApHCP koje koriste celobrojno programiranje, kao i heurističke metode za rešavanje varijante problema sa jednostrukim i višestrukim alokacijama. Prva formulacija koristi četvoroindeksne binarne promenljive y_{ijkm}, koje označavaju da je čvor snabdevač i pridružen habu k, a čvor korisnik j habu m. Međutim, ispostavlja se da promenljiva y_{ijkm} ne mora biti celobrojna, već ako se uslov $y_{ijkm} \in \{0, 1\}$ zameni uslovom $y_{ijkm} \geq 0$, postojaće uvek jedno optimalno rešenje kod koga je promenljiva y_{ijkm} binarna. Ovo je omogućilo razvoj formulacije koja koristi troindeksne promenljive i u praksi daje bolje rezultate nego odgovarajuća četvoroindeksna. U [57] je razvijena metoda grananja i ograničavanja (BnB) i heuristički pristup (Heur) za rešavanje UMApHCP. Metoda BnB je bazirana na činjenici da se fiksiranjem uspostavljenih habova problem svodi na rešavanje problema najkraćeg puta. Pomoću BnB metode dostugnuta su optimalna rešenja za AP instance manje i srednje veličine, dok je u nekim od ovih slučajeva odstupanje predložene heuristike od optimalnog rešenja bilo veliko. Predložena metoda grananja i ograničavanja je dostigla optimalna rešenja za AP instance većih dimenzija, za koje je $n = 100$, $p \leq 10$ i $n = 200$, $p = 3$.

U [92] je predložen efikasan genetski algoritam za rešavanje UMApHCP. Genetski algoritam koristi binarno kodiranje, fino-gradiranu turnirsku selekciju [61], kao i ukrštanje i mutaciju koji zadržavaju isti broj jedinica i nula u genetskom kodu. Kod mutacije je korišćena ideja sa zaleđenim bitovima, koja povećava verovatnoću promene bitova na pozicijama za koje je data vrednost bita za sve jedinke u populaciji ista [92], [145]. Primenjeno je više strategija za očuvanje raznovrsnosti jedinki u populaciji i sprečavanje preuranjene konvergencije [92]. Brzina izvršavanja algoritma je poboljšana korišćenjem keširanja [91]. Algoritam je primenjen na instance dimenzija do $n = 200$ i $p = 50$, koje su uspešno rešene za kratak vremenski period. Za $n \leq 100$ i $p \leq 10$ dostignuta su optimalna rešenja, a instance za koje je $n = 100, 15 \leq p \leq 30$ i $n = 200, 2 \leq p \leq 50$ su prvi put tada rešavane.

U [17] je primenjena efikasna osnovna metoda promenljivih okolina za rešavanje UMApHCP. Dobijeni rezultati su upoređeni sa heuristikom iz [57] nad CAB i AP instancama, kao i heuristikom zasnovanom na lokalnom pretraživanju nad instancama većih dimenzija. Dodatno, implementirana je varijanta lokalne pretrage koja je upoređena sa BVNS, pri čemu se pokazalo da BVNS dala bolje rezultate u pogledu kvaliteta rešenja i efikasnosti.

U literaturi postoje brojni radovi u kojima su razmatrani hab locacijski problemi koji se od UMApHCP razlikuju po funkciji cilja ili načinu alokacije. Na primer, u [146] razmatran je problem p-hab medijane neograničenih kapa-

citeta sa višestrukim alokacijama (Uncapacitated multiple allocation p-hub median problem – UMApHMP), kod koga se minimizuju ukupni transportni troškovi u mreži. Razvijen je genetski algoritam za njegovo rešavanje. Heuristika bazirana na elektromagnetizmu je primenjena na UMApHMP u [93], pri čemu su dobijeni kvalitetniji rezultati u odnosu na genetski algoritam. Tada su i po prvi put testirane instance dimenzija $n \leq 1000$ i $p \leq 20$. U [23] se razmatra specijalan slučaj problema p-hab centra, kod koga su poznate lokacije habova. Razvijene su matematičke formulacije za razne varijante problema. Za neke varijante problema je dokazano da su NP-teški, a za druge da su polinomske složenosti. Meyer i sar. u [107] razmatraju problem p-hab centra sa jednostrukim alokacijama. Egzaktan algoritam koji rešava problem se bazira na metodi grananja i ograničavanja, kao i varijanti problema nalaženja najkraćeg puta.

Međutim, postojeće metode koji su razvijene za druge varijante hab lokacijskih problema ne mogu se direktno primeniti na UMApHCP, pa je potrebno razviti algoritam koji je prilagođen karakteristikama razmatranog porblema. U ovom radu biće predlozena nova metaheuristička metoda koja je zasnovana na hibridizaciji optimizacije rojem čestica i lokalnog pretraživanja. Detalji implementacije su opisani u sekciji 4.4.

4.2 Matematička formulacija

Neka je sa I, $|I| = n$ označen skup svih čvorova u mreži, pri čemu svaki čvor odgovara lokaciji snabdevača, korisnika, kao i potencijalnoj lokaciji za uspostavljanje hab čvora. Neophodno je uspostaviti tačno p habova, pri čemu su preostali čvorovi označeni kao ne-hab čvorovi. Između svaka dva čvora date su jedinične cene transporta c_{ij}, $i, j \in I$, za koje je ispunjeno $c_{ij} = c_{ji}$. Dati graf ne mora biti kompletan, pa u opštem slučaju svaka dva čvora ne moraju biti direktno povezana. Ukoliko čvorovi i i j nisu povezani, tada se po dogovoru uzima da je $c_{ij} = \infty$. Jedinične cene transporta između čvorova mogu zadovoljavati i nejednakost trougla [22]. Jedinične cene transporta c_{ij} su proporcionalne rastojanjima d_{ij} između svaka dva čvora i i j, a za količinu transporta w_{ij} od i do j se obično uzima $w_{ij} = 1$ [57], [92]. Ovde će biti razmatrana formulacija za koju ne mora važiti $w_{ij} = 1$. Razlika između razmatrane linearne formulacije problema i formulacije iz [57] je što su u uslovu (4.7) parametri koji predstavljaju odgovarajuće količine transporta uključeni u sumu transportnih troškova.

Svaki ne-hab čvor se dodeljuje jednom ili više hab čvorova, dok se hab čvorovi dodeljuju sami sebi. Pritom se transport vrši obavezno kroz habove. Drugim rečima, početni i krajnji čvorovi u mreži ne mogu biti direktno pove-

zani. Tako put od početnog do krajnjeg čvora mora proći kroz jedan ili više habova. Potrebno je odrediti koji čvorovi će biti uspostavljeni kao habovi, tako da maksimalna cena transporta od proizvoljnog čvora snabdevača preko jednog ili više habova do proizvoljnog čvora korisnika bude minimalna.

Neka je $\alpha \in [0,1]$ realan parametar koji predstavlja koeficijent uštede prilikom transporta robe preko habova (ušteda iznosi $1-\alpha$). Ako je $\alpha = 1$, tada se transport između hab čvorova posmatra ravnopravno kao i protok između ostalih čvorova u mreži. Sa $c_{max} = \{c_{ij} : i,j \in I\}$ je označena maksimalna cena transporta između bilo koja dva čvora u mreži. Neka nenegativna realna promenljiva z_{max} označava vrednost funkcije cilja i neka su binarne promenljive H_k, $k \in I$, X_{ij}^k, $i,j,k \in I$ i Y_{ij}^l, $i,j,l \in I$ definisane na sledeći način:

$$H_k = \begin{cases} 1, & \text{ako je hab na lokaciji } k \text{ uspostavljen,} \\ 0, & \text{inače,} \end{cases}$$

$$X_{ij}^k = \begin{cases} 1, & \text{ako je snabdevač } i \text{ pridružen habu } k \\ & \text{prilikom transporta robe do korisnika } j, \\ 0, & \text{inače,} \end{cases}$$

$$Y_{ij}^l = \begin{cases} 1, & \text{ako je korisnik } j \text{ pridružen habu } l \\ & \text{prilikom transporta robe od snabdevača } i, \\ 0, & \text{inače.} \end{cases}$$

Imajući u vidu prethodno navedenu notaciju, UMApHCP se može formulisati kao problem mešovitog celobrojnog linearnog programiranja na sledeći način [57]:

$$\min z_{max} \tag{4.1}$$

uz ograničenja

$$\sum_{k \in I} H_k = p \tag{4.2}$$

$$X_{ij}^k \leq h_k \quad \forall i,j,k \in I, \tag{4.3}$$

$$Y_{ij}^l \leq h_l \quad \forall i,j,l \in I, \tag{4.4}$$

$$\sum_{k \in I} X_{ij}^k = 1 \quad \forall i,j \in I, \tag{4.5}$$

$$\sum_{l\in I} Y_{ij}^{l} = 1 \quad \forall i,j \in I, \tag{4.6}$$

$$\sum_{k\in I}(c_{ik}+\alpha c_{kl})w_{ij}X_{ij}^{k} + \sum_{m\in I} c_{mj}w_{mj}Y_{ij}^{m} - \alpha(1-Y_{ij}^{l})w_{ij}c_{max} \leq z_{max} \quad \forall i,j,l \in I, \tag{4.7}$$

$$H_k \in \{0,1\} \quad \forall k \in I, \tag{4.8}$$

$$X_{ij}^{k} \in \{0,1\} \quad \forall i,j,k \in I, \tag{4.9}$$

$$Y_{ij}^{l} \in \{0,1\} \quad \forall i,j,l \in I, \tag{4.10}$$

$$z_{max} \geq 0. \tag{4.11}$$

Minimizacija maksimalne cene transporta između dva čvora u mreži obezbeđena je sa (4.1) uz uslove (4.7). Ograničenje (4.2) postavlja broj uspostavljenih habova na tačno p. Uslovima (4.3) i (4.5) određeno je da je svaki čvor snabdevača u mreži dodeljen tačno jednom, prethodno uspostavljenom habu. Budući da je reč o višestrukim alokacijama, za svakom korisnika j, snabdevač i može biti dodeljen bilo kom od uspostavljenih p habova. Tako, na primer, za dva različita korisnika, i izbor habova može biti različit. Analogno je i značenje uslova (4.4) i (4.6) u odnosu na čvorove korisnika. Uslovi (4.8), (4.9) i (4.10) se odnose na binarnu prirodu promenljivih iz modela, a uslov (4.11) označava da realna promenljiva z_{max} mora biti nenegativna. Kako se podrazumeva da su parametri w_{ij} i c_{ij}, $i,j \in I$ nenegativni, uslov (4.11) može biti izostavljen, što važi i za deterministčku i za robusnu varijantu problema.

Uslovi (4.3) i (4.4) se mogu zameniti uslovima

$$\sum_{i\in I}\sum_{j\in I} X_{ij}^{k} \leq n^2 H_k \quad k \in I,$$

$$\sum_{i\in I}\sum_{j\in I} Y_{ij}^{l} \leq n^2 H_l \quad l \in I,$$

respektivno. Iako ovo rezultuje manjim brojem uslova, eksperimentalni rezultati su pokazali da je ova formulacija manje efikasna imajući u vidu vreme izvršavanja [57].

Primer 4.2 U primeru će biti razmotrene dve varijante problema p-hab centra neograničenih kapaciteta – sa jednostrukim i višestrukim alokacijama. Neka je data mreža od $n = 5$ čvorova od kojih je potrebno uspostaviti tačno $p = 2$ haba i neka je matrica rastojanja između čvorova $C = [c_{ij}]$, $1 \leq i, j \leq n$ definisana sa

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix} = \begin{bmatrix} 0 & 400 & 510 & 300 & 510 \\ 400 & 0 & 141 & 500 & 583 \\ 510 & 141 & 0 & 539 & 566 \\ 300 & 500 & 539 & 0 & 224 \\ 510 & 583 & 566 & 224 & 0 \end{bmatrix}$$

Za matricu C važi da je $c_{ij} = c_{ji}$ i $c_{ii} = 0$ za $1 \leq i, j \leq n$. Neka su jedinične cene transporta između ne-hab čvorova 1, a između habova 0.75.

