Univerzitet u Beogradu

Matematički fakultet

Master rad

Rešavanje lokacijskog problema ograničenih kapaciteta sa modularnim vezama korišćenjem memetskog algoritma

Student:
Miloš Perić

Mentor:
prof. dr Zorica Stanimirović

Beograd, 2015.

Master rad

Rešavanje lokacijskog problema ograničenih kapaciteta sa modularnim vezama korišćenjem memetskog algoritma

Autor:

Miloš Perić

Članovi komisije:

prof. dr Zorica Stanimirović, mentor

prof. dr Miodrag Živković

doc. dr Miroslav Marić

Sadržaj

Predgovor

U ovom radu su predložene metaheuristike tabu pretraživanja, genetskog algoritma i njihova hibridizacija u formi memetskog algoritma za rešavanje jedne varijante lokacijskog problema. Razmatrani problem je poznat pod nazivom Hab lokacijski problem ograničenih kapaciteta sa modularnim vezama (Capacitated hub location problem with modular link capacities - HMLC). Opisani su hab lokacijski problemi, njihova klasifikacija po raznim kriterijumima, osnovne i napredne verzije hab lokacijskih problema. Takođe je navedeno nekoliko primera rešavanja lokacijskih i hab lokacijskih problema primenom memetskog algoritma. Rad sadrži opis tabu pretraživanja i svih njegovih specifičnosti, kao i konkretnu implementaciju tabu pretraživanja za lokacijski problem ograničenog kapaciteta sa modularnim vezama. Takođe je opisan genetski algoritam sa najbitnijim svojstvima i primena na rešavanje razmatranog problema. Zatim je predložena hibridizacija tabu pretrage i genetskog algoritma, u okviru koncepta memetskog algoritma, i naveden je opis konkretne implementacije.

Rad se sastoji od šest poglavlja. U prvom poglavlju su opisani hab lokacijski problemi, njihove osnovne karakteristike, način klasifikacije i najznačajniji predstavnici svake od klasa. U drugom poglavlju su date osnovne postavke memetskog algoritma. U trećem poglavlju opisan je hab lokacijski problem ograničenih kapaciteta sa modularnim raspodelom cena i data je njegova matematička formulacija. Četvrto poglavlje govori o konkretnoj implementaciji algoritma. Peto poglavlje sadrži eksperimentalne rezultate dobijene prilikom rešavanja problema. U zaključku su data bitna zapažanja koja su proizišla tokom izrade celog rada.

Želeo bih da se zahvalim mentoru prof. dr Zorici Stanimirović na svesrdnoj pomoći, korisnim predlozima i velikom strpljenju tokom izrade ovog rada, takođe se zahvaljujem članovima komisije prof. dr Miodragu Živkoviću i doc. dr Miroslavu Mariću na detaljnom čitanju rukopisa.

1. Hab lokacijski problemi

1.1. Definicija hab lokacijskih problema

Neka je G potpun graf G = (I, E), a I je konačan neprazan skup čvorova. Elementi skupa I predstavljaju početna i krajnja odredišta kao i potencijalne habove. Skup dvoelementnih podskupova skupa I je označen sa E i njegovi elementi se zovu grane. Količina protoka ili količina robe koju treba transportovati od čvora $i \in I$ do čvora $j \in J$ je w_{ij}, a d_{ij} rastojanje od čvora $i \in I$ do čvora $j \in J$ koje može (ne obavezno) da zadovoljava nejednakost trougla. Cilj je da neke od ovih tačaka proglasimo habovima, a zatim njima pridružiti čvorove koji nisu hab tako da se minimizuje ili maksimizuje vrednost funkcije cilja pri određenim uslovima.

Troškovi transporta se sastoje od tri komponente: troškovi sakupljanja (collection) od čvora-snabdevača do alociranog haba, troškovi prenosa (transfer) transporta između habova i troškovi raspodele (distribution) od alociranog haba do čvorova-korisnika.

Habovi su čvorovi koje smo na osnovu određenih kriterijuma odabrali i oni predstavljaju centre konsolidacije i kolekcije protoka u mreži između dve lokacije, čvora-snabdevača i čvora korisnika. Preusmeravanjem transporta kroz habove smanjuju se troškovi transporta (komunikacije) između svaka dva čvora. Habovi se koriste da bi se smanjio broj transportnih grana između početnih i krajnjih lokacija. Na primer, potpuno povezana mreža sa k čvorova koja nema habove ima k (k-1) grana koje povezuju polazište i odredište. Međutim, ako je izabran hab koji će međusobno povezati sve ne-hab čvorove, neophodno je samo 2 (k-1) grana koje povezuju polazišta sa odredištima. Habovi su specijalne lokacije koje služe sa preusmeravanje, sortiranje i grupisanje u sistemima sa višestrukim polazištima i odredištima. Umesto da se svaki par polazište-odredište poveže direktno, što bi eksponencijalno povećalo cenu uspostavljanja i održavanja takvog sistema, habovi služe kao tačke gde se koncentriše protok kroz sistem radi umanjena ukupnih transportnih troškova. Roba iz istih polazišta a sa različitim odredištima su udružene na svom putu do haba, pa se zatim kombinuju sa robom koja ima različita polazišta a isto odredište. Ta spajanja se mogu izvršiti od polazišta do haba, od jednog do drugog haba (ako ih u sistemu ima više) i od haba do odredišta.

Habovi nalaze primenu u raznim privrednim oblastima. Čini se da su telekomunikacion sistemi jedno od prvih mesta gde su hubovi našli svoju primenu. Takođe, logistički sistemi, među kojima najviše aviokompanije i poštanske kompanije, se takođe oslanjaju na primenu habova. Danas su se habovi sveprisutni u mnogim privrednim granama kao što je pomorski transport, kurirske službe na lokalnom i globalnom nivou, javni gradski prevoz, međugradski prevoz i elektronski sistemi komunikacija. Najviše proučavani slučajevi gde se mogu primeniti hab lokacijski problemi su avionski transport, aerodromska logistika i transportni sistemi. Može se izvući zaključak da su u zadnjih desetak i više godina hab lokacijski problemi postali veoma zastupljeni u rešavanju problema iz realnog života.

Primena habova ima i potencijalne nedostatke ukoliko se ne primeni adekvatan model koji odgovara situaciji i potrebama korisnika. Na primer, kod aerodroma koji služe kao hab centri, dolazi do velikog nagomilavanja putnika. Ako se na mali prostor smešta veliki broj putnika, njihova udobnost opada. Sa ekonomske strane to je bitan faktor, koji u ekstremnim slučajevima može dovesti do toga da putnici izbegavaju da koriste aerodrome koji su pretrpani. U hab lokacijskoj oblasti, kriterijum za predložene modele najčešće je cena sistema. Međutim, koristi se popust (discount factor) za cenu transporta između dva haba. Štaviše, ne postoji istraživanje o činiocima koji utiču na protok između

habova i čvorova koji nisu habovi. Ovo istraživanje bi bilo važno jer ne postoji garancija da jedna grana između haba i čvora koji nije hab ne bi mogla da ima veći obim saobraćaja nego grana između dva haba. U nekim oblastima u kojima se primenjuju hab lokacijski problemi kao što su avio saobraćaj, sistemi za ekspresnu isporuku pošiljki i ostali globalni sistemi komunikacije i distribucije (komunikacija svemirskim satelitima koji šalju signale velikom broju korisnika koji su raspoređeni na velikoj površini) pretpostavlja se da postojeća postrojenja pokrivaju dovoljno veliki prostor. Stoga, pretpostavke o mrežnim i diskretnim prostorima rešenja više ne važe. Potrebne su drugačije pretpostavke da bi problemi mogli realistično da se sagledaju.

Hab lokacijski problemi sa višestrukim kriterijumima (MCDM-Multi Criteria Decision Making) se još ne proučavaju u dovoljno velikoj meri. Do sada je ciljna funkcija bila u obliku maksimiziranja profita ili minimalizacije troškova. Međutim, u nekim oblastima kao što je aviotransport imamo i druge funkcije cilja kao što je maksimizacija vlasništva u udelu tržišta, kao i maksimizacija ugođaja putnika što je jako teško formulisati. Analizirajući više konfliktnih ciljeva, dolazimo u situaciju da možemo više problema iz stvarnog života na efikasniji način.