Pri jednostrukoj alokaciji, u optimalnom rešenju uspostavljeni habovi su čvorovi 2 i 4, čvor 1 je pridružen habu 4, čvor 3 habu 2, a čvorovi 2 i 4 sami sebi. Najveći troškovi transporta se ostvaruju između čvorova 1 i 3 i iznose $c_{14} + 0.75c_{42} + c_{23} = 816$. Kod višestruke alokacije su uspostavljeni habovi 3 i 4, a svaki od čvorova je istovremeno pridružen svakom od habova. U optimalnom rešenju, maksimalna cena transporta je ostvarena između čvorova 2 i 5, i iznosi $c_{23} + c_{35} = 141 + 566 = 707$. Odavde se može uočiti da je ušteda prilikom primene višestruke alokacije u odnosu na jednostruku oko 13 procenata. Može se primetiti da je sam princip višestruke alokacije uticao da druga dva čvora postanu habovi. Ono što je olakšavajuća okolnost pri višestrukoj alokaciji je činjenica da pri transportu do dva različita korisnika jedan snabdevač može biti pridružen različitim habovima među uspostavljenim, odnosno pri transportu od dva različita snabdevača jedan korisnik može biti takođe pridružen različitim habovima.

4.3 Robusna fomulacija i složenost

U praksi se može dogoditi da se cena transporta između dva čvora u hab mreži poveća usled nepredviđenih okolnosti. Na primer, ekstremni vremenski uslovi, hitni ili nesrećni slučajevi mogu usporiti ili čak zaustaviti transport jednog dela mreže. U [136] se razmatra robusna formulacija hab lokacijskog problema neograničenih kapaciteta sa jednostrukim i višestrukim alokacijama. Predloženi robusni model koristi mešovito celobrojno nelinearno programiranje, da bi kasnije bio reformulisan kao model kvadratnog programiranja. Razvijena je efikasna linearna relaksacija koja dostiže optimalna rešenja za sve razmatrane instance. Prikazani su eksperimentalni

rezultati za instance sa najviše 25 čvorova u mreži. U [142] je prezentovan stohastički problem p-hab centra, u kome je vreme isporuke između snabdevača i korisnika podložno variranju. Razvijena je heuristika prilagođena problemu i prezentovani su odgovarajući eksperimentalni rezultati. Razne varijante hab lokacijskih problema neograničenih kapaciteta, gde cene transporta i potražnja korisnika mogu varirati se razmatraju u [34]. Razvijen je algoritam baziran na Monte-Karlo simulaciji, gde su uspešno rešene instance sa najviše 50 čvorova. U literaturi se može pronaći još radova koji razmatraju nepouzdanost ulaznih parametara u hab mrežama [80, 115, 139]. Ovde se po prvi put u literaturi razmatra robusna varijanta UMApHCP, u oznaci UMApHCP-R, kod koje je količina transporta između čvorova podložna variranju.

Neka količine transporta w_{ij} između svaka dva čvora i i j iz skupa I variraju, što je čest slučaj u praksi. Slično kao u poglavlju 2, pretpostavka je da parametri variraju u intervalu $[w_{ij}, w_{ij} + \hat{w}_{ij}]$ za $\hat{w}_{ij} \geq 0$. Neka je za sve parove $(i, j) \in I \times I$ skup G_{ij} definisan sa

$$G_{ij} = \begin{cases} \{(i,j)\}, & \text{ako je } \hat{w}_{ij} > 0, \\ \emptyset, & \text{ako je } \hat{w}_{ij} = 0 \end{cases}$$

i celobrojni parametar $\Gamma_{ij} \in [0, |G_{ij}|] \cap \mathbb{N} = \{0, 1\}$. Ako je $\Gamma_{ij} = 1$, to znači da parametar w_{ij} može, mada ne obavezno, biti podložan variranju, dok $\Gamma_{ij} = 0$ označava da on ostaje nepromenjen.

Analogno kao u poglavlju 2, na osnovu teoreme 1.1, sledi da se robusni model može napisati u obliku (4.1) – (4.6), (4.8) – (4.11) i

$$\sum_{k \in I} (c_{ik} + \alpha c_{kl}) w_{ij} X_{ij}^k + \sum_{m \in I} c_{mj} w_{mj} Y_{ij}^m - \alpha (1 - Y_{ij}^l)(w_{ij} + \Gamma_{ij} \hat{w}_{ij}) c_{max} + \Gamma_{ij} z'_{ij} + \sum_{(s,t) \in G_{ij}} r_{ijst} \leq z_{max} \quad \forall i, j, l \in I, \quad (4.12)$$

$$z'_{ij} + r_{ijst} \geq \hat{w}_{ij} (X_{ij}^k + Y_{ij}^l) \quad \forall i, j, k, l \in I \quad \forall (s,t) \in G_{ij}, \quad (4.13)$$

$$z'_{ij} \geq 0 \quad \forall i, j \in I, \quad (4.14)$$

$$r_{ijst} \geq 0 \quad \forall i, j \in I \quad \forall (s,t) \in G_{ij}. \quad (4.15)$$

Kako je skup G_{ij} najviše jednočlan, uslovi (4.12), (4.13) i (4.14) se mogu zameniti ekvivalentnim uslovima

$$\sum_{k\in I}(c_{ik}+\alpha c_{kl})w_{ij}X_{ij}^{k}+\sum_{m\in I}c_{mj}w_{mj}Y_{ij}^{m}$$
$$-\alpha(1-Y_{ij}^{l})(w_{ij}+\Gamma_{ij}\hat{w}_{ij})c_{max}+\Gamma_{ij}z_{ij}\leq z_{max}\quad\forall i,j,l\in I, \quad (4.16)$$

$$z_{ij}\geq\hat{w}_{ij}(X_{ij}^{k}+Y_{ij}^{l})\quad\forall i,j,k,l\in I, \quad (4.17)$$

$$z_{ij}\geq 0\quad\forall i,j\in I, \quad (4.18)$$

dok se uslov (4.15) u tom slučaju može izostaviti. Konačno, kod robusne formulacije problema, potrebno je odrediti (4.1) uz ograničenja (4.2) – (4.6), (4.8) – (4.11) i (4.16) – (4.18).

U [84] je pokazano da je UMApHCP NP-težak, pri čemu je izvršena redukcija sa problema dominirajućeg skupa. Zatim je u [57] dat jednostavan dokaz da su i UMApHMP i UMApHCP NP-teški, čak i za $\alpha = 0$. Robusna varijanta problema je takođe NP-teška [14].

4.4 Predloženi memetski algoritam

U ovom odeljku će biti predstavljen memetski algoritam (MA) koji rešava dati problem. On predstavlja hibridizaciju optimizacije rojem čestica i lokalne pretrage (LS), o čemu će u nastavku biti više reči. U algoritmu učestvuje n_r rešenja skupa N_r, $|N_r| = n_r$ i u svakoj iteraciji algoritma, dok se ne zadovolji kriterijum zaustavljanja, bira se deo manje kvalitetnijih rešenja na koji se primenjuje PSO algoritam, dok se kvalitetnija rešenja propuštaju direktno u narednu fazu. Po završenoj primeni PSO, u cilju dobijanja što kvalitetnijih rešenja, primenjuje se lokalna pretraga na svaku jedinku. Osnovna struktura memetskog algoritma za rešavanje UMApHCP i UMApHCP-R je prikazana algoritmom 4.1.

4.4.1 Kodiranje i funkcija cilja

Svako rešenje memetskog algoritma se kodira binarno, pri čemu je kôd dužine n. Bit na i-tom mestu ima vrednost 1 ako i samo ako je na i-toj poziciji uspostavljen hab. Svako dopustivo rešenje sadrži tačno p bitova čija je vrednost 1, budući da tačno p habova treba uspostaviti. Na primer, ako je kôd rešenja jednak 10110001, to znači da je $n = 8$, $p = 4$ i da su uspostavljeni habovi na pozicijama 1, 3, 4 i 8.

Algoritam 4.1 Memetski algoritam za rešavanje UMApHCP

```
1: Učitavanje ulaznih podataka
2: Generisanje početne populacije N_r
3: while nije ispunjen kriterijum zaustavljanja do
4:    Odrediti vrednosti funkcije cilja jedinki iz N_r
5:    Izabrati podskup elitnih jedinki N_e ⊂ N_r
6:    Primena PSO algoritma na jedinke iz N_r \ N_e
7:    for all r ∈ N_r do
8:       Primena lokalne pretrage na r
9:    end for
10:   Ažuriranje populacije N_r
11: end while
12: Ispis rešenja
```

Pri računanju funkcije cilja, najpre je potrebno izdvojiti sve čvorove koji su habovi. Neka je skup svih habova označen sa P. Kod dopustivog rešenja važi $|P| = p$. Kreira se kompletan graf koji se sastoji od $2n + p$ čvorova. Za svaki čvor $i \in I$ i svaki čvor $j \in P$ se kreira ivica, čija težina odgovara jediničnoj ceni transporta c_{ij} između ta dva čvora. Na sličan način se dodele vrednosti svim ivicama koje povezuju sve parove čvorova iz skupa P. Takođe, potrebno je konstruisati ivice koje polaze od čvorova iz skupa P do izlaznog skupa I. Vrednost težine svih ostalih ivica treba da iznosi ∞. Broj ivica čija je težina konačna iznosi $2np + p^2$. Na konstruisani graf se primenjuje Flojd-Voršalov algoritam za nalaženje najkraćeg puta za svaki par čvorova u mreži. Zatim je potrebno odrediti maksimalnu vrednost proizvoda najkraćeg puta među svim parovima (i, j) ($i \in I$ je čvor snabdevač, a $j \in P$ čvor korisnik) i parametra w_{ij}, gde je iskorišćena modifikacija Flojd-Voršalovog algoritma. Kod robusne formulacije se, ukoliko dolazi do variranja količine transporta, najkraći put između parova čvorova (i, j) množi sa $w_{ij} + \hat{w}_{ij}$. Za računanje funkcije cilja, najveću složenost ima Flojd-Voršalov algoritam, a imajući u vidu način na koji je konstruisan graf, složenost funkcije cilja iznosi $O(n^2 p)$.

4.4.2 PSO algoritam za rešavanje determinističke i robusne varijante UMApHCP

Algoritam optimizacije rojem čestica se, u sklopu memetskog algoritma, primenjuje na sličan način kao u problemu predloženom u poglavlju 2. Vektori pozicije i brzine $\mathbf{x}_i$ i $\mathbf{v}_i$ za svaku česticu i iz skupa nad kojim se primenjuje PSO su redom binarni i realni jedinični vektor dužine n, pri čemu $\mathbf{x}_i$ prestavlja ujedno i kôd i-tog rešenja. Početne vrednosti tih vektora se dodeljuju

uniformno, a zatim se pri svakoj iteraciji vrši njihovo ažuriranje. Pritom treba voditi računa jedino da vektor $\mathbf{x}_i$ nakon svake iteracije sadrži tačno p jedinica.

U svakoj iteraciji se l-ta koordinata vektora $\mathbf{x}_i$, $1 \leq l \leq n$ određuje na sledeći način:

$$\mathbf{x}_{i,l} \leftarrow \begin{cases} 1, & \text{ako je } r < (1 + e^{-\mathbf{v}_{i,l}})^{-1}, \\ 0, & \text{inače.} \end{cases}$$

Pri računanju vrednosti $\mathbf{x}_{i,l}$ koristi se sigmoidalna funkcija $(1 + e^{-\mathbf{v}_{i,l}})^{-1}$ [88]. Parametar r se bira uniformno iz intervala $(0, 1)$, a za sve l, $1 \leq l \leq n$ važi $\mathbf{x}_{i,l} \in \{0, 1\}$ i $\mathbf{v}_{i,l} \in [0, 1]$. Pri ovakvom računanju vrednosti vektora $\mathbf{x}_i$, može se dogoditi da se dobije nekorektno rešenje. U tom slučaju se izabere na proizvoljan način skup bitova koji imaju vrednost 1, koje je potrebno odbaciti u slučaju da je ukupna suma bitova veća od p, odnosno skup bitova koji imaju vrednost 0, koje je potrebno promeniti u 1 u slučaju da je ukupna suma bitova manja od p.