1.2. Klasifikacija hab lokacijskih problema

Da bismo klasifikovali hab lokacijske probleme, navešćemo neke pojmove i kriterijume koji su bitni za njihovo proučavanje.

Prostor rešenja se deli na mrežni, diskretni i neprekidni. Kod mrežnog prostora habove biramo iz skupa svih čvorova. Kod diskretnog prostora habove biramo iz skupa nekih određenih čvorova, dok kod neprekidnog prostora habove biramo iz celokupne površine ili sfere. Na primer, u slučaju da želimo da odaberemo aerodrom koji bi bio hab u aerotransportu, možemo da odaberemo bilo koji aerodrom kao hab, što bi bio primer mrežnog prostora rešenja. Ako aerodrom koji biramo kao hab ima neki minimalni broj portala (gates) ili neki drugi uslov, onda je to primer diskretnog prostora rešenja. Primer neprekidnog prostora rešenja bi bilo ako umesto da koristimo neki postojeći aerodrom napravimo novi aerodrom na proizvoljnoj lokaciji.

Kriterijum odlučivanja može biti Min-Max i Min-Sum. Min-Max je kriterijum odlučivanja čiji je cilj da se postigne da maksimalna cena transporta od čvora polazišta do čvora odredišta bude minimizovana. Min-Sum je kriterijum odlučivanja kojim se postiže da ukupna cena određivanja habova u zbiru sa cenom spajanja čvorova koji nisu habovi sa habovima bude minimizovana.

Kriterijum za određivanje broja čvorova može biti egzogeni ili endogeni. Kriterijum po kojem određujemo broj čvorova je egzogeni ako je broj habova koje treba locirati određen unapred uslovima problema. Kriterijum na osnovu kojeg određujemo broj čvorova je endogeni ako broj habova koje treba locirati nije unapred određen već će biti određen kao deo rešenja. Po broju habova problemi se mogu podeliti na probleme sa jednim hab čvorom(single-hub) i probleme sa više hab čvorova (multiple-hub).

Po kapacitetu hab lokacijski problemi mogu da se podele na hab lokacijske probleme bez kapaciteta (uncapacitated, unlimited) i hab lokacijske probleme sa kapacitetom (capacitated, limited). Ograničeni hab lokacisjki problemi, tj. oni sa kapacitetima, mogu imati ograničenu veličinu hab čvora ili se ograničenja mogu odnositi i na grane kojima su povezani čvor snabdevač i hab čvor, hab čvor i čvor korisnik, kao i grane koje povezuju dva hab čvora.

Po načinu alociranja ne-hab čvora nekom hab čvoru hab lokacijski problemi mogu da se podele na jednostruku alokaciju (single allocation scheme) i višestruku alokaciju (multiple allocation scheme). Jednostruka alokacije je vrsta alokacije kod koje je svaki ne-hab čvor pridružen tačno jednom habu.

Transport ide iz jednog ne-hab čvora ka samo jednom habu i transport koji dolazi ka odredištu kreće iz samo jednog haba, iz onog kome je pridružen taj čvor. Višestruka alokacija je alokacija koja dopušta svakom ne-hab čvoru da ima vezu sa jednim ili više hab čvorova. Time se ostvaruje veći broj mogućnosti komunikacije između čvorova koji predstavljaju polazišta i čvorova koji predstavljaju ciljeve. Takođe, tako se može odrediti putanja sa nižom cenom transporta, nego što je cena transporta kod jednostruke alokacije. Kada sistem sa jednostrukom alokacijom pretvorimo u sistem sa višestrukom alokacijom time što omogućimo veći broj habova, transport robe uz pomoć novih habova ima manju cenu nego preko starog (nekada jedinstvenog) haba.

Kategorizacija po ceni povezivanja ne-hab čvorova sa hab čvorovima može biti bez cene, sa fiksnom cenom ili sa promenljivom cenom. Po ceni lociranja hab čvorova hab lokacijski problemi dele se na hab lokacijske probleme bez cene, sa fiksiranom cenom i one sa promenljivom cenom lociranja haba. Kategorizacija na osnovu cene uspostavljanja hab čvorova deli hab lokacijske probleme na one sa fiksnom cenom, promenljivom cenom ili bez cene.

1.3. Osnovni hab lokacijski problemi

U ovom poglavlju biće navedeni neki osnovni problemi iz oblasti hab lokacijskih problema. Prvi koji ćemo opisati su problemi koji se mogu svrstati u grupu probleme hab medijane. Problemi u toj grupi su problemi jednostruke alokacije, koji se dalje dele na probleme neogranicenog kapaciteta (USAHLP i USApHMP) i probleme ograničenog kapaciteta (CSAHLP i CSApHMP), i problemi višestruke alokacije, koji se se takođe mogu podeliti na probleme neograničenog kapaciteta (UMAHLP i UMApHMP) i probleme ograničenog kapaciteta (CMAHLP i CMApHMP).

Cilj problema hab medijane je da minimiziraju sumu transportnih troškova za svaki par polazište-odredište preko odgovarajućih habova. Ako je broj habova unapred zadat i fiksan (neka je to broj p), onda je to skup problema p-hab medijane. Ako broj habova nije unapred određen, potrebno je da sistem ima fiksne troškove za uspostavljanje habova.

1.3.1. Hab lokacijski problem neograničenog kapaciteta sa jednostrukom alokacijom

Kod hab lokacijskih problema neograničenog kapaciteta sa jednostrukom alokacijom (Uncapacitated single allocation hub location problem) potrebno locirati habove i njima pridružiti ne-hab čvorove tako da je suma transportnih troškova u mreži čvorova minimalna. Ovaj problem je NP-težak, a to je pokazao O’Kelly u radu [85], isti autor je ovaj problem prvi put formulisao. Broj habova nije unapred definisan, problem je jednostruke alokacije što znači da svaki ne-hab čvor može biti pridružen samo jednom habu, postoje fiksni troškovi za uspostavljanje svakog pojedinačnog haba.

Za rešavanje ovog problema u radu [1] Abdinnour-Helm 1998. godine predlaže tabu pretraživanje i genetski algoritam. Konkretno, prvo se koristi genetski algoritam da bi se našli habovi, zatim se koristi TS (Tabu Search) da bi se našla optimalna alokacija. Isti naučnik je 1998. godine kao dodatno rešenje u drugom radu [2] kao rešenje predložio genetski algoritam i algoritam Branch-and-bound. Labbe, Yaman i Gordiny u radu [62] za rešenje problema predlažu algoritam Branch-and-cut. Egzaktna metoda Branch-and-cut koja služi rešavanju USAHLP problema je opisana u [62]. Algoritam koji je tu predložen prvo preprocesira problem i proverava njegovu dopustivost i, u slučaju ako je potrebno, dodaje nove uslove na osnovu dobijenih donjih granica za količinu protoka i broj habova

koje treba locirati. Način da se problem USAHLP reši uz pomoć tabu pretraživanja i HGA (Hibridni genetski algoritam) je naveden u radu [1]. Hibridni genetski algoritam ima bolje performanse od čistog genetskog algoritma. Eksperimentalni rezultati su pokazali da je, u ovom slučaju, genetski algoritam dobar za diversifikaciju pretrage a da je tabu pretraživanje dobar za lokalnu pretragu prostora rešenja. Genetski algoritam se nije često koristio kao moguć pristup rešenju lokacijskih problema, već kao rešenje za probleme operacionih istraživanja kao što su Scheduling problem i TSP (Traveling salesman problem). U radu [104] su upoređena rešenja koja su dobijena primenom klasičnog genetskog algoritma i kombinacije genetskog algoritma i tabu pretrage (Genetic algorithm & Tabu search - GATS). GATS je metoda za rešavanja problema USAHLP koja se od klasičnog GA razlikuje po tome što koristi efektivnijie kodirane jedinke i prilagođen operater ukrštanja.