Pri ažuriranju vektora brzine čestice i, najpre se određuje vektor promene brzine koji iznosi

$$\Delta\mathbf{v}_{i,l} \leftarrow r_p c_p(\mathbf{p}_i - \mathbf{x}_i) + r_g c_g(\mathbf{g} - \mathbf{x}_i) + r_{g'} c_{g'}(\mathbf{g}' - \mathbf{x}_i),$$

gde je $r_p, r_g, r_{g'} \in U(0, 1)$, $c_p = c_g = 1.5$, $c_{g'} = 5$ i $1 \leq l \leq n$. Ovde je sa $\mathbf{g}'$ označeno drugo najbolje globalno rešenje. Ideja koja uzima u obzir drugo najbolje globalno rešenje je upotrebljena u [138], gde je eksperimentalnim putem pokazano da daje bolje rezultate od osnovne verzije PSO algoritma. Vrednosti parametara $c_p, c_g, c_{g'}$ se poklapaju sa vrednostima iz [138].

Vektor brzine se određuje po formuli

$$\mathbf{v}_{i,l} \leftarrow \begin{cases} 1, & \text{ako je } \mathbf{v}_{i,l} + \Delta\mathbf{v}_{i,l} > 1, \\ 0, & \text{ako je } \mathbf{v}_{i,l} + \Delta\mathbf{v}_{i,l} < 0, \\ \mathbf{v}_{i,l} + \Delta\mathbf{v}_{i,l}, & \text{inače.} \end{cases}$$

Algoritam PSO se primenjuje samo na ne-elitne jedinke. Ukoliko je vrednost funkcije cilja za vektor $\mathbf{x}_i$ bolja od vrednosti funkcije cilja za vektor $\mathbf{p}_i$, vrši se njegovo ažuriranje. Analogno se vrši ažuriranje i vektora $\mathbf{g}$ i $\mathbf{g}'$ u zavisnosti od svih vrednosti vektora $\mathbf{p}_i$, za sve čestice i iz roja.

4.4.3 Primena lokalne pretrage

U svakoj iteraciji memetskog algoritma lokalna pretraga se primenjuje na svaku jedinku po 10 puta u cilju dobijanja što kvalitetnijeg rešenja. Jedna

iteracija lokalne pretrage se primenjuje sve dok postoji poboljšanje. U svakom koraku se na proizvoljan način biraju dva bita na pozicijama l_1 i l_2, $1 \leq l_1, l_2 \leq n$, koji imaju različitu vrednost koda (jedan od čvorova na lokacijama l_1 i l_2 je hab, a drugi nije), a zatim se izvrši njihova zamena. Ukoliko dobijeno rešenje ima manju vrednost funkcije cilja od prvobitnog, ta vrednost se ažurira, a inače se vrši restauracija zamene bitova.

Pri prelasku u naredno rešenje, kada su dva bita l_1 i l_2 zamenila vrednosti, posmatra se najpre samo promena rastojanja između dva najudaljenija čvora kod prethodnog rešenja. Ukoliko se to rastojanje nije smanjilo, onda nema smisla iznova računati funkciju cilja, jer novo rešenje samim tim neće biti kvalitetnije od prethodnog. U tom slučaju se ta zamena automatski ne uzima u obzir i vrši se povratak na trenutno rešenje. Ovo u većini slučaja znatno smanjuje vreme računanja funkcije cilja, pa i algoritma u celini.

4.4.4 Ostali aspekti algoritma

Tokom izvršavanja memetskog algoritma, primenjuje se elitistički pristup, koji podrazumeva da se PSO algoritam ne primenjuje na sva rešenja iz skupa, već samo na ona koja nisu elitna, tj. ona čija je funkcija cilja manja od unapred zadate. Kvalitetnija rešenja se direktno prosleđuju u narednu fazu. Algoritam lokalne pretrage se, u cilju dobijanja što kvalitetnijih rešenja, primenjuje na sve elemente iz skupa N_r.

Početni skup rešenja se generiše uniformno. Pri tom odabiru treba jedino voditi računa da je broj uspostavljenih bitova čija je vrednost 1 jednak p. Algoritam se zaustavlja nakon što se najbolje rešenje ponovilo rep puta. Svi parametri algoritma će u nastavku biti određeni eksperimentalnim putem.

4.5 Eksperimentalni rezultati

U ovom odeljku će biti predstaljveni rezultati memetskog algoritma. Implementacija algoritma je izvršena u programskom jeziku C++, a rešavač CPLEX 12.1 je takođe inkorporiran u C++. Za svaku instancu, MA je pokretan po 15 puta. Sva testiranja su izvršena pod Windows 7 operativnim sistemom sa procesorom Intel i5-2430M od 2.4 GHz i RAM memorijom od 8 GB.

4.5.1 Test instance

Za testiranje opisanog problema izabrane su tri grupe instanci. Prva grupa instanci, u oznaci AP, odnosi se na realne podatke dobijene na osnovu

sistema isporuke australijske državne pošte [56]. Najveća instanca iz te grupe sadrži 200 čvorova, a grupisanjem pojedinih čvorova generisane su manje grupe instanci sa 10, 20, 25, 50 i 100 čvorova. Najveći broj habova koji je potrebno uspostaviti u tim instancama iznosi 50. Dok jedan čvor predstavlja jedno mesto koje ima svoj poštanski kôd, hab predstavlja odgovarajući centar konsolidacije. AP instance su prvi put u [57] iskorišćene za rešavanje UMApHCP, a nad njima je testiran i genetski algoritam koji rešava predloženi problem u [92]. Za sve AP instance je uzeto $w_{ij} = 1$, kako bi dobijeni rezultati mogli biti upoređeni sa vrednostima rezultata iz [57] i [92].

Druga grupa instanci, dimenzija $n = 300$ i $n = 400$, predstavlja modifikovane AP instance koje su korišćene pri rešavanju hab lokacijskog problema neograničenih kapaciteta sa jednostrukim alokacijama (USAHLP) [141]. Broj habova koje je potrebno uspostaviti iznosi $5 \leq p \leq 50$. Treća grupa instanci predstavlja modifikovane AP instance velikih dimenzija, za koje je $520 \leq n \leq 900$ i $5 \leq p \leq 100$. Instance su korišćene u [97] za testiranje hibridnog genetskog algoritma za USAHLP.

Iz pomenute tri grupe instanci, izabrane su ukupno tri instance za testiranje UMApHCP-R. Iz AP grupe, izabrana je ona za koju je $n = 40$ i $p = 10$, a nad njom je testiran algoritam sa 15 različitih vrednosti parametra Γ, za koji je $0 \leq \Gamma \leq 40^2 = 1600$. Za razliku od determinističke varijante problema, pri testiranju algoritma nad AP instancama za rešavanje UMApHCP-R, u obzir se uzima i količina transporta. Iz grupe modifikovanih AP instanci je odabrana jedna instanca dimenzija $n = 400$, $p = 20$. Pritom je $0 \leq \Gamma \leq 300^2$. Treća instanca za testiranje MA za robusnu varijantu predloženog problema je iz grupe velikih modifikovanih AP instanci, gde je $n = 520$, $p = 25$ i $0 \leq \Gamma \leq 270400$.

Za fiksiranu vrednost parametra Γ, pretpostavlja se da Γ vrednosti matrice $[w_{ij}]$, $1 \leq i \leq n$, $1 \leq j \leq n$ može varirati.

4.5.2 Podešavanje parametara memetskog algoritma

U memetskom algoritmu postoji nekoliko parametara čije je vrednosti neophodno preciznije odrediti kako bi algoritam konvergirao što kvalitetnijim rešenjima. Među njima su:

- $|N_r|$ – broj čestica u skupu svih rešenja N_r;
- rep – maksimalni broj ponavljanja najboljeg rešenja u algoritmu, koji predstavlja i kriterijum zaustavljanja;
- p_{el} – procenat elitnih jedinki iz algoritma (pri testiranju se broj elitnih jedinki uvek zaokružuje na najbliži ceo broj).

Tabela 4.1: Rezultati primene analize varijanse na podešavanje parametara

n	p	Parametar	SK	SS	PK	F	p-vr.
40	10	$\|N_r\|$	0.000	2	0.000	0.0000	1.0000
40	10	rep	0.000	1	0.000	0.0000	1.0000
40	10	p_{el}	0.000	1	0.000	0.0000	1.0000
200	30	$\|N_r\|$	0.000	2	0.000	0.0000	1.0000
200	30	rep	0.000	1	0.000	0.0000	1.0000
200	30	p_{el}	0.000	1	0.000	0.0000	1.0000
600	50	$\|N_r\|$	45.218	2	22.609	51.0008	0.0000
600	50	rep	1.024	1	1.024	0.2126	0.9219
600	50	$\|p_{el}\|$	0.829	1	0.829	0.1714	0.9364

U tom cilju iskorišćena je analiza varijanse [113]. Za pomenuta tri parametra korišćeni su sledeći nivoi:

- parametar $|N_r|$ ima tri nivoa – 10, 20 i 40;
- parametar rep ima dva nivoa – 100 i 200;
- parametar p_{el} ima dva nivoa – 66.67% i 75%.

Ukupan broj kombinacija iznosi $3 \cdot 2 \cdot 2 = 12$. U cilju testiranja svih ovih kombinacija, od skupa svih instanci su izabrane tri reprezentativne, čije dimenzije su navedene u tabeli 4.1.

Svaka kombinacija parametara za svaku instancu pokretana je po 15 puta i pritom je za svaku kombinaciju zabeležen najbolji dostignut rezultat. Ukupan broj pokretanja algoritma na taj način iznosi $12 \cdot 3 \cdot 15 = 540$.

Kritična vrednost za ovaj test iznosila je 0.05. Iz tabele 4.1 se može uočiti da jedino faktor $|N_r|$ ima značajnijeg uticaja na vrednost funkcije cilja, što se može zaključiti na osnovu vrednosti p za instancu za koju je $n = 600$ i $p = 50$. Za instance za koje je $n = 40$ i $p = 10$, odnosno $n = 200$ i $p = 30$, pri testiranju, svaka kombinacija parametra je dala isti rezultat, pa u tom slučaju se pokazalo da promena parametara nema nikakvog uticija. Slično, na osnovu rezultata se može uočiti da parametri rep i p_{el} nemaju značajnijeg uticaja na promenu vrednosti rezultata. U svim slučajevima vrednost $|N_r| = 40$ se pokazala najboljom, a za ostale parametre pri testiranju svih instanci je odabrano $rep = 200$ i $p_{el} = 75\%$.

4.5.3 Rezultati za deterministič̆ku varijantu problema

U ovom pododeljku će najpre biti prikazani rezultati za AP instance iz [56] dimenzija do $n = 200$ i $p = 40$. Rešenja dobijena memetskim algorit-

mom (MA) su upoređena sa rešenjima dobijenim pomoću metode grananja i ograničavanja BnB [57], heurističke metode Heur [57], genetskog algoritma GA [92] i BVNS metode iz [17]. Implementacija GA je testirana na istom računaru kao i implementacija MA i za svaku instancu oba algoritma su pokretana po 15 puta. Nažalost, za razliku od implementacija GA, implementacije BnB i Heur iz [57] i BVNS iz [17] nisu bile dostupne, tako da su pri poređenjima navedena vremena izvršavanja tih algoritama iz radova [57] i [17], respektivno. Impementacije BnB i Heur su testirane na DEC alpha sistemu [57], dok je implementacija BVNS testirana na Intel i7 2.8 GHz procesoru [17].

U tabeli 4.2 prikazani su rezultati i poređenja za AP instance dimenzija $10 \leq n \leq 100$, $p \leq 10$ i $n = 200$, $p = 3$. Za ove instance su pomoću BnB metode prethodno dostignuta optimalna rešenja [57]. Značenja naziva zaglavlja u tabelama 4.2–4.4 su:

- n – broj čvorova u mreži;
- p – broj habova koje je potrebno uspostaviti;
- BnB – vrednost optimalnog rešenja dostignutog pomoću BnB metode;
- t_{BnB} – prosečno vreme izvršavanja BnB za koje je po prvi put dostignuto najbolje rešenje (u sekundama);
- MA – vrednost najboljeg rešenja dostignutog pomoću MA, gde je sa *opt* naznačeno ukoliko je dostignuto optimalno rešenje, odnosno *najb* ukoliko je dostignuto najbolje poznato rešenje;
- t_{MA} – prosečno vreme izvršavanja MA za koje je po prvi put dostignuto najbolje rešenje (u sekundama);
- t_{MA}^{tot} – prosečno ukupno vreme izvršavanja MA (u sekundama);
- σ_{MA} – vrednost standardne devijacije MA u odnosu na optimalno ili najbolje poznato rešenje (u procentima);
- GA – vrednost najboljeg rešenja dostignutog pomoću GA, gde je sa *opt* naznačeno ukoliko je dostignuto optimalno rešenje, odnosno *najb* ukoliko je dostignuto najbolje poznato rešenje;
- t_{GA} – prosečno vreme izvršavanja GA za koje je po prvi put dostignuto najbolje rešenje (u sekundama);
- t_{GA}^{tot} – prosečno ukupno vreme izvršavanja GA (u sekundama);

- σ_{GA} – vrednost standardne devijacije GA u odnosu na optimalno ili najbolje poznato rešenje (u procentima);
- Heur – vrednost najboljeg rešenja dostignutog pomoću heuristike Heur, gde je sa *opt* naznačeno ukoliko je dostignuto optimalno rešenje, odnosno *najb* ukoliko je dostignuto najbolje poznato rešenje;
- t_{Heur} – prosečno vreme izvršavanja heuristike Heur za koje je po prvi put dostignuto najbolje rešenje (u sekundama);
- $odst_{\text{Heur}}$ – vrednost prosečnog odstupanja rešenja dobijenog heuristikom Heur od optimalnog ili najboljeg poznatog rešenja (u procentima);
- BVNS – vrednost najboljeg rešenja dostignutog pomoću BVNS, gde je sa *opt* naznačeno ukoliko je dostignuto optimalno rešenje, odnosno *najb* ukoliko je dostignuto najbolje poznato rešenje;
- t_{BVNS} – prosečno vreme izvršavanja BVNS za koje je po prvi put dostignuto najbolje rešenje (u sekundama);
- $odst_{\text{BVNS}}$ – vrednost prosečnog odstupanja rešenja dobijenog pomoću BVNS od optimalnog ili najboljeg poznatog rešenja (u procentima).

Na osnovu rezultata iz tabele 4.2, može se uočiti da su sve četiri heurističke metode, MA, GA, Heur i BVNS, efikasno dostigle optimalne rezultate, dobijene pomoću egzaktne BnB metode. Memetski algoritam je za svaku instancu, tokom svakog izvršavanja, dostigao optimalno rešenje, budući da je u svim slučajevima vrednost standardne devijacije 0%. Prosečna standardna devijacija GA je 0.03%, dok je vrednost prosečnog odstupanja heuristike Heur 0.79%, a BVNS 0.00%. Na osnovu vremena izvršavanja, može se zaključiti da MA oko 7 puta brže u proseku dostiže najbolje rešenje od GA. U proseku, ukupno vreme izvršavanja MA je manje od sekunde, što je oko 6 puta brže od ukupnog vremena izvršavanja GA, koje iznosi 6.503 s. Imajući u vidu da su metode BnB i Heur iz [57], BVNS iz [17] i predloženi MA testirani na različitim platformama, neophodno je normalizovati vremena BnB, Heur i BNVS metoda u cilju adekvatnog poređenja vremena izvršavanja. Koristeći pristup iz [45] i na osnovu podataka sa http://www.cpubenchmark.net/, vremena izvršavanja Heur iz [57] i BVNS iz [17] se normalizuju na sledeći način:

$$\text{NAT(ALG)} = \text{AT(ALG)} \cdot \frac{\text{PCPUS(CPU)}}{\text{PCPUS(Intel Core i5-2430M 2.4 GHz)}},$$

gde AT(ALG) predstavlja prosečno vreme izvršavanja BnB, Heur, odnosno BVNS algoritma, a PCPUS predstavlja Passmark CPU skor odgovarajućeg procesora. Prosečna vremena izvršavanja heusirtike BnB, Heur i BVNS metode iznose redom 3.58 s, 2.63 s i 0.12 s, a nakon normalizacije 0.57 s, 0.42 s i 0.19 s, respektivno. U tabeli su prikazana prosečna vremena izvršavanja pre normalizacije.

U tabeli 4.3 prikazani su rezultati i poređenja MA, GA i BVNS metode na velikim AP instancama dimenzija $n = 100$ i $n = 200$, za koje optimalna rešenja nisu poznata. Rezultati su prikazani na sličan način kao u tabeli 4.2, s tim što se u koloni Rešenje nalazi vrednost najboljeg poznatog rešenja, pa su standardna devijacija MA i GA i vrednost prosečnog odstupanja BVNS izračunate u odnosu na najbolje poznato rešenje. Prosečna standardna devijacija GA iznosi 0.04%, a prosečno odstupanje MA i BVNS metode 0.00%, pri čemu nisu poznate vrednosti rezultata BVNS metode za sve instance.

Iz tabele 4.3 se može uočiti da oba pristupa dostižu najbolja poznata rešenja za sve razmatrane AP instance dimenzija $n = 100$ i $n = 200$. Ipak, u ovom slučaju predloženi memetski algoritam se pokazao značajno efikasnijim od genetskog: ukupno vreme izvršavanja MA u proseku je 7.583 s, što je oko 58 puta brže u odnosu na ukupno vreme izvršavanja GA, koje iznosi 440.779 s. Dodatno, MA je u proseku dostigao najbolje poznato rešenje za 1.125 s, a GA za 4.338 s, što je oko 4 puta sporije. Prosečno vreme dostizanja najboljeg rešenja BVNS iznosi 1.86 s, a nakon normalizacije 2.90, što je oko 2.6 puta sporije. Imajući u vidu vremena izvršavanja po svim AP instancama iz tabela 4.2 i 4.3, može se zaključiti da je memetski algoritam dostizao u proseku 4.4 puta brže rešenje od genetskog, a 2.3 puta brže od BVNS metode.

Budući da su u praksi često prisutne mreže habova velikih dimenzija, memetski algoritam je dodatno testiran na modifikovanom skupu AP instanci velikih dimenzija iz [141] i [97]. Dobijeni rezultati su prikazani u tabeli 4.4, gde su prvi put u literaturi razmatrane instance za UMApHCP dimenzija do $n = 900$ i $p = 100$.

Prosečno ukupno vreme izvršavanja MA za modifikovane AP instance velikih dimenzija iznosilo je 6933.329 s, dok je MA dostizao najbolje rešenje u proseku za 4540.373 s. Prosečna vrednost standardne devijacije iznosila je 1.423%, što indukuje stabilnost memetskog algoritma. Odgovarajuće prosečne vrednosti parametra σ za fiksiranu vrednost broja čvorova za instance velikih dimenzija su prikazane na slici 4.1.

Tabela 4.2: Rezultati za AP instance sa poznatim optimalnim rešenjem

n	p	BnB	$t_{\text{BnB}}[s]$	MA	$t_{\text{MA}}[s]$	$t^{tot}_{\text{MA}}[s]$	$\sigma_{\text{MA}}[\%]$	GA	$t_{\text{GA}}[s]$	$t^{tot}_{\text{GA}}[s]$	$\sigma_{\text{GA}}[\%]$	Heur	$t_{\text{Heur}}[s]$	$odst_{\text{Heur}}[\%]$	BVNS	$t_{\text{BVNS}}[s]$	$odst_{\text{BVNS}}[\%]$
10	2	39922.112	0.00	*opt*	0.002	0.07	0.00	*opt*	0.001	0.34	0.00	*opt*	0.00	1.15	*opt*	0.00	0.00
10	3	32713.937	0.00	*opt*	0.002	0.07	0.00	*opt*	0.001	0.34	0.00	*opt*	0.00	0.00	*opt*	0.00	0.00
10	4	31577.965	0.00	*opt*	0.002	0.07	0.00	*opt*	0.001	0.35	0.00	*opt*	0.00	0.00	*opt*	0.01	0.00
10	5	30371.323	0.00	*opt*	0.003	0.13	0.00	*opt*	0.001	0.37	0.00	*opt*	0.00	0.00	*opt*	0.00	0.00
20	2	45954.151	0.00	*opt*	0.003	0.11	0.00	*opt*	0.001	0.37	0.00	*opt*	0.00	0.00	*opt*	0.00	0.00
20	3	40909.592	0.01	*opt*	0.005	0.19	0.00	*opt*	0.004	0.37	0.00	*opt*	0.00	6.09	*opt*	0.01	0.00
20	4	38320.251	0.01	*opt*	0.005	0.24	0.00	*opt*	0.010	0.41	0.00	*opt*	0.01	0.00	*opt*	0.03	0.00
20	5	37868.148	0.01	*opt*	0.003	0.05	0.00	*opt*	0.001	0.46	0.00	*opt*	0.01	0.00	*opt*	0.02	0.00
20	10	37868.148	0.05	*opt*	0.001	0.21	0.00	*opt*	0.006	2.19	0.00	*opt*	0.05	0.00	*opt*	0.04	0.00
25	2	51533.298	0.00	*opt*	0.004	0.12	0.00	*opt*	0.004	0.38	0.00	*opt*	0.00	0.00	*opt*	0.01	0.00
25	3	45552.497	0.01	*opt*	0.003	0.06	0.00	*opt*	0.011	0.42	0.00	*opt*	0.01	8.67	*opt*	0.01	0.00
25	4	45552.497	0.02	*opt*	0.003	0.03	0.00	*opt*	0.013	0.48	0.00	*opt*	0.02	0.00	*opt*	0.01	0.00
25	5	45552.497	0.03	*opt*	0.003	0.05	0.00	*opt*	0.008	0.78	0.00	*opt*	0.03	0.00	*opt*	0.01	0.00
25	10	45552.497	0.14	*opt*	0.001	0.22	0.00	*opt*	0.010	3.47	0.00	*opt*	0.13	0.00	*opt*	0.01	0.00
40	2	61140.798	0.03	*opt*	0.008	0.29	0.00	*opt*	0.009	0.45	0.00	*opt*	0.02	0.00	*opt*	0.02	0.00
40	3	56309.875	0.08	*opt*	0.018	0.44	0.00	*opt*	0.058	0.66	0.00	*opt*	0.05	0.37	*opt*	0.02	0.00
40	4	51279.142	0.14	*opt*	0.073	0.75	0.00	*opt*	0.130	1.15	0.00	*opt*	0.11	5.06	*opt*	0.02	0.00
40	5	49741.201	0.18	*opt*	0.017	0.17	0.00	*opt*	0.024	1.49	0.00	*opt*	0.17	0.00	*opt*	0.01	0.00
40	10	49741.201	1.00	*opt*	0.012	0.23	0.00	*opt*	0.011	8.85	0.00	*opt*	0.98	0.00	*opt*	0.02	0.00
50	2	61179.031	0.05	*opt*	0.017	0.41	0.00	*opt*	0.017	0.50	0.00	*opt*	0.04	0.00	*opt*	0.03	0.00
50	3	56729.936	0.16	*opt*	0.033	0.54	0.00	*opt*	0.056	1.02	0.00	*opt*	0.11	0.00	*opt*	0.02	0.00
50	4	52905.770	0.28	*opt*	0.041	0.76	0.00	*opt*	0.154	2.16	0.00	*opt*	0.20	0.00	*opt*	0.04	0.00
50	5	50707.866	0.52	*opt*	0.076	0.31	0.00	*opt*	0.102	3.02	0.00	*opt*	0.50	0.00	*opt*	0.03	0.00
50	10	50707.866	2.17	*opt*	0.020	0.25	0.00	*opt*	0.020	12.36	0.00	*opt*	2.15	0.00	*opt*	0.05	0.00
100	2	63197.103	0.56	*opt*	0.121	1.95	0.00	*opt*	0.085	1.23	0.00	*opt*	0.44	0.95	*opt*	0.08	0.00
100	3	57925.660	1.67	*opt*	0.322	2.65	0.00	*opt*	0.385	7.10	0.00	*opt*	1.48	0.00	*opt*	0.06	0.00
100	5	53949.329	22.52	*opt*	1.248	3.80	0.00	*opt*	10.833	26.59	0.28	*opt*	5.91	0.76	*opt*	0.50	0.00
100	10	51860.026	39.04	*opt*	0.178	0.65	0.00	*opt*	0.562	39.72	0.00	*opt*	38.07	0.00	*opt*	0.49	0.00
200	3	62945.552	35.37	*opt*	4.168	13.41	0.00	*opt*	32.331	71.44	0.46	*opt*	25.77	0.00	*opt*	1.92	0.00
Prosek			3.58	*opt*	0.220	0.97	0.00	*opt*	1.547	6.50	0.03	*opt*	2.63	0.79	*opt*	0.12	0.00