1.3.2. Problem p-hab medijane neograničenog kapaciteta jednostruke alokacije

Problem neograničenog kapaciteta p-hab medijane sa jednostrukom alokacijom (Uncapacitated single allocation p-hub median problem - USApHMP) je skoro isti problem kao USAHLP, razlikuje se po tome što je broj habova fiksiran. Dobijamo problem p-hab medijane neograničenog kapaciteta jednostruke alokacije (Uncapacitated single allocation p-hub median problem - USApHMP).

USApHMP je problem neograničenog kapaciteta, kod koga je neophodno iz skupa postojećih čvorova locirati p habova i svaki par snabdevač-korisnik pridružiti jednom habu. Ovi problemi su NP-teški problemi [65]. Neki od algoritama koji su korišćeni za rešenje ovog problema su simulirano kaljenje (Simulated annelaling - SA) u radu [96] u kome je dobijena gornja granica za poboljšanje metode Grananja i ograničavanja (Branch and Bound - BaB). U radu [97] su upoređene metode simulirano kaljenje i tabu pretraživanje po kriterijumima kvaliteta rešenja i vremena izračunavanja, ali samo na sistemima sa ograničenim dimenzijama (sistemi sa 50 ili manje čvorova). Način na koji je tabu pretraživanje iskorišćeno za rešavanje ovog problema kao i način na koji je rešen uz pomoć heuristike pohlepne procedure slučajnog prilagođavanja (Greedy Randomized Adaptive Search Procedure - GRASP) gde se ne-hab čvorovi pridružuju najbližem habu dao je Klincewicz u [53]. Takođe je korišćena metoda ponovnog povezivanja puteva (Path Relinking - PR) kojoj je glavno svojstvo da kombinuje rešenja koristeći prostore susedstava [87].

Jedan od novijih načina da se reši ovaj problem je algoritam generalnog promenljivog pretraživanja susedstva (General variable neighborhood search - GVNS) u kojoj su korišćena tri susedstva i efikasno ažuriran skup podataka za izračunavanje ukupnog protoka u celoj mreži [44]. Predložena je ugnježdena strategija u projektovanju determinističkih promenljivih susedstava iz lokalne pretrage. Ovaj način je bolji od svih do sada korišćenih heuristika po pitanju kvaliteta rešenja i složenosti izračunavanja. Ernst i Krishnamoorthy u radu [28] koriste metodu najkraćeg puta za nalaženje svih mogućih lokacija habova. Algoritam je polinomijalan po broju čvorova n i eksponencijalan je po broju habova p. To znači da daje egzaktna rešenja u nekom razumnom vremenu ako imamo mali broj habova (ako je broj habova manji ili jednak pet). Ako su u pitanju problemi većih dimenzija metoda najkraćeg puta se kombinuje sa algoritmom BaB.

1.3.3. Hab lokacijski problem ograničenih kapaciteta sa jednostrukim alokacijama

Cilj Hab lokacijskog problema ograničenih kapaciteta sa jednostrukim alokacijama (Capacitated single allocation hub location problem - CSAHLP) je minimizovati sumu troškova transporta kroz mrežu kao i fiksirane troškove uspostavljanja hab lokacija. Razlika između ovog problema i problema

USAHLP je što ovde postoji ograničenje kapaciteta. Zbog toga je ovaj problem složeniji za rešavanje, ali je bitniji jer će u stvarnosti retko kada postojati neograničeni kapaciteti. Problem CSAHLP je NP-težak problem, pošto je USAHLP dokazan kao NP-težak problem (Kara and Tansel 1998).

Problem CSAHLP razmatran je u radu [29]. U njihovom radu predloženi algoritmi su dva heuristička algoritma koja se zasnivaju na simuliranom kaljenju i slučajnom silaženju (Random descent - RDH). Algoritmi SA i RDH su hibridizovani sa algoritmom Branch-and-bound koji je imao optimalna rešenja za probleme sa manjim brojem čvorova (pedeset ili manje čvorova). Kada se broj čvorova poveća na veći broj, što je zapravo slučaj koji odgovara realnosti, algoritmi RDH i SA su dali rešenja u zadovoljavajućem vremenu. Ovaj problem zadaje poteškoće kod instanci velikih dimenzija. Rešenje tog problema predloženo je u radu [16] a to je primena Lagranžove relaksacije, koja je poboljšana redukovanim testovima. Njom je izračunata čvrsta gornja i donja granica za instace velikih dimenzija za problem CSAHLP. Egzaktna metoda Branch-and-cut koja služi rešavanju CSAHLP problema je opisana u [62].

Ne može se zanemariti kapacitet haba u odabiru rešenja za problem CSAHLP. Do sada su razmatrana rešenja problema gde svi habovi imaju isti kapacitet. U radu [100] razmotreno je rešenje ovog potproblema sa habovima različitih kapaciteta, koji se zove hab lokacijski problem jednostruke alokacije ograničenog kapaciteta sa višestrukim nivoima kapaciteta (Capacitated single assigment hub location hub location problem with multiple capacity level - CSAHLPM). Zbog ove varijacije, fiksni troškovi za uspostavljanje habova se određuju na osnovu njegovog kapaciteta.

Vraćamo se rešenjima za klasični CSAHLP. Predlog genetskog algoritma za rešavanje ovog problema prikazan je u radu [100]. Sortiranjem hab lokacija u neopadajući redosled na osnovu njihovog rastojanja za svaki ne-hab čvor postiže se da se GA usmerava u obećavajuće regione. Za male instance problema genetski algoritam je jako koristan jer nalazi visoko kvalitetna rešenja za kratki vremenski period. Uz to, genetski algoritam se pokazao kao korisna metaheuristika za rešavanje CSAHLP problema prilikom rešavanja instanci većih dimenzija.

U radu [21] predložena su dva modela sa dvostrukim kriterijumom (bi-criteria) za rešavanje problema CSAHLP. Prvim kriterijumom se minimizuje ukupno vreme servisiranja, dok se drugim kriterijumom minimizuje maksimalno servisno vreme habova. Novim metodama omogućeno je da se jedan kriterijum poboljša, a da se pritom drugi kriterijum ne promeni u prevelikoj meri.

1.3.4. Problem p-hab medijane ograničenih kapaciteta jednostruke alokacije

Problem p-hab medijane ograničenih kapaciteta jednostruke alokacije (Capacitated single allocation p-hub median problem - CSApHMP) ima jednostruku alokaciju, što znači da se sav protok iz polazišta mora usmeriti na jedan hab, odnosno da će iz jednog haba ići protok u sva odredišta. Količina protoka koji može biti primljen u hab je ograničen i broj habova koji je potrebno uspostaviti je unapred poznat. Cilj CSApHMP je minimizacija ukupnih transportnih troškova i trošak eventualnih fiksnih troškova za uspostavljanje hab centara. CSApHMP je NP-težak jer je njegov potproblem sa neograničenim kapacitetima, USApHMP, NP-težak [65].

Za rešavanje problema CSApHMP predložene su dve verzije genetskog algoritma [99]. Rešenje je realizovano genetskim operatorima koji čuvaju korektnost jedinki tako što ne menjaju broj uspostavljenih habova i ne menjaju ograničenja kapaciteta habova. Još jedno rešenje problema CSApHMP dato je u [88]. U tom radu je primenjena evolutivna metoda ponovnog povezivanja puteva (Path Relinking - PR), algoritam pohlepne procedure slučajnog prilagođavanja (Greedy randomize search procedure - GRASP). Međutim, rezultati su nestandardni za ovaj tip problema, dalja objašnjenja su u radu[99]. U radu [4] Aykin je predstavio problem CSApHMP sa fiksnim cenama gde habovi imaju

ograničene kapacitete. On je formulisao taj problem tako da su direktne veze između dva ne-hab čvora moguće. Predložen je algoritam grananja i ograničavanja gde su donje granice dobijene primenom Lagranžove relaksacije, koja je rešena uz pomoć optimizacije subgradijenta (subgradient optimization). U radu [29] isti autor analizira problem CSApHMP sa fiksiranim cenama i datim brojem habova. Upoređene su dva načina povezivanja habova koje je nazvao striktni i nestriktni (direktne veze su dozvoljene). Predložen je algoritam enumeracija i algoritam Simuliranog kaljenja (Simulated Annealing - SA).

1.3.5. Problem neograničene višestruke alokacije pozicije habova

Kod problema neograničene višestruke alokacije pozicije habova (Uncapacitated multiple allocation hub location problem - UMAHLP) ne postoje ograničenja na kapacitet habova, broj habova nije unapred zadat i svaki ne-hab čvor može biti pridružen većem broju habova. Broj neophodnih habova se određuje na osnovu toga kakav model problema se rešava. Lociranje tih habova dovodi do fiksnih troškova. Cilj je odrediti optimalne lokacije habova i alokacije ne-hab čvorova uspostavljenim habovima tako da se minimizuje suma troškova transporta i fiksnih troškova.

UMAHLP je prvi formulisao Campbel (1994). U radu [85] već je dokazano da je problem UMAHLP NP-teški problem. Problem UMAHLP je u radu [12] rešavan metodom Branch-and-bound. Isti problem je u radu [14] rešavan uz pomoć algoritma efektivne heuristike (effective heuristic). Efektivna heuristika se zasniva na algoritmu simuliranog kaljenja i tabu liste. U radu [68] je problem UMAHLP rešen tako što je uvedena nova formulacija kojom je omogućeno da se hab poseti jednom ili dva puta, uz napomenu da cene troškova ne moraju da zadovoljavaju nejednakost trougla.

U radu [71] Majer i Vagner su predložili da se problem UMAHLP reši algoritmom HabLokator (HubLocator algorithm). Ovaj algoritam pripada grupi algoritama grananja i ograničavnja. Donja granica se dobija algoritmom dual ascent, dok se gornja granica dobija putem jednostavne heurističke procedure. Ovim algoritmom se u razumnom vremenu mogu rešiti problemi koji imaju do četrdeset čvorova ($n <= 40$).

Problem UMAHLP je rešen algoritmom grananja i ograničavnja, rezultati i implementacija su prikazani u [12]. Prvi korak podrazumeva da se konstruiše heuristički metod koji se zasniva na metodu dual ascent, a koji je pojačan specijalnim sabrutinama i pruža dobru donju granicu čvorovima stabla grananja. Ova heuristika daje veliki broj (preko 67%) optimalnih rešenja za instance koje nemaju više od 120 čvorova ($n \leq 120$). Za dobijanje egzaktnog rešenja UMAHLP problema predložen je algoritam Branch-and-bound koji se zasniva na metodu dual ascent i metodu dual adjustment [54].

U radu [14] je za potrebe nalaženja optimalnog broja habova u problemu UMAHLP korišćena efektivna heuristika (effective heuristic) koja je bazirana na metodi simuliranog kaljenja i tabu liste. Testiranja su pokazala da se primenjivanjem gornje granice broja habova, koja je dobijena heuristikom, mogu dobijati optimalna rešenja za instance koje su korišćene u tom članku.

1.4. Napredni hab lokacijski problemi

Ovde će biti razmotreni neki hab lokacijski problemi koji se mogu smatrati naprednijim u odnosu na ranije navedene hab lokacijske probleme. Zbog sve veće popularnosti hab lokacijskih problema nastaju nove varijante ovih problema koje na realističniji način predstavljaju probleme iz industrije, telekomunikacije, transportnih službi i saobraćaja. U ovoj sekciji će biti navedeni samo neki od tih problema jer bi bilo nemoguće sve te probleme nabrojati [3].

1.4.1. p-HLP sa fiksnim cenama veze

Pošto se ne-hab čvorovi moraju dodeliti hab-čvoru, može se razmatrati uvođenje fiksne cene svih veza između ne-hab čvorova i hab-čvorova. Međutim, da bi povezali ne-hab čvor i hab čvor, fiksna cena uspostavljanja te veze se mora uvesti. U radu [11] je predložen problem p-HLP sa fiksnim cenama veze (p-HLP with fixed link cost), kod koga je osnovni model p-HLP proširen uvodjenjem fiksnih troškova povezivanja ne-hab čvorova i hab čvorova. Ulazne informacije i izlazne informacije su slične kao i kod lokacijskog problema p-hab medijane.

1.4.2. Hab lokacijski problem minimalnog protoka na vezama

Umesto uslovljavanja da svaki ne-hab čvor treba da bude alociran jednom hab čvoru, nekad je bolje specificirati da protok između ne-hab čvora i hab čvora mora biti veći ili jednak nekom minimumalnoj količini protoka. Campbell u radu [11] predstavlja hab lokacijski problem minimalnog protoka na vezama (Minimum-value flow on links problem), koji je sličan problemu p-HLP. Ovaj problem se analogno može formulisati kao p-HLP sa minimalnom količinom protoka na svakoj konekciji između ne-hab čvora i hab čvora.

1.4.3. p-HLP sa višestrukim funkcijama cilja

U radu [21] autori predlažu p-HLP sa višestrukim funkcijama cilja (Multi-objective p-HLP). Prva funkcija cilja minimizuje ukupne troškove protoka, dok druga funkcija cilja minimizuje maksimalno vreme koje je potrebno hab čvoru da obradi protok (npr. minimizuje se maksimalno vreme servisiranja hab čvorova). U ovom problemu svaki ne-hab čvor je alociran samo jednom hab čvoru. U ovom problemu, kriteriji funkcija cilja su Mini-Sum i Min-Max, prostor rešenja je mrežni (skup svih čvorova u sistemu), habovi su kompletno povezani i svaki ne-hab čvor je povezan sa jednim hab čvorom. Broj habova koji se alocira je egzogeni (unapred je poznat postavkom problema). Protok između dva ne-hab čvora mora proći kroz najmanje jedan a najviše dva hab čvora. Habovi su neograničenih kapaciteta, promenljive odlučivanja su binarne i fiksne cene uspostavljanja habova nisu razmatrane u ovom modelu problema**.**

1.4.4. Hab lokacijski problemi sa mrežom strukture zvezde

Kod hab lokacijskih problema sa mrežom strukture zvezde (HLP with star network structure)

polazi se od skupa čvorova sa centralnim hab čvorom. Cilj je izabrati p hab čvorova koji su direktnim vezama povezani sa centralnim hab čvorom. Uz to, svaki ne-hab čvor mora biti povezan sa hab čvorom. Dobijena mreža habova će biti zvezda, tj. mreža sa strukturom zvezde. U radu [110] autor predlaže novu formulaciju problema p-hab medijane sa strukturom zvezde gde bi ukupna cena biranja ograničenih veza bila minimizovana. U radu [111] je razvijena formulacija lokacijskog problema p-hab medijane sa ograničenim dužinama veza, u kojoj je ukupna cena rutiranja bila minimizovana od strane ograničenja koja su se odnosila na dužinu veza između čvorova.

1.4.5. Lokacijski p-hab problem maksimalnog pokrivanja

Lokacijski p-hab problem maksimalnog pokrivanja (p-Hub maximal covering location problem) je poseban slučaj lokacijskog problema hab pokrivanja. Predložena formulacija ovog problema je analogna formulaciji lokacijskog problema p-hab medijane, jer se broj habova koje treba alocirati određuje egzogeno, tj. unapred je poznat na osnovu postavke problema. U ovom problemu se ne razmatra fiksna cena za uspostavljanje habova. U radovima [27] i [107] dokazano je da su problemi ovog tipa NP-teški. U radu [51] lokacijski p-hab problem maksimalnog pokrivanja je rešen metodom dvostruke heuristike. Ciljna funkcija prve heurisike računa fiksnu cenu uspostavljanja habova i heuristika daje egzaktna rešenja. Ciljnu funkciju prve heuristike treba minimizovati. Ciljna funkcija druge heuristike maksimizuje protok kroz mrežu čvorova. Drugom heuristikom se dobijaju rešenja po kvalitetu jako bliska egzaktnim rešenjima. U radu [26] Drezner predlaže algoritam grube sile koji ima eksponencijalnu složenost po broju čvora i habova.