Tabela 4.3: Rezultati za AP instance čije optimalno rešenje nije poznato

n	p	Rešenje	$t_{\mathrm{MA}}[s]$	$t^{tot}_{\mathrm{MA}}[s]$	$\sigma_{\mathrm{MA}}[\%]$	$t_{\mathrm{GA}}[s]$	$t^{tot}_{\mathrm{GA}}[s]$	$\sigma_{\mathrm{GA}}[\%]$	$t_{\mathrm{BVNS}}[s]$	$odst_{\mathrm{BVNS}}[\%]$
100	15	51860.026	0.1	0.8	0.00	0.4	69.1	0.00	0.56	0.00
100	20	51860.026	0.1	1.4	0.00	0.3	100.3	0.00	1.04	0.00
100	25	51860.026	0.1	2.6	0.00	0.4	131.5	0.00	–	–
100	30	51860.026	0.1	4.1	0.00	0.2	167.4	0.00	1.88	0.00
200	2	67083.276	1.1	7.1	0.00	2.8	13.5	0.00	0.97	0.00
200	5	57419.319	9.3	22.6	0.00	39.9	120.6	0.68	2.15	0.00
200	10	55958.751	1.2	2.8	0.00	4.2	180.5	0.00	2.32	0.00
200	15	55958.751	0.5	1.5	0.00	3.4	293.0	0.00	4.24	0.00
200	20	55958.751	0.3	2.3	0.00	2.2	417.1	0.00	7.35	0.00
200	25	55958.751	0.4	3.6	0.00	3.7	634.1	0.00	–	–
200	30	55958.751	0.5	5.6	0.00	2.6	736.6	0.00	0.00	0.00
200	35	55958.751	0.6	8.4	0.00	0.8	811.3	0.00	–	–
200	40	55958.751	0.5	11.9	0.00	1.0	809.3	0.00	0.00	0.00
200	45	55958.751	0.7	16.5	0.00	1.1	962.1	0.00	–	–
200	50	55958.751	0.8	21.9	0.00	1.3	1164.7	0.00	0.00	0.00
Prosek			1.1	7.5	0.00	4.3	440.7	0.04	1.86	0.00

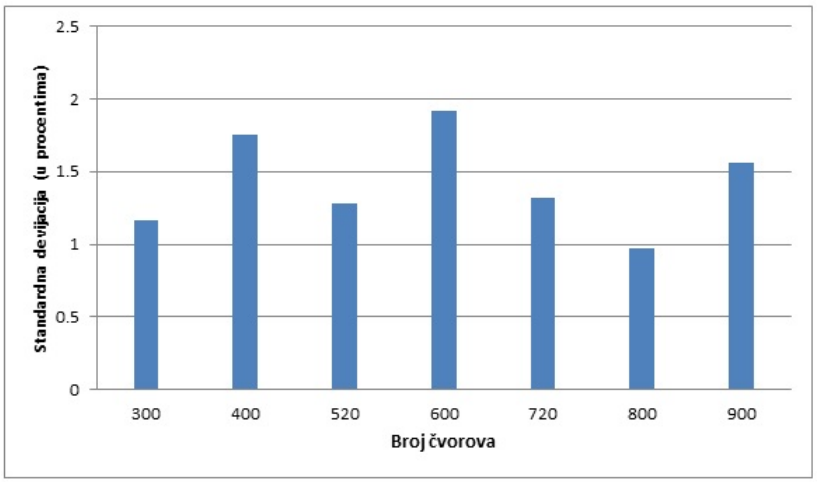
Tabela 4.4: Rezultati memetskog algoritma nad modifikovanim AP instancama

n	p	Rešenje	$t[s]$	$t_{tot}[s]$	$\sigma[\%]$
300	5	334334.59	17.1	115.6	0.36
300	10	254122.22	85.7	266.5	0.90
300	20	254122.22	239.2	572.8	1.46
300	25	254122.22	95.2	522.5	1.92
300	50	254122.22	417.2	1266.3	1.16
400	5	259021.13	181.8	360.2	1.84
400	10	259021.13	111.9	439.1	1.48
400	20	235229.97	494.9	1110.3	2.19
400	25	235229.97	330.6	1097.1	1.33
400	50	234836.11	550.3	2056.0	1.93
520	5	7406.01	266.7	572.2	0.50
520	10	6858.18	983.9	1607.8	0.95
520	20	6677.04	2078.6	3067.4	1.23
520	25	6649.27	1149.1	2386.9	2.75
520	50	6649.27	6495.4	8623.9	0.44
520	65	6645.11	2526.3	2782.1	1.28
520	75	6363.04	4137.2	5739.4	1.84
600	5	53224.34	176.0	425.6	1.13
600	10	53224.34	624.7	1108.0	0.29

600	20	53224.34	841.6	1706.3	1.73
600	25	46570.41	1258.9	2412.9	3.27
600	50	27303.54	10548.1	12625.3	2.75
600	65	27057.91	9660.6	11814.3	2.43
600	75	27057.91	4068.5	6870.4	1.87
720	5	12977.39	743.1	1161.1	0.02
720	10	11975.57	1506.5	2222.1	1.83
720	20	11975.57	2920.5	4211.7	2.19
720	25	11975.57	2144.4	3702.2	0.64
720	50	11701.93	2435.5	5291.8	1.76
720	65	11483.35	7217.2	11508.3	2.09
720	75	10863.65	11898.8	14932.2	0.76
720	85	10847.26	6489.0	11913.6	1.27
800	5	7881.32	165.2	620.1	0.25
800	10	6506.07	2105.4	3198.5	0.41
800	20	6017.34	10459.6	12039.6	2.55
800	25	6017.34	4902.1	7539.5	0.54
800	50	5940.96	8726.4	13731.3	1.81
800	65	5940.96	16741.0	19252.8	0.68
800	75	5940.96	3607.8	9368.6	0.17
800	85	5855.96	18079.6	22586.1	1.39
900	5	7445.84	385.9	8175.2	0.13
900	10	7192.44	2274.4	3704.3	0.34
900	20	6862.95	2889.0	5207.8	2.49
900	25	6484.80	3428.8	5660.3	2.03
900	50	6281.78	18365.6	23248.3	1.17
900	65	6281.78	6148.6	12361.3	1.97
900	75	6281.78	3275.8	9708.5	2.14
900	85	5836.01	13998.5	20623.1	0.99
900	95	5691.34	14691.5	22866.7	2.24
900	100	5691.34	14076.4	22279.7	2.06
Prosek		62340.47	4540.3	6933.3	1.42

4.5.4 Rezultati za UMApHCP-R

U ovom pododeljku je razmotren uticaj parametra Γ na vrednost funkcije cilja, pri čemu on uzima celobrojne vrednosti između 0 i $|I|^2$. Svi parametri w_{ij} u robusnoj varijanti problema su uvećani do 10% u odnosu na njihove



Slika 4.1: Prosečna vrednost standardne devijacije MA nad instancama većih dimenzija

početne vrednosti. Jasno je da se za $\Gamma = 0$ razmatra deterministička varijanta UMApHCP. Ako su ivice grafa redom numerisane, prema matrici vrednosti w_{ij}, u svim slučajevima se podrazumeva da prvih Γ ivica varira.

Pri generisanju eksperimentalnih rezultata za UMApHCP-R, korišćen je jedino predloženi memetski algoritam, budući da CPLEX rešavač nije u mogućnosti da dostigne optimalno rešenje čak ni za instance malih dimenzija i manje vrednosti parametra Γ. Iz skupa instanci za UMApHCP, kao što je prethodno opisano, izdvojene su tri instance, i nad njima je testiran MA za različite vrednosti parametra Γ. Dobijeni rezultati su prikazani u tabelama 4.5–4.7, čija zaglavlja imaju sledeća značenja:

- n, p – broj čvorova, odnosno habova u mreži;
- Γ – opisani parametar koji utiče na vrednost funkcije cilja;
- Rešenje – najbolje rešenje koje je dostigao memetski algoritam;
- $t[s]$ – prosečno vreme za koje je MA po prvi put dostigao najbolje rešenje (u sekundama);
- Pov[%] – povećanje vrednosti funkcije cilja u procentima u odnosu na slučaj $\Gamma = 0$;

Na osnovu rezultata iz tabele 4.5, može se uočiti da za instancu manjih dimenzija ($n = 40$ i $p = 10$), vrednost funkcije cilja ostaje ista kao u determinističkom slučaju za $0 \leq \Gamma \leq 500$. Za $\Gamma = 1000$, vrednost funkcije cilja

Tabela 4.5: Rezultati za robusnu instancu dimenzija $n = 40$ i $p = 10$

n	p	Γ	Rešenje	$t[s]$	Pov[%]
40	10	0	587366.466	0.01	0.00
40	10	1	587366.466	0.04	0.00
40	10	2	587366.466	0.43	0.00
40	10	5	587366.466	1.60	0.00
40	10	10	587366.466	1.01	0.00
40	10	20	587366.466	0.66	0.00
40	10	50	587366.466	0.60	0.00
40	10	100	587366.466	0.96	0.00
40	10	200	587366.466	0.85	0.00
40	10	300	587366.466	1.78	0.00
40	10	500	587366.466	0.67	0.00
40	10	1000	623960.093	0.77	6.23
40	10	1200	623960.093	0.31	6.23
40	10	1500	623960.093	0.41	6.23
40	10	1600	623960.093	0.94	6.23
Prosek			597124.766	0.74	1.66

se uvećava za 6.23% i ostaje nepromenjena pri daljem povećavanju vrednosti parametra Γ. U svim slučajevima, MA je dostigao najbolje rešenje za manje od 1.8 s.

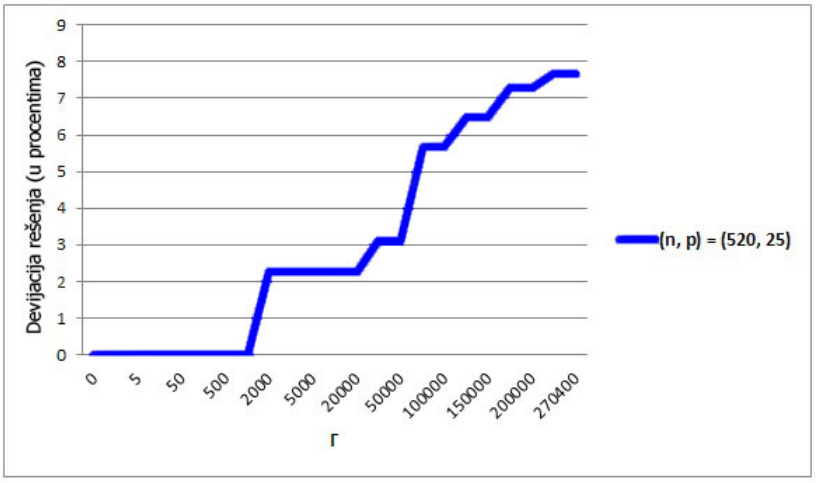
Za modifikovanu AP instancu dimenzija $n = 400$, $p = 20$, iz tabele 4.6 se može uočiti da se vrednost funkcije cilja povećava vrlo rano, za $\Gamma = 50$. Sa daljim povećanjem parametra, ta vrednost se postepeno povećava. Za $\Gamma \geq 75000$, vrednost funkcije cilja je povećana za 6.78% i ostaje takva za sve $75000 \leq \Gamma \leq 400^2$. U svim slučajevima, ukupno vreme izvršavanja je iznosilo manje od 11 minuta.