Postoji još mnogo varijanti hab lokacijskih problema. Zbog široke primene u praksi broj hab lokacijskih problema se stalno uvećava, zbog toga je u ovoj sekciji opisano samo nekoliko tipova hab lokacijskih problema. Najskoriji pregled hab lokacijksih modela moze se naći u [17].

2. Memetski algoritmi

2.1. Osnove memetskog algoritma

Posmatrano sa stanovišta izračunljivosti, problemi kombinatorne optimizacije se mogu karakterisati različitom složenošću.

Analizom rasta potrošnje resursa od strane algoritma u zavisnosti od veličine instance problema, moguće je doći do klasifikacije problema po složenosti. Većina problema kombinatorne optimizacije je NP-kompletno, tj. mogu da se reše za polinomijalno vreme uz pomoć nedeterminističke Tjuringove mašine. Klasa P problema je podskup klase NP problema i, iako to još nije dokazano, rasprostranjeno je uverenje da je P pravi podskup (proper subset) od NP, što dalje dovodi do zaključka $P \neq NP$. Dalje zaključujemo da ne postoji efikasan algoritam koji može da reši problem u polinomijalnom vremenu. Za puno praktičnih problema iz industrije se teško može dobiti rešenje koje je približno optimalnom, tj. ne postoji efikasan algoritam koji je sposoban da da rešenje čiji kvalitet ne odstupa više od optimalnog nego za određeni stepen. U radu [106] su opisane različite klase složenosti.

Kada se ne može napraviti pretpostavka za određeni optimizacioni problem, konstruiše se uopšteni algoritam za rešavanje tog problema, tj. za nalaženje optimuma ili barem za nalaženje rešenja visokog kvaliteta. Ti uopšteni algoritmi se zovu metaheuristike. Reč metaheuristika je grčkog porekla, konkretno potiče od reči "μετα" i "ευρισχω", što znači "tražiti izvan" ili "nakon pretrage". To zapravo znači da se pretraga može obavljati na apstraktnom nivou neke procedure pretraživanja.

Metaheuristike su se razvijale tokom prethodnih decenija u korak sa razvojem kompjuterskog hardvera, pa danas postoji veliki broj različitih tipova metaheuristika za razne probleme. Neke metaheurisitke su inspirisane prirodnim pojavama kao što su prinicipi evolucije, razni fizicki fenomeni i ponašanje životinja. Izdvajamo memetski algoritam (Memetic Algorithms - MA) kao jako uspešan algoritam koji se razvio u zadnje dve decenije i koji iz godine u godinu postaje sve bitniji algoritam po pitanju rešavanja složenih optimizacionih problema. U ovom segmentu ćemo opisati sam algoritam i detalje implementacije memetskog algoritma i različite verzije MA koje se primenjuju na optimizacione probleme pod raznim okolnostima.

Memetski algoritmi su populacijske metaheuristike čiju glavnu strukturu predstavlja populacijski (najčešće evolutivni) algoritam koji u ciklusima pokreće skup lokalnih pretraga [42]. Najranija implementacija memetskog algoritma opisana je u radu [82] gde je primenjena za rešenje problema putujućeg prodavca (Travelling Salesman Problem - TSP). Najranija detaljna definica algoritma prikazana je u radu [83]. Koncept mema je pozajmljen iz filozofije i predstavlja jedinicu prenosa kulture. Drugim rečima, složene ideje se mogu razložiti na memee koji se mogu raširiti i mutirati u okviru populacije, i na taj način kultura evoluira i teži napretku. Dobre ideje teže da opstanu i da se prošire u okviru društva dok slabe ideje nisu odabrane i teže nestajanju. Pomoću prethodne metafore zaključujemo da su ideje operatori pretrage, najkvalitetnije će biti upotrebljene a neadekvatne će verovatno nestati.

Pošto memetski algoritmi predstavljaju široku klasu algoritama koji kombinuju razne algoritamske komponente, za svaki problem je potreba isključivo odgovarajuća kombinacija. Memetski algoritmi su vremenom postali jako popularni i u nekim slučajevim i neophodni prilikom rešavanja problema u računarstvu, inženjerstvu i drugim oblasima prakse i istraživanja.

U opštem slučaju memetski algoritam se definiše kao algoritam koji se sastoji od jedne ili više procedura lokalne pretrage koja deluje nad skupom rešenja (populacijom) *pop* za koji važi $|pop| \geq 2$ i

koja se u periodičnim epizodama rekombinuje. Opšti oblik memetskog algoritma prikazan je Algoritmom 1.

Algoritam 1. Osnovni memetski algoritam

1. Osnovni Memetski Algoritam (Problem *P*, parametri *par*)
2. *pop* ← *Inicijalizacija(par, P)*
3. dok nije zadovoljen kriterijum zaustavljanja ponavljati sledeće
4. *novapop1* ← *Rekombinuj*(*pop, par, P)*
5. *novapop2* ← *Poboljšaj(novapop1, par, P)*
6. *pop* ← *Takmičenje(pop, novapop2)*
7. ako nema poboljšanja
8. *pop* ← *Restartuj(pop, par)*
9. izlazak iz upita
10. izlazak iz petlje nakon ispunjenog kriterijuma zaustavljanja
11. *VratitiNtiNajbolji(pop, 1)*

Procedura *Inicijalizacija* je zadužena za kreiranje populacije početnih rešenja. Za razliku od većine evolutivnih algoritama koji uglavnom generišu proizvoljnih |*pop*| početnih rešenja, memetski algoritam pokušava da koristi početna rešenja većeg kvaliteta. To se postiže korišćenjem naprednijeg nivoa za generisanje kvalitetnih rešenja ili korišćenjem lokalne pretrage da bi popravila proizvoljna rešenja. Ovaj pristup poboljšanja kvaliteta početne populacije će biti prikazan Algoritmom 2.

Algoritam 2. Ubacivanje početnih rešenja visokog kvaliteta u inicijalnu populaciju

1. *Inicijalizacija*(parametri *par*, problem *P*)
2. *pop* je prazan skup
3. for *j* ← od 1 do *par.popsize*
4. *i* ← ProizvoljnoRešenje(*P*)
5. *i* ← LokalnaPretraga *(i, par, P)*
6. *pop* ← *pop* ∪ *{i}*
7. kraj for petlje
8. vratiti *pop*

Kao kriterijum završetka se najčešće koristi maksimalan broj iteracija ili maksimalno vreme izvršavanja algoritma. Takođe se može koristiti i maksimalan broj iteracija bez popravke najboljeg rešenja ili se algoritam zaustavlja ako je dostignuto optimalno rešenje (ukoliko je ono poznato). Ipak, u praksi se pokazalo da najbolje rezultate daje kombinacija više kriterijuma zaustavljanja.Procedure *Rekombinuj* i *Poboljšaj* iz prvog algoritma čine srž memetskog algoritma. Implementacija prve metode se najčešće se zasniva na dva operatora za selekciju rešenja iz populacije i njegovog rekombinovanja. Ova procedura se može proširiti tako da koristi razne operatore. *Poboljšaj* procedura primenjuje

lokalnu pretragu na rešenja u okviru populacije. U opštem slučaju lokalna pretraga se može predstaviti kao unarni operator, pa je mogla biti u okviru procedure *Rekombinuj*. Međutim, pošto lokalna pretraga igra važnu ulogu u memetskom algoritmu, njena primena zahteva zasebnu proceduru. Prilikom primene lokalne pretrage važno je se vodi računa o tome na koja rešenja se primenjuje, koliko često se primenjuje, koliko dugo.

Procedura *Rekombinuj* se koristi da se rekonstruše trenutna populacija koristeći staru populaciju *pop* i nova generisana populacija *novapop2*. U radovima [92, 95] predstavljene su dve glavne mogućnosti da se realizuje ta rekonstrukcija: plus strategija i strategija zapete. Strategija zapete ima manju tendenciju stagnacije, prilikom koje je odnos |*novapop*| i |*pop*| najčešće šest. Ova strategija može biti skupa za izračunavanje ako je funkcija kvaliteta složena i vremenski zahtevna. U tom slučaju koristi se plus strategija sa malom kardinalnošću nove populacije [108]. Ova strategija obično daje bržu konvergenciju ka rešenjima visokog kvaliteta, ali se mora voditi računa o preuranjenim konvergencijama ka područijima pretraživačkog prostora.

Odluka o restartovanju populacije se donosi na osnovu nekoliko faktora. Prvo, mora se utvrditi da li je populacija degradirana koristeći meru raznovrsnosti informacije kao što je, npr. Šenonova entropija (Shannon`s entropy) [23]. Ako je utvrđeno da je populacija degenerisana pokreće se restartovanje. Restartovanje populacije može biti implementirano na razne načine. Uobičajeni pristup je da se zadrži deo postojeće populacije i da se generišu nova rešenja, proizvoljno ili korišćenjem neke heuristike. Ova metoda je poznata kao metoda proizvoljnog imigranta (random-immigrant strategy) [15]. Druga mogućnost bi bila da se primeni mutacija velikih razmera tako da se populacija dovoljno promeni.

Uopšteni memetski algoritam iz prethodnog odeljka se mora koristiti sa precizno određenim komponentama da bi mogao da se koristi za rešavanja konkretnog optimizacionog problema. Memetski algoritmi se obično implementiraju kao evolutivni algoritmi pojačani komponentom lokalne pretrage. Najspecifičnije odluke vezane za implementaciju memetskog algoritma odnose se na komponente lokalne pretrage. Ne misli se samo na parametrizaciju, već i na sam način kako se lokalna pretrage implementira. Problem implementacija lokalne pretrage predstavljen je u radu autora Merc i Frajsleben [33]. Oni u svom radu razmatraju korišćenje heuristike [63] koja je lokalna pretraga visokog intenziteta. Primećeno je da je prosečna udaljenost između lokalnih optimuma slična kao i prosečna udaljenost lokalnih optimuma od globalnog optimuma. Uvodi se operator očuvanja udaljenosti (distance-preserving crossover operator, DPX) koji generiše potomke čija udaljenost od roditelja je ista kao međusobna udaljenost roditelja.

Nakon što se odabere vrsta lokalne pretrage, potrebno je uraditi adekvatan izbor parametara memetskog algoritma. Na primer, neophodno je odlučiti koliko će se puta lokalna pretraga primeniti, kako izabrati rešenje koje će biti podvrgnuto lokalnoj pretrazi i koliko dugo treba da deluje poboljšavanje odabranog rešenja. Ako se pogrešno uradi parametrizovanje, dokazano je da problem može da se transformiše iz lakog rešenja do nepolinomijalnog rešenja [60, 102]. Verovatnoća da se lokalna pretraga primeni zavisi isključivo od problema koji se razmatra. Zbog toga se uvode prilagodljive i samoprilagodljive metode kako bi algoritam odlučio šta je najbolje.

Na koju individuu će biti primenjena lokalna pretraga, najčešće se odlučuje proizvoljnim izborom jedinke ili na osnovu funkcije kvaliteta. U slučaju da se primenjuje funkcija kvaliteta, samo najbolje jedinke će biti odabrane za lokalnu pretragu. U radu [84] autor predstavlja strategiju u kojoj je populacija sortirana i podeljena na n nivoa, gde n predstavlja broj primena lokalne pretrage. Proizvoljno se bira jedna individua po nivou. Takva strategija se može primeniti na struktuirane memetske algoritme u kojima su dostupni slojevi sortirani na osnovu kvaliteta [8, 10, 32, 72, 73].

Memetski algoritmi su pragmatični, interdisciplinarni optimizacioni pristup koji se pojavio u zadnjih dvadeset godina. Danas je to jedan od najrasprostranjenijih pristupa prilikom rešavanja

optimizacinoih problema. Neke od oblasti u kojima se primenjuju memetski algoritmi su mašinsko učenje, planiranje, problemi raspoređivanja [22], lokacijsko planiranje [69, 74], bioinformatika [42], elektronika [77], razne inženjerske grane [78], telekomunikacije [79], ekonomija [80], fizika [81].

U literaturi su pored osnovne varijante memetskog algoritma, predloženi i neki napredniji koncepti za rešavanje specifičnih problema. Jedan od njih je memetski algoritam za rešavanje problema višekriterijumske optimizacije (Multi Objective Memetic Algorithm, MOMA). MOMA se koristi prilikom rešavanja problema čiji višestruki kriterijumi su delimično međusobno konfliktni. MOMA algoritmi se mogu podeliti na dve klase. Prva klasa se zasniva na skalabilnom pristupu. Skalabilni pristup se zasniva na mehanizmima agregacije uz pomoć kojih se svi ciljevi kombinuju u skalarnu vrednost [45, 46, 48, 49]. Druga klasa MOMA koristi pareto-dominaciju (Pareto-dominance). Pareto-dominacija upravlja prelaskom sa jednog na drugo rešenje koja su u međusobnom susedstvu [55, 56].

Prilagodljivi memetski algoritmi su još jedna vrsta sofisticiranih memetskih algoritama. Odluke vezane za parametrizaciju su suštinske za implementaciju efikasnog memetskog algoritma. U radovima [5, 66, 75, 76] su predstavljene ideje da se algoritmu prepusti nalaženje optimalnih vrednosti parametara. Pojam meta-Lamarkijanovo učenje (meta-Lamarckian learning) je metoda obeležavanja strategija uz pomoć kojih algoritam uči kako da odabere odgovarajući operator lokalne pretrage iz skupa ponuđenih operatora. Sledeći korak unapred napravljen je kod takozvanog višestrukog memetskog algoritma (multi-memetic algorithms). Kod višestrukih memetskih algoritama svako rešenje sadrži gen koji ukazuje na to koja lokalna pretraga će biti primenjena na njega. Najčešći načini implementacije su parametrizacija uopštene lokalne pretrage ili korišćenjem gramatike da bi se definisao novi operator [57, 58, 59]. Na nekim višim nivoima, povećava se kvalitet rešenja simultano sa poboljšanjem lokalne pretrage.

Postoji rastući trend kombinovanja memetskih algoritama sa celokupnim tehnikama optimizovanja kao što je grananja i ograničavanja i grananja i odsecanja. Egzaktna tehnika se može koristiti kao operacija unutar memetskog algoritma, što je pokazano u radovima [36, 91]. U radu [52] prikazano je kako se algoritami grananja i ograničavanja i grananja i odsecanja mogu iskoristi kao metode postprocesiranja memetskog algoritma, dok je u radovima [34, 35, 90] prikazano kako se memetski algoritam može paralelnom kombinacijom koristiti sa algoritmima drugih tipova.

2.2. Primena memetskih algoritama za rešavanje lokacijskih problema

U ovoj sekciji će biti predstavljene neke od mnogobrojnih primena memetskog algoritma. Ovaj pregled primena memetskog algoritma nije sveobuhvatan zbog toga što se svakodnevno pronalaze novi načini primene memetskog algoritma za rešavanje problema kombinatorne optimizacije. Memetski algoritmi su sa velikim uspehom korišćeni prilikom rešavanja problema particionisanja grafova (Graph Partitioning - GP), minimalnog particionisanja brojeva (Min Number Partitioning - MNP), maksimalnog nezavisnog skupa (Max Independent Set - MIS), pokrivanja skupa (Set Covering - SC), problemi raspoređivanja mašina i problemi paralelnog raspoređivanja (Single Machine Scheduling - SMS i Parallel Machine Scheduling- PMS), problem putujućeg trgovca (Travelling Salesman Problem - TSP) kao i mnoge druge probleme.

Memetski algoritam je primenjem za rešavanje hab lokacijskog problema neograničenih kapaciteta (USAHLP) u radu [1]. Memetski algoritam iz [1] je implementiran kao kombinacija genetskog algoritma i tabu pretrage. Takva implementacija memetskog algoritma daje mnogo

kvalitetnije rezultate u poređenju sa rezultatima koje daje običan genetski algoritam. Ovaj algoritam je dao bolje rezultate od bilo kog drugog algoritma koji je predložen do tada.