U tabeli 4.7 prikazani su rezultati memetskog algoritma za UMApHCP-R u slučaju velike instance, dimenzija $n = 520$, $p = 25$. Kao i u prethodnim slučajevima, u koloni Pov[%] se može uočiti da se vrednost funkcije cilja povećava sa povećanjem vrednosti parametra Γ. Najveća povećanje vrednosti funkcije cilja iznosilo je 7.664% i dostignuto je za $\Gamma \geq 250000$. Memetski algoritam se i u slučaju ove instance velikih dimenzija pokazao efikasnim, budući da je dostizao optimalno rešenje u svim slučajevima za manje od 40 minuta.

Na slici 4.2 prikazan je grafik povećanja vrednosti funkcije cilja u zavisnosti od parametra Γ za slučaj sa 520 čvorova i 25 habova. Na osnovu slike se može uočiti da povećanja nije bilo za $0 \leq \Gamma \leq 1000$. Vrednost

Tabela 4.6: Rezultati za robusnu instancu dimenzija $n = 400$ i $p = 20$

n	p	Γ	Rešenje	$t[s]$	Pov[%]
400	20	0	235229.97	494.9	0.00
400	20	1	235229.97	449.3	0.00
400	20	5	235229.97	447.4	0.00
400	20	10	235229.97	358.8	0.00
400	20	50	239224.29	492.4	1.69
400	20	100	239224.29	216.6	1.69
400	20	500	239224.29	501.8	1.69
400	20	1000	239224.29	213.3	1.69
400	20	5000	239224.29	592.0	1.69
400	20	10 000	239224.29	357.9	1.69
400	20	20 000	241664.35	534.1	2.73
400	20	50 000	248739.06	300.9	5.74
400	20	75 000	251178.91	398.3	6.78
400	20	90 000	251178.91	201.2	6.78
400	20	100 000	251178.91	640.9	6.78
400	20	150 000	251178.91	157.6	6.78
400	20	160 000	251178.91	468.6	6.78
Prosek			242503.740	401.5	3.09



Slika 4.2: Povećanje vrednosti funkcije cilja u zavisnosti od Γ za instancu dimenzija $n = 520$ i $p = 25$

Tabela 4.7: Rezultati za robusnu instancu dimenzija $n = 520$ i $p = 25$

n	p	Γ	Rešenje	$t[s]$	Pov[%]
520	25	0	6649.27	1149.1	0.00
520	25	1	6649.27	1535.7	0.00
520	25	5	6649.27	1371.0	0.00
520	25	10	6649.27	2315.3	0.00
520	25	50	6649.27	2024.7	0.00
520	25	100	6649.27	1309.0	0.00
520	25	500	6649.27	978.1	0.00
520	25	1000	6649.27	2152.8	0.00
520	25	2000	6799.23	1756.6	2.25
520	25	3000	6799.23	1370.0	2.25
520	25	5000	6799.23	2265.5	2.25
520	25	10 000	6799.23	2581.6	2.25
520	25	20 000	6799.23	2266.4	2.25
520	25	30 000	6854.92	1761.0	3.09
520	25	50 000	6854.92	1951.3	3.09
520	25	75 000	7027.89	2408.7	5.69
520	25	100 000	7027.89	2353.9	5.69
520	25	125 000	7079.42	1419.9	6.46
520	25	150 000	7079.42	2142.2	6.46
520	25	175 000	7134.90	1811.3	7.30
520	25	200 000	7134.90	1907.8	7.30
520	25	250 000	7158.88	1507.3	7.66
520	25	270 400	7158.88	1794.0	7.66
Prosek			6856.623	1831.9	3.12

funkcije cilja se povećava do 2.255%, pri čemu je vrednost parametra Γ u tim slučajevima između 0.74% i 7.4% od odgovarajuće maksimalne vrednosti 520^2. Za veće vrednosti Γ, vrednost funkcije cilja se postepeno povećava, a najveće povećanje iznosilo je oko 7.664%.

5 Zaključak

U ovoj disertaciji su razmatrana tri min-max problema diskretne optimizacije. Prvi razmatrani problem je višeperiodni lokacijski problem za raspoređivanje jedinica za reagovanje u hitnim slučajevima i predstavlja uopštenje problema iz [144]. Uopštenje u odnosu na ranije razmatrani problem se ogleda u tome što je, između ostalog, razmatrano više od jednog vremenskog perioda u kojem se vrši raspoređivanje jedinica, kao i minimalan broj jedinica u svakom od posmatranih perioda, što bolje odslikava situaciju u praksi. Drugi problem koji je razmatran je dinamički lokacijski problem maksimalnog pokrivanja sa više poluprečnika, koji predstavlja uopštenje dinamičkog lokacijskog problema maksimalnog pokrivanja iz [162]. Uopštenje je dobijeno uvođenjem više od jednog maksimalnog poluprečnika pokrivanja i parametara kojima se kontroliše mogućnost pokrivanja korisnika za svaki poluprečnik pokrivanja. Treći razmatrani problem je problem p-hab centra neograničenih kapaciteta sa višestrukim alokacijama. Naučni doprinos disertacije ogleda se u sledećem:

- Za višeperiodni lokacijski problem za raspoređivanje jedinica za reagovanje u hitnim slučajevima i dinamičkog lokacijskog problema maksimalnog pokrivanja sa više poluprečnika, razvijeni su odgovarajući matematički modeli, odnosno predložene mešovite celobrojne linearne formulacije problema. Ovi problemi do sada nisu razmatrani u literaturi. Matematički modeli predloženi u ovoj disertaciji predstavljaju uopštenja postojećih modela iz literature iz [144], odnosno [162].

- Za višeperiodni lokacijski problem za raspoređivanje jedinica za reagovanje u hitnim slučajevima je dat dokaz da je NP-težak. Preciznije, prvi put u literaturi je pokazano da je njegov potproblem razmatran u [144] NP-težak, odakle sledi da je problem razmatran u disertaciji takođe NP-težak.

- Za svaki od tri razmatrana diskretna problema razvijen je odgovarajući robusni model, kod koga određeni ulazni parametri variraju: broj incidenata u svakoj od posmatranih lokacija (naseljenih mesta) kod prvog

problema, zahtevi korisnika (potražnja za robom odnosno uslugom) u slučaju drugog problema i količina protoka (robe) između čvorova-snabdevača i čvorova-korisnika u slučaju trećeg problema.

- Razvijeni su hibridni metaheuristički algoritmi za rešavanje determinističkih i robusnih varijanti predloženih problema, koji su nastali kombinovanjem optimizacije rojem čestica kao populacijske metaheuristike i neke od heuristika zasnovanih na lokalnom pretraživanju u okviru koncepta memetskog algoritma. U slučaju višeperiodnog lokacijskog problema za raspoređivanje jedinica za reagovanje u hitnim slučajevima, umesto klasičnog lokalnog pretraživanja, korišćena je redukovana metoda promenljivih okolina. Za dinamički lokacijski problem maksimalnog pokrivanja sa više poluprečnika, predloženi memetski algoritam je hibridizovan sa metodom zasnovanom na linearnom programiranju. Svi elementi predloženih algoritama su prilagođeni osobinama razmatranih problema. Primenjene su razne strategije za ubrzanje predloženih algoritama, posebno pri računanju funcije cilja i heuristika lokalnog pretraživanja.

- Detaljno je analiziran uticaj različitih vrednosti parametara hibridnih algoritama na kvalitet dobijenih rešenja. Adekvatne vrednosti značajnih paramatara su određene analizom varijanse.

- Svaki od predloženih hibridnih algoritama je upoređen sa metodama iz literature koje su razvijene za specijalan slučaj odgovarajućeg problema, a dobijeni rezultati ukazuju na prednosti predloženih algoritama u smislu kvaliteta rešenja i/ili brzine izvršavanja, što se posebno ogleda u slučaju instanci većih dimenzija.

- Po prvi put u literaturi prikazani su rezultati za robusne i determinističke varijante problema koji do sada nisu bili razmatrani.

- U slučaju robusnih varijanti, analiziran je uticaj variranja ulaznih parametara na odstupanja u vrednostima funkcije cilja u odnosu na vrednost funkcije cilja determinističke varijante. U slučaju prvog razmatranog problema, dokazano je da se vrednosti funkcije cilja njegove robusne varijante mogu dobiti na osnovu odgovarajućih vrednosti funkcije cilja determinističke varijante.

Pravci daljeg rada uknjučuju dalje unapređenje predloženih hibridnih algoritama i njihovo prilagođavanje za rešavanje srodnih problema diskretne optimizacije i njihovih robusnih varijanti. Pritom se može razmatrati hibridizacija memetskih algoritama sa nekom egzaktnom heuristikom. Dodatno,

algoritam PSO, kao i primenjene varijante lokalnog pretraživanja su pogodne za paralelizaciju, tako da bi jedan od pravaca daljeg rada mogao biti njihova paralelizacija, što dodatno povećava efikasnost memetskih algoritama. Takođe, princip korišćen za dobijanje robusne varijante problema, može se primeniti i na srodne probleme i tako modelirati njihova robusna varijanta, u slučajevima da robusnosti podležu ulazni parametri iz funkcije cilja ili oni koji se nalaze u odgovarajućim ograničenjima.

Literatura

[1] Abdinnour, H. S. (1998). A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research*, 106 (2-3), 489-499.

[2] Aggarwal, C. C., Orlin, J. B., Tai, R. P. (1997). Optimized crossover for the independent set problem. *Operations Research*, 45 (2), 226-234.

[3] Alumur, S., Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, 190 (1), 1-21.

[4] Balcik, B., Beamon, B. M. (2008). Facility location in humanitarian relief. *International Journal of Logistics: Research and Applications*, 11 (2), 101-121.

[5] Bandler, J. W., Charalambous, C. (1974). Nonlinear Programming Using Minimax Techniques. *Journal of Optimization Theory and Applications*, 13, 607-619.

[6] Beasley, J., Chu, P. C. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4, 63-86.

[7] Ben-Tal, A., Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23, 769-805.

[8] Ben-Tal, A., Nemirovski, A. (1999). Robust solutions to uncertain programs. *Operations Research Letters*, 25, 1-13.

[9] Ben-Tal, A., Nemirovski, A. (2000), Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88, 411-424.

[10] Berman, O., Krass, D. (2002). The generalized maximal covering location problem. *Computers and Operations Research*, 29, 563-581.

[11] Berman, O., Krass, D., Drezner, Z. (2003). The gradual covering decay location problem on a network. *European Journal of Operational Research*, 151, 474-480.

[12] Berman, O., Wang, J. (2011). The minmax regret gradual covering location problem on a network with incomplete information of demand weights. *European Journal of Operational Research Society*, 208, 233-238.

[13] Bertsimas, D., Sim, M. (2001). The Price of Robustness. *Operations research*, 52 (1), 35-53.

[14] Bertsimas, D., Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98, 49-71.

[15] Birge, J., Louveaux, F. (1997). *Introduction to stochastic programming.* Springer-Verlag.

[16] Brady, R. M. (1985). Optimization strategies gleaned from biological evolution, *Nature*, 317, 804-806.

[17] Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D. (2015). A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem. *Optimization Letters*, 1-15.

[18] Brotcone, L., Laporte, G., Semet, F. (2003). Ambulance location and relocation models. *European Journal of Operational Research*, 147, 451-463.

[19] Bui, T. N., Moon, B. R. (1996). Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45 (7), 841-855.

[20] Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72, 387-405.

[21] Campbell, J. F. (1996). Hub location and p-hub median problem. *Operations Research*, 44 (6), 923-935.

[22] Campbell, J. F., Ernst, A., Krishnamoorthy, M. (2002). Hub location problems. In *Location Theory: Applications and Theory*, Z. Drezner, H. Hamacher (eds.), 373-407. Berlin-Heidelberg: Springer-Verlag.

[23] Campbell, A. M., Lowe, T. J., Zhang, L. (2007). The p-hub center allocation problem. *European Journal of Operational Research*, 176 (2), 819-835.

[24] Cao, F., Du, D. Z., Gao, B., Wan, P. J., Pardalos, P. M. (1995). Minimax problems in combinatorial optimization. In: *Minimax and Applications*, 269-292, Springer US.