Problem raspoređivanja fabrika (Facilty Layout Problem) rešen je pomoću memetskog algoritma u radu [94]. Genetski algoritam u okviru memetskog algoritma implementiran je uz pomoć drvoidne strukture podataka. Fabrike su uparene koristeći iterativno uparivanje na osnovu količine protoka između dve fabrike ili na osnovu sume protoka između fabrika koje čine meta-blok. Koriste se samo dve operacije mutacije koje razmenjuju čvorove ili menjaju drvoidnu strukturu koja se koristi.

Diaz u [25] kao metodu za rešavanje problema dinamičkih lokacija predlaže interaktivni memetski algoritam sa višestrukim ciljevima. Algoritam radi kao standardni memetski algoritam sa jedinstvenim ciljem, međutim i skup svih potencijalnih pareto optimalnih rešenja može biti razmatran. Algoritam koristi tehniku vrućeg starta (hot start technique). Kada se promene paramatri na osnovu kojih se donose odluke, trenutna populacija je dalje optimizovana novom funkcijom agregacije.

Bitan problem u mrežama mobilne komunikacije je problem upravljanja lokalnih područja (Location Area Management - LAM). U radu [50] je predložen memetski algoritam čija lokalna pretraga generiše početnu populaciju i koja takođe obavlja funkciju operatora mutacije evolutivnog dela memetskog algoritma.

Memetski algoritam je takođe predložen za rešavanje problema USAHLP u radu [70]. U okviru evolutivnog algoritma su implementirane dve različite vrste lokalne pretrage. Eksperimentalnim rezultatima dokazano je da je memetski algoritam bolji od drugih algoritama koji su korišćeni za ovaj problem. Za neke instance velikih dimenzije (do 200 čvorova) algoritam je uspeo da postigne bolji rezultat nego što je to ranije bio slučaj. Na instancama koje imaju izmedju 300 i 400 čvorova, primena memetskog algoritma je dovela do značajnog poboljšanja kvaliteta rešenja i vremena izračuvanja.

3. Hab lokacijski problem ograničenih kapaciteta sa modularnim vezama

3.1. Opis problema

Optimizacioni problemi koji su vezani za dizajn i instalaciju telekomunikacionih mreža dobijaju na važnosti zbog sve većeg prisustva telekomunikacionih sistema u našim životima. Instalacija telekomunikacionih mreža je skupa, pa je zbog toga nastala potreba za dodatnim korakom u projektovanju tih mreža. U pitanju je faza optimizacije čiji je cilj da se minimizuje cena celokupnog procesa.

Problem koji se razmatra u ovom radu motivisan je činjenicom da u mnogim realnim situacijama, cena veze između snabdevača i korisnika je mnogo kompleksnija od prostog ograničenja koje se uglavnom javlja u literaturi. Razmatranje faktora različitih cena na vezama snabdevač-korisnik dobija na važnosti, na primer, kada na raspolaganju ima više različitih tipova prevoznih sredstava na jednoj vezi od snabdevača do korisnika. Zaključuje se da veza snabdevač-korisnik nekad mora da se uspostavi sa kombinacijom različitih ograničenja, tj. modula. To je slučaj kada na raspolaganju imamo skup kamiona različite nosivosti, različitog dometa sa punim rezervoarom i različite potrošnje goriva. Ako imamo više od jednog kamiona svakog tipa, dobija se situacija sa ograničenim modularnim vezama. Dodatni primer mogu biti telekomunikacije u slučaju da su dostupni kablovi različitog kapaciteta protoka signala.

Hab lokacijski problem ograničenih kapaciteta sa modularnim vezama (Capacitated hub location problem with modular link capacities - HMLC) je specijalni slučaj klasičnog hab lokacijskog problema. Kod HMLC podrazumeva se da cena korišćenja veza nije linearna već stepenasta (progresivna). Za razliku od većine hab lokacijskih problema sa ograničenim kapacitetima kod kojih je ograničen protok robe koji dolazi u hab, kod ovog problema habovi imaju ograničen kapacitet tranzita. Ovaj problem su prvi put formulisali Yaman i Carello u radu [112], gde je predložen egzaktna metoda grananja i sečenja (Branch and Cut -BnC) i metaheuristika Tabu pretrazivanja.

Postoji nekoliko radova u kojima su predstavljeni modeli sa opštom cenom lokacija. Na primer, u radu [19] autori Correia i Captivo proučavaju problem gde se ograničenje svakog čvora (lokacije) mora izabrati iz skupa mogućih ograničenja, pri tom je svako ograničenje povezano sa drugačijom cenom. U radu [109] razmotrena je podesiva cena (set-up cost) koja zavisi od čvorova, i predstavlja neopadajuću funkciju broja čvorova odredišta koje su servisirane od strane jednog čvora snabdevača. U radu [109] razmotrena je uopštena podesiva cena koja je funkcija po veličini lokacije. Problem HMLC je NP-težak problem što je pokazano u radu [42].

Slični hab lokacijski problemi su u literaturi razmatrani u nekoliko radova. U radu [9] autor razmatra problem gde svaki ne-hab čvor može biti dodeljen većem broju hab čvorova. U radovima [29] i [62] autori predlažu da se razmotri hab lokacijski problem ograničenih kapaciteta sa jednostrukom alokacijom gde postoji cena za rutiranje protoka ali ne postoji cena za uspostavljanje veza. U [29] ograničenje habova se odnosi samo na dolazeći protok. Međutim, u radu [45] ograničenje se odnosi na količinu protoka koja prolazi kroz hab, slično kao u problemu koji je tema ovog rada. U radu [13] Carello je razmotrio generalizaciju HMLC problema, ali bez predloga egzaktnog metoda za rešavanje.

3.2. Matematička formulacija

Neka je I skup potencijalnih lokacija fabrika, svaka sa kapacitetom Q, i neka je J skup lokacija korisnika ili čvorova odredišta. Za svakog korisnika $j \in J$, označimo sa d_j količinu robe koju zahteva ta lokacija. Neka je f_i fiksna cena za uspostavljanje lokacije fabrike na lokaciji $i \in I$. Pretpostavlja se da na svakoj vezi fabrika-korisnik može postojati više modula različitih veličina. U praksi, moduli mogu odgovarati određenom tipu usluge ili proizoda. Skup različitih modula ćemo označiti na sledeći način {1, 2, ..., L}. Neka je C^l kapacitet modula tipa l i neka je g^l cena tog modula. Neka je c_{ij} cena distribucije na vezi (i, j), izmedju fabrike na lokaciji $i \in I$ i korisnika na lokaciji $j \in J$. Cena rešenja je suma cena uspostavljanja lokacija i fiksne i promenljive cene uspostavljanja veza izmedju lokacije fabrike i lokacije korisnika. Cilj problema je obezbediti da je svaki zahtev ispunjen sa minimalnim cenama.

Da bi smo formulisali naš problem uvodimo i sledeće promenljive. Celobrojna promenljiva x_{ij} pokazuje količinu protoka (broj jedinica količine robe) koji fabrika na lokaciji i šalje lokaciji j. Promenljiva u^l_{ij} predstavlja broj različitih modula l koje možemo instalirati na vezu između čvora snabdevača i i čvora korisnika j. Binarna promenljiva y_i pokazuje da li je uspostavljena fabrika na lokaciji i. Matematička formulacija problema preuzeta je iz rada [20].

Matematička formulacija glasi :

$$\text{Min} \quad \sum_{i\in I} f_i y_i + \sum_{i\in I}\sum_{j\in J}\sum_{l=1}^{L} g^l u^l_{ij} + \sum_{i\in I}\sum_{j\in J} c_{ij} x_{ij} \tag{1}$$

pri uslovima:

$$\sum_{i\in I} x_{ij} = d_j \quad , j \in J \tag{2}$$

$$\sum_{j\in J} x_{ij} \leqslant Q y_i \quad , i \in I \tag{3}$$

$$x_{ij} \leqslant \sum_{l=1}^{L} C^l u^l_{ij} \quad , i \in I, j \in J \tag{4}$$

$$u^l_{ij} \geqslant 0 \quad , \quad u^l_{ij} \in \mathbb{Z} \quad , i \in I, j \in J, l \in \{1, ..., L\} \tag{5}$$

$$x_{ij} \geqslant 0 \quad , \quad x_{ij} \in \mathbb{Z} \quad , i \in I, j \in J \tag{6}$$

$$0 \leqslant y_i \leqslant 1 \quad , \quad y_i \in \mathbb{Z} \quad , i \in I \tag{7}$$

Funkcija cilja (1) minimizuje ukupne troškove, koji se sastoje od troškova uspostavljanja fabrika na čvorovima snabdevačima, uspostavljanja modula na vezama između čvorova snabdevača i čvorova korisnika i troškova distribucije u mreži. Uslov (2) garantuje da je svaki zahtev za robom od strane čvora korisnika zadovoljen od strane čvorova snabdevača. Uslov (3) obezbeđuje da su ispunjeni uslovi ograničenja ćvorova snabdevača. Uslov (4) obezbeđuje da moduli na vezi između čvora snabdevača i i čvora korisnika j zajedno imaju dovoljan kapacitet da propuste protok na toj vezi.