[25] Chen, W., Zhang, J. (2010). A novel set-based particle swarm optimization method for discrete optimization problem. *IEEE Transactions on Evolutionary Computation*, 14 (2), 278-300.

[26] Charnes, A., Cooper, W. (1959). Chance-constrained programming, *Management Science*, 6 (1): 73-79.

[27] Church, R. ReVelle C. S. (1974). The maximal covering location problem. *Papers of the Regional Science Association*, 32, 101-118.

[28] Church, R.L., Cohon, J.L. (1976). Multiobjective location analysis of regional energy facility siting problems. *Report prepared for the U.S. Energy Research and Development Administration*, BNL 50567.

[29] Church, R. L. (1984). The planar maximal covering location problem. *Journal of Regional Science*, 2(24), 185-201.

[30] Clerc, M. (2012). *Standard Particle Swarm Optimisation*, Particle Swarm Central, Technical Report.

[31] Clerc, M. (2004). Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem. *New Optimization Techniques in Engineering*, Springer, 219-239.

[32] Clerc, M. (2005). *Binary Particle Swarm Optimisers: toolbox, derivations, and mathematical insights*, Technical Report.

[33] Colombo, F., Cordone, R., Lulli, G. (2016). The multimode covering location problem. *Computers and Operations Research*, 67, 25-33.

[34] Contreras, I., Cordeau, J. F., Laporte, G. (2011). Stochastic uncapacitated hub location. *European Journal of Operational Research*, 212 (3), 518-528.

[35] Costa, D., Dubuis, N., Hertz, A. (1995). Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics*, 1 (1), 105-128.

[36] Croft, H. T., Falconer, K. J., Guy, R. K. (1991). *Unsolved Problems in Geometry*, Springer-Verlag, New York.

[37] Current, J. R., Storbeck, J. E. (1988). Capacitated covering models. *Environment and Planning B: Planning and Design*, 15(2), 153-163.

[38] Current, J., Schilling, D. A. (1994). The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research* 73 (1), 114-126.

[39] Curtin, K. M., Hayslett, K., Qiu, F. (2007). Determining optimal police patrol areas with maximal covering and backup covering location models. *Networks and Spatial Economics*, 10, 125-145.

[40] Dantzig, G. (1955). Linear programming under uncertainty. *Management Science*, 1 (3-4): 197-206.

[41] Daskin, M. S. (1995). *Network and discrete location: Models, algorithms and applications*. New York, US, John Wiley and Sons.

[42] Dawkins, R. (1976). *The Selfish Gene*, Clarendon Press, Oxford.

[43] V.F. Demyanov, V. F., Molozemov, V. N. (1974). *Introduction to Minimax*, Wiley: New York.

[44] Dessouky, M., Ordóñez, F., Jia, H., Shen, Z. (2006). Rapid distribution of medical supplies. In: *Delay management in health care systems*, 309-338, Springer, New York.

[45] Dongarra, J. J. (2014). Performance of Various Computers Using Standard Linear Equations Software. *CS-89-85s, University of Manchester*.

[46] Drakulić, D., Marić, M., Takači, A. (2012). Solving Maximal Covering Location Problem (MCLP) by Using the Particle Swarm Optimization (PSO) Method. *Scientific work of the University of Ruse*, 51, 19-22.

[47] Drezner, Z., Wesolowsky, G. O., Drezner, T. (2004). The gradual covering problem. *Naval Research Logistics*, 51(6), 841-855.

[48] Drezner, T., Drezner, Z., Goldstein, Z. (2010). A stochastic gradual cover location problem. *Naval Research Logistics*, 57 (4), 367-372.

[49] Du, D. Z., Pardalos, P. M. (1995). *Minimax and Applications*, Kluwer: Dordrecht.

[50] Duhamel, C., Lacomme, P., Prins, C., Prodhon, C. (2008). A memetic approach for the capacitated location routing problem. *Proceedings of the 9th EU/Meeting on Metaheuristics for Logistics and Vehicle Routing*, Troyes, France.

[51] Eberhart, R. C., Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation*, 1, 84-88.

[52] Edelsbrunner, H., Tan, T. S., Waupotitsch, R. (1992). An O $(n^2 \log n)$ Time Algorithm for the Minmax Angle Triangulation. *SIAM Journal on Scientific and Statistical Computing*, 13 (4), 994-1008.

[53] Edelsbrunner, H. Tan, T. S. (1991). A quadratic time algorithm for the minmax length triangulation. *Proceedings of 92-nd Annual Symposium on Foundations of Computer Science*, San Jun, Purrto Rico, 414-423.

[54] El-Ghaoui, Lebret, H. (1997). Robust solutions to least-square problems to uncertain data matrices. *SIAM Journal on Matrix Analysis and Applications*, 18, 1035-1064.

[55] El-Ghaoui, L., Oustry, F., Lebret, H. (1998). Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9, 33-52.

[56] Ernst, A. T., Krishnamoorthy, M. (1998). Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. *European Journal of Operational Research*, 104, 100-112.

[57] Ernst, A. T., Hamacher, H., Jiang, H., Krishnamoorthy, M., Woeginger, G. (2009). Uncapacitated single and multiple allocation p-hub center problems. *Computers and Operations Research*, 36 (7), 2230-2241.

[58] Espejo, I., Marn, A., Rodrguez-Cha, A.M. (2012). Closest assignment constraints in discrete location problems. *European Journal of Operational Research*, 219 (1), 49-58.

[59] Farahani, R. Z., Hekmatfar, M., Arabani, A. B., Nikbakhsh, E. (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers and Industrial Engineering*, 64 (4), 1096-1109.

[60] Feder T., Greene, D. H. (1988). Optimal algorithms for approximate clustering. *Proccedings of 20th ACM Symposium of Theory of Computing*, 434-444.

[61] Filipović, V. (2003). Fine-grained tournament selection operator in genetic algorithms. *Computing and Informatics*, 22 (2), 143-161.

[62] Fischer, T., Merz, P. (2007). A memetic algorithm for the optimal communication spanning tree problem. *Hybrid Metaheuristics 4th International Workshop*, HM 2007. In: *Lecture Notes in Computer Science*, 4771, Springer, 170-184.

[63] Francis, R. L., Lowe, T. J. (1992). On worst-case aggregation analysis for network location problems, *Annals of Operations Research*, 40, 229-246.

[64] Freisleben, B., Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *IEEE International Conference on Evolutionary Computation*, IEEE Press, Nagoya, Japan, 616-621.

[65] Freisleben, B., Merz, P. (1996). New genetic local search operators for the traveling salesman problem. In: *Parallel Problem Solving from Nature IV*, 890-900.

[66] Garey, M. R., Johnson, D. S. (1978). *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman.

[67] Goel, A., Gruhn, V. (2008). A general vehicle routing problem. *European Journal of Operational Research*, 191 (3), 650-660.

[68] Goldberg, D. E., Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 1, 69-93.

[69] Grefenstette, J. J. (1987). Incorporating problem specific knowledge into genetic algorithms. In: L. Davis (Ed.), Genetic Algorithms and Simulated Annealing, *Research Notes in Artificial Intelligence*, Morgan Kaufmann Publishers, 42-60.

[70] Hakimi, S. L. (1964). Optimum location of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12, 450-459.

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

[72] Hale, W. K. (1980). Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68, 1497-1514.

[73] Hanjoul, P., Peeters, D. (1987). A facility location problem with clients' preference orderings. *Regional Science and Urban Economics*, 17, 451-473.

[74] Hansen, P., Jaumard, B., Mladenović, N., Parreira, A. (2000). Variable neighborhood search for weighted maximum satisfiability problem. *Les Cahiers du GERAD*, 2000–2062.

[75] Hansen, P., Mladenović, N., Perez-Brito, D. (2001). Variable neighborhood decomposition search. *Journal of Heuristics*, 7 (4), 335-350.

[76] Hansen, P., Mladenović, N. (2003). Variable neighborhood search. Glover F., Kochenberger, G. (eds). *Handbook of Metaheuristics*, Kluwer, Dordrecht, 145-184.

[77] Hart, W. (1994). *Adaptive global optimization with local search.* PhD thesis, University of California, San Diego.

[78] Hart, W. E., Krasnogor, N. Smith, J. E. (2004). Memetic evolutionary algorithms, in: W. E. Hart, N. Krasnogor, J. E. Smith (Eds.), *Recent Advances in Memetic Algorithms*, Springer, Berlin, Germany.

[79] Hogg, J. (1968). The siting of fire stations. *Operational Research Quarterly*, 19 (3), 275-287.

[80] Huang, J., Wang, Q. (2009). Robust Optimization of Hub-and-Spoke Airline Network Design Based on Multi-Objective Genetic Algorithm. *Journal of Transportation Systems Engineering and Information Technology*, 9 (3), 86-92.

[81] Jia, H., Ordóñez, F., Dessouky, M. (2005). A modeling framework for facility location of medical services for large-scale emergencies. *IIE Transactions*, 39 (1), 41-55.

[82] Jog, P., Suh, J., Gucht, D. V. (1989). The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem. *3rd International Conference on Genetic Algorithms*, 110-115.

[83] Kall, P., Mayer, J. (2005). *Stochastic linear programming: Models, theory and computation.* Springer-Verlag.

[84] Kara, B. Y., Tansel, B. C. (2000). On the single-assignment p-hub center problem. *European Journal of Operational Research*, 125 (3), 648-655.

[85] Karasakal, O., Karasakal, E. K. (2004). A maximal covering location model in the presence of partial coverage. *Computers and Operations Research*, 31, 1515-1526.

[86] Karp, R. M. (1972). Reducibility among combinatorial problems. In: R. E. Miller and J. W. Thatcher (Eds.), *Complexity of Computer Computations*, 85-104. New York: Plenum.

[87] Kennedy, J., Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, IV, 1942-1948.

[88] Kennedy, J. (1997). The particle swarm: social adaptation of knowledge. *Proceedings of IEEE International Conference on Evolutionary Computation*, 303-308.

[89] Kim, D. G., Kim, Y. D. (2010), A branch and bound algorithm for determining locations of longterm care facilities. *European Journal of Operational Research*, 206. 168-177.

[90] Kouvelis, P., Yu, G. (1997). *Robust discrete optimization and its applications*. Kluwer Academic Publishers, Norwell, MA.

[91] Kratica, J. (1999). Improving performances of the genetic algorithm by caching. *Computers and Artificial Intelligence*, 18 (3), 271-283.

[92] Kratica, J., Stanimirović, Z. (2006). Solving the uncapacitated multiple allocation p-hub center problem by genetic algorithm, *Asia-Pacific Journal of Operational Research*, 23 (4), 425-437.

[93] Kratica, J. (2013). An electromagnetism-like metaheuristic for the uncapacitated multiple allocation p-hub median problem. *Computers and Industrial Engineering*, 66 (4), 1015-1024.

[94] Lee, D. T., Lin, A. K. (1986). Gernalized Delaunay triangulations for planar graphs. *Discrete and Computational Geometry*, 1, 161-194.

[95] Legg, S., Hutter, M., Kumar, A. (2004). *Tournament versus Fitness Uniform Selection*. Technical Report, IDSIA-04-04.

[96] Marianov, V., ReVelle, C. (1995). Siting of emergency services. In: *Facility Location: A Survey of Applications and Methods*, Springer, New York, 199-223.

[97] Marić, M., Stanimirović, Z., Stanojević, P. (2013). An efficient memetic algorithm for the uncapacitated single allocation hub location problem. *Soft Computing*, 17 (3), 445-466.

[98] Marić, M., Stanimirović, Z., Djenić, A., Stanojević, P. (2014). Memetic Algorithm for Solving the Multilevel Uncapacitated Facility Location Problem. *Informatica*, 25 (3), 439-466.

[99] Marić, M., Stanimirović, Z., Božović, S. (2013). Hybrid metaheuristic method for determining locations for long-term health care facilities. *Annals of Operations Research*, 227 (1), 3-23.

[100] Marín, A. (2011). The discrete facility location problem with balanced allocation of customers. *European Journal of Operational Research*, 210 (1), 27-38.

[101] Mathias, K., Whitley, D. (1992). Genetic operators, the fitness landscape and the traveling salesman problem, In: *Parallel Problem Solving from Nature II*, 219-228.

[102] Megiddo, N., Zemel, E., Hakimi, S. L. (1983). The maximum coverage location problem. *SIAM Journal on Algebraic Discrete Methods*, 4 (2), 253-261.

[103] Meggido, N., Supowit, K. (1984). On the complexity of some common geometriclocation problems, *SIAM Journal of Computing*, 13, 182-196.

[104] Merz, P., Freisleben, B. (1998). Memetic algorithms and the fitness landscape of the graph Bi-partitioning Problem. In: *Lecture Notes in Computer Science*, 1498, Springer-Verlag, Amsterdam, The Netherlands, 765-774.

[105] Merz, P., Freisleben, B. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, *IEEE Transactions on Evolutionary Computation*, 4 (4), 337-352.

[106] Merz, P., Fischer, T. (2007). A memetic algorithm for large traveling salesman problem instances. In: *7th Metaheuristics International Conference.*

[107] Meyer, T., Ernst, A. T., Krishnamoorthy, M. (2009). A 2-phase algorithm for solving the single allocation p-hub center problem. *Computers and Operations Research*, 36 (12), 3143-3151.

[108] Miller, B. L., Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9 (3), 193-212.

[109] Mišković, S., Stanimirović, Z., Grujičić I. (2016). Solving the robust two-stage capacitated facility location problem with uncertain transportation costs. *Optimization Letters*, 1-16.

[110] Mišković, S., Stanimirović, Z. (2013). A Memetic Algorithm for Solving Two Variants of the Two-Stage Uncapacitated Facility Location Problem. *Information Technology and Control*, 42 (2), 131-149.

[111] Mladenović, N., Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24 (11), 1097-1100.

[112] Mladenović, N., Petrović, J., Kovačević-Vujčić, V., Čangalović, M. (2003). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search. *European Journal of Operational Research*, 151, 389-399.

[113] Montgomery, D. C. (2005). *Design and Analysis of Experiments*, sixth edition, John Wiley and Sons, New York.

[114] Moreno-Perez, J. A., Hansen, P., Mladenovic, N. (2005). Parallel variable neighborhood search. Alba, E. (ed). *Parallel metaheuristics: a new class of algorithms*.

[115] Morteza, A.,Mahdavi-Amiri, N., Shiripour, S. (accepted in 2015). Modeling and solving a capacitated stochastic location-allocation problem using sub-sources. *Soft Computing*.

[116] Moscato, P., Norman, M. (1989). A competitive and cooperative approach to complex combinatorial search, *Technical Reports 790*, Caltech Concurrent. Computation Program.

[117] Moscato, P., Norman, M. G. (1992). A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *Parallel computing and transputer applications*, 1, 177-186.

[118] Mulvey, J., Vanderbei, R., Zenios, S. (1995). Robust optimization of large-scale systems. *Operations Research*, 43 (2), 264-281.

[119] Muthuswamy, S., Lam, S. (2011). Discrete particle swarm optimization for the orienteering problem. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 18 (2), 92-102.

[120] Neri, F., Cotta, C. (2012). *Handbook of memetic algorithms*, vol. 379. Springer, Heidelberg.

[121] Neri, F., Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2, 1-14.

[122] O'Kelly, M. E. (1986). The location of interacting hub facilities. *Transportation Science*, 20, 92-106.

[123] O'Kelly, M. E. (1986). Activity levels at hub facilities in interacting networks. *Geographical Analysis*, 18 (4), 343-356.

[124] O'Kelly, M. E. (1987). A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32, 393-404.

[125] O'Kelly, M. E., Miller, H. J. (1991). Solution strategies for the single facility minimax hub location problem. *Papers in Regional Science: The Journal of the RSAI*, 70, 367-380.

[126] O'Kelly, M. E. (1992). Hub facility location with fixed costs. *Papers in Regional Science*, 71, 292-306.

[127] Pirkul, H., Schilling D. A. (1988). The siting of emergency service facilities with workload capacities and backup service. *Management Science*, 34 (7), 896-908.

[128] Poli, R. (2007). An analysis of publications on particle swarm optimisation applications. *Technical Report CSM-469*. Department of Computer Science, University of Essex, UK.

[129] Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 1-10.

[130] Preparata, F. P., Samos, M. L. (1985). *Computational Geometry – an Introduction*. SpringerVerlag, New York.

[131] Rajagopalan, H. K., Saydam, C., Xiao, J. (2008). A multiperiod set covering location model for dynamic redeployment of ambulances. *Computers and Operations Research*, 35 (3), 814-826.

[132] ReVelle, C., Swain, R. (1970). Central facilities location. *Geographical Analysis*, 2, 30-42.

[133] ReVelle, C., Hogan, K. (1989). The maximum availability location problem. *Transportation Science*, 23, 192-200.

[134] Roy, R., Dehuri, S., Cho, S. B. (2012). A Novel Particle Swarm Optimization Algorithm for Multi-Objective Combinatorial Optimization Problem. *International Journal of Applied Metaheuristic Computing*, 2(4), 41-57.

[135] Savas, E. S. (1969). Simulation and cost-effectiveness analysis of New York emergence ambulance service. *Management Science*, 18 (12), 608-626.

[136] Shahabi, M., Unnikrishnan, A. (2014). Robust hub network design problem. *Transportation Research Part E: Logistics and Transportation Review*, 70, 356-373.

[137] Shi, Y., Eberhart, R. C. (1998). A modified particle swarm optimizer. *Proceedings of IEEE International Conference on Evolutionary Computation*, 69-73.

[138] Shin Y. B., Kita E. (2014). Search performance improvement of Particle Swarm Optimization by second best particle information. *Applied Mathematics and Computation*, 246, 346-354.

[139] Sibel, A., Nickel, S. Saldanha-da-Gama, F. (2012). Hub location under uncertainty. *Transportation Research Part B: Methodological*, 46 (4), 529-543.

[140] Sibson, R. (1978). Locally equiangular triangulation. *Computer Journal*, 21, 243-245.

[141] Silva, M. R., Cunha, C. B. (2009). New simple and efficient heuristics for the uncapacitated single allocation hub location problem. *Computers and Operations Research*, 36, 3152-3165.

[142] Sim, T., Lowe, T. J., Thomas, B. W. (2009). The stochastic p-hub center problem with service-level constraints. *Computers and Operations Research*, 36 (12), 3166-3177.

[143] Soyster, A. L. (1973). Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operational Research*, 21, 1154-1157.

[144] Stanimirović, Z., Grujičić, I., Trifunović, D. (accepted for publication in 2015). Modeling the emergency service network of police special forces units for high-risk law enforcement operations. *Information Systems and Operational Research*.

[145] Stanimirović, Z. (2004). Rešavanje nekih diskretnih lokacijskih problema primenom genetskih algoritama. Magistarski rad. Matematički fakultet, Beograd.

[146] Stanimirović, Z. (2008). An efficient genetic algorithm for the uncapacitated multiple allocation p-hub median problem. *Control and Cybernetics*, 37, 669-692.

[147] Stanimirović, Z., Marić, M., Božović, S. (2011). An Efficient Hybrid Algorithm for Locating Long-Term Care Facilities. *IX Balkan Conference on Operational Research*, 409-416.

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

[149] Stanimirović, Z., Mišković, S. (2013). Efficient Metaheuristic Approaches for Exploration Of Online Social Networks. In: W. C. Hu, N. Kaabouch (Eds.), *Big Data Management, Technologies, and Applications*, 222-269, Pennsylvania, USA: IGI Global.

[150] Stanimirović, Z., Mišković, S. (2014). A Hybrid Evolutionary Algorithm for Efficient Exploration of Online Social Networks. *Computing and Informatics*, 33 (2), 410-430.

[151] Swersey, A. J. (1994). The deployment of Police, Fire and Emergency Medical Units. In: Pollock, S. M. et al. (eds). *Handbooks in OR and MS*, Vol. 6, Elsevier Science, B.V., Chapter 6, 151-200.

[152] Takači, A., Marić, M., Drakulić, D. (2012). The role of fuzzy sets in improving maximal covering location problem (MCLP). In: Intelligent Systems and Informatics (SISY), *2012 IEEE 10th Jubilee International Symposium*, 103-106.

[153] Takači, A., Marić, M., Drakulić, D. (2013). Solving minimal covering location problem (MinCLP) with the aid of fuzzy sets. In: Intelligent Systems and Informatics (SISY), *2013 IEEE 11th International Symposium*, 177-180.

[154] Toh, R. S., Higgins, R. C. (1985). The impact of hub and spoke network centralization and route monopoly on domestic airline profitability. *Transportation Journal*, 24, 16-27.

[155] Trelea, I. C. (2003). The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85 (6), 317-325.

[156] Valinsky, D. (1955). A determination of the optimum location of firefighting units in New York City. *Operations Research*, 4 (3), 494-512.

[157] Vanderbei, R. J. (2001). *Linear programming. Foundations and Extensions*. Second Edition-International Series in Operations Research and Management Science, Princeton University.

[158] Weaver, J. R., Church, R. L. (1985). A Median Location Model with Nonelosest Facility Service. *Transportation Science*, 19, 58-74.

[159] White, J. A., Case, K. E. (1974). On covering problems and the central facilities location problem. *Geographical Analysis*, 6, 281-293.

[160] Wolf, S., Merz, P. (2007). Evolutionary local search for the super-peer selection problem and the p-hub median problem. In: *Hybrid Metaheuristics*, 1-15, Springer.

[161] Yu, G., Kouvelis, P. (1995). On min-max optimization of a collection of classical discrete optimization problems. *Minimax and Applications*, 157-171, Springer US.

[162] Zarandi, M. H. F., Davari, S., Sisakht, S. A. H. (2013). The large-scale dynamic maximal covering location problem. *Mathematical and Computer Modelling*, 57 (3), 710-719.

[163] Zuhe, S., Neumaier A., Eiermann, M. C. (1990). Solving Minimax Problems by Interval Methods, *BIT*, 30, 742-751.

Biografija autora

Stefan Mišković je rođen 29. decembra 1987. u Čačku. Osnovnu školu "Dr Dragiša Mišović" i Gimnaziju u Čačku je završio kao đak generacije. Na Matematički fakultet u Beogradu, studijski program Računarstvo i informatika, upisao se 2006. godine, a diplomirao je 2010. godine sa prosekom 9.90. Master studije, na studijskom programu Matematika, modul Računarstvo i informatika, završio je 2011. sa prosekom 10.00, odbranivši master rad pod nazivom "Rešavanje dvostepenog problema instalacije neograničenih kapaciteta primenom genetskog algoritma". Kao učenik gimnazije i student Matematičkog fakulteta, učestvovao je na brojnim nacionalnim i međunarodnim takmičenjima iz matematike i programiranja i osvajao nagrade.

Od oktobra 2010. do oktobra 2012. bio je zaposlen na Matematičkom fakultetu u Beogradu, na Katedri za računarstvo i informatiku, kao saradnik u nastavi, a od oktobra 2012. do danas kao asistent na istom fakultetu. Tokom tog perioda držao je vežbe iz predmeta: Uvod u organizaciju računara, Uvod u arhitekturu računara, Mrežno računarstvo i Naučna izračunavanja. Od 2011. godine do danas angažovan je na projektu Ministarstva prosvete, nauke i tehnološkog razvoja pod nazivom "Razvoj novih informaciono-komunikacionih tehnologija, korišćenjem naprednih matematičkih metoda, sa primenama u medicini, energetici, e-upravi, telekomunikacijama i zaštiti nacionalne baštine", broj 044006. Od novembra 2011. do januara 2013. bio je zaposlen sa delom radnog vremena kao istraživač na Matematičkom institutu SANU.

Прилог 1.

Изјава о ауторству

Потписани-а _____

број уписа _____

Изјављујем

да је докторска дисертација под насловом

_____

_____

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

Потпис докторанда

У Београду, _____

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора _____

Број уписа _____

Студијски програм _____

Наслов рада _____

Ментор _____

Потписани _____

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

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

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

Потпис докторанда

У Београду, _____

_____

Прилог 3.

Изјава о коришћењу

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

_____

_____

која је моје ауторско дело.

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

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

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

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

Потпис докторанда

У Београду, _____

_____

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

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

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

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

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

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