4.Predloženi algoritam za rešavanje problema

4.1. Osnove algoritma tabu pretraživanja

Za rešavanje teških kombinatornih problema koriste se heurističke metode, koje se još zovu i metode približnih rešenja. Većina problema kombinatorne optimizacije koje imaju primenu u realnim situacijama spada u klasu NP-teških problema, iz čega može da se zaključi da je izuzetno teško naći egzaktno rešenje na efektivan način. Zbog toga je kao generalni pristup prilikom rešavanja kombinatornih problema odabrana primena heurističkih metoda. Postoji puno vrsta heurističkih metoda koje se primenjuju na razne kombinatorne probleme u zavisnosti od prirode samog problema. Jedna od najpopularnijih heurističkih metoda je ona koja se zasniva na lokalnom pretraživanju. Takve heuristike u opštem slučaju predstavljaju iterativne metode pretraživanja koje polaze od početnog dopustivog rešenja i unapređuju ga u svakoj narednoj iteraciji pomoću serije pokreta. Od početnog rešenja i od skupa transformacija zavisi kvalitet krajnjeg rešenja i vreme izvršavanja samog algoritma.

Fred Glover je 1986. godine predstavio novu heuristiku pod nazivom tabu pretraživanje [40]. Drugi nazivi ove heuristike koji nisu zaživeli u upotrebi su heuristika najstrmijeg uspona – najblažeg pada (steepest ascent – mildest descent). Fred Glover tabu pretraživanje nije posmatrao kao heuristiku već kao metaheuristiku, tj. smatrao je da može da se koristi kao opšta strategija kontrolisanja heuristike koja je prilagođena specifičnom problemu.

Tabu pretraživanje je sa velikim uspehom predstavljeno u velikom broju radova koji se tiču najrazličitijih problema kombinatorne optimizacije [41, 86, 89, 93]. Najčešće, tabu pretraživanje je nalazilo optimalno rešenje ili rešenje visokog kvaliteta. Takođe, tabu pretraživanje se pokazalo kao veoma efikasna metoda za rešavanje skoro svih vrsta problema kombinatorne optimizacije. Zbog toga je tabu pretraživanje postalo jako popularno među onima koji se bave pronalaženjem rešenja visokog kvaliteta za kombinatorne probleme velikih dimenzija.

Reč tabu potiče iz Polinezije i označavala je stvari koje su nedodirljive, kao što su svetinje. U današnje vreme, reč tabu ima šire značenje i može se definisati kao "zabrana određena pravilima društva" ili prostije "nešto što je zabranjeno". Opis reči tabu je pokazatelj kakva je priroda tabu pretraživanja. Rizik je jedna od karakteristika tabu pretraživanja, kao i svih heuristika koje se oslanjaju na pretraživanje. Taj rizik se ogleda u tome što kretanje (pretraživanje) u smeru koji donosi poboljšanje može da dovede do toga da se ne dobije kvalitetno rešenje. Takođe, zabranjena rešenja mogu biti ponovo izabrana kada to rešavanje problema zahteva. Ovo znači da se lista tabua, tj. zabrana, sa vremenom menja, da nešto što je nekad bilo zabranjeno sada ponovo dozvoljeno. Bitna stavka tabu pretraživanja je memorija koja igra bitnu ulogu u određivanju tabu statusa elementima pretraživačkog prostora.

Tabu pretraživanje povećava mogućnost rešavanja velikog broja problema koji su jako značajni za industriju. Ovaj algoritam se primenjuje u oblastima telekomunikacije, bankarstva, berzanskim transakcijama, planiranje prostora, planiranju resursa, raspoređivanje energije, biomedicina, bioinženjering, dizajn kompjuterskih komponenti, bojenje grafova, deljenje grafova .

Tabu pretraživanje je iterativni postupak koji polazi od datog početnog rešenja koje pokušava da poboljša (da ga zameni boljim rešenjem) u svakoj iteraciji. Osnovni pristup tabu pretraživanja je da se nastavi sa pretraživanjem kada god algoritam naiđe na lokalni optimum primenom poteza koji ne donose unapređenje sa ciljem da se lokalni optimum prevaziđe. Da bi sprečili pretragu da se vraća

rešenjima koje je već obradila, potezi koji vode do tih rešenja postaju zabranjeni , tj. postaju tabui. Podaci o toku pretrage se čuvaju u memoriji na osnovu koje pretraga biva sprečena da se vrati u posećena rešenja.

Algoritam tabu pretraživanja se može posmatrati kao specijalni slučaj lokalnog pretraživanja. Osnovni algoritam tabu pretraživanja se sastoji od lokalne pretrage i adaptivne memorije. Dve bitne karakteristike tabu pretraživanja su pretraživački prostor i okolina. Način na koji se odabere pretraživački prostor i okolina rešenja je najbitniji korak u celom dizajnu algoritma tabu pretraživanja. Pretraživački prostor je skup svih mogućih rešenja koja mogu biti posećenja tokom pretrage. Struktura pretraživačkog prostora može biti raznovrsna i zavisi isključivo od problema koji se razmatra. Pretraživački prostor može da bude skup svih dopustivih rešenja problema, skup tačaka koje sadrže neke podatke o samom rešenju, skup vektora, skup ekstremnih tačaka i tako dalje. Treba naglasiti da nije preporučljivo ograničiti pretraživački prostor na skup dopustivih rešenja. Nekada, dozvoljavanje pretrazi da posećuje i razmatra nedopustiva rešenja može biti poželjno, a u nekim situacijama je i neophodno. Sledi zaključak da se u tim situacijama u pretraživačkom prostoru nalaze i nedopustiva rešenja.

Okolina rešenja je takođe bitna karakteristika tabu pretraživanja. U svakoj iteraciji pretraživanja, transformacije koje se mogu primeniti na tekuće rešenje predstavljaju okolinu tog rešenja (skup susednih rešenja). Te transformacije se mogu nazvati potezi. Okolina određenog rešenja se može zamisliti kao podskup skupa pretraživačkog prostora, tako da se u tom skupu nalaze rešenja dobijena primenom jedne lokalne transformacije na tekuće rešenje. Za svaki problem postoji više različitih okolina nego pretraživačkog prostora. Okolina se može dobiti primenom jednostavnih transformacija, ali i primenom komplikovanih. Takve okoline su samim time veće i složenije, pa pretraživanje takvih okolina može pogoršati efikasnost celokupnog algoritma. U tim situacijama je neophodno izvršiti neku vrstu redukcije okoline. Takođe, definicija okoline rešenja zavisi od definicije pretraživačkog prostora.

Tabu je glavna karakteristika tabu pretraživanja. Tabui se koriste kako bi se sprečilo da algoritam udje u cikluse prilikom udaljavanja od lokalnog optimuma. Kada pretraživanje dođe u lokalni optimum, algoritam pokušava da poboljša lokalni optimum potezima koji ne dovode do poboljšanja. Takvim potezima se u okolini tekućeg rešenja bira novo rešenje koje nije bolje od tekućeg i proces pretrage se nastavlja u okolini tog rešenja. Neophodno je zabraniti da se proces pretrage vrati u već posećena rešenja, jer se tako dobijaju ciklusi. Ta zabrana se realizuje tako što se već posećena rešenja proglase zabranjenim (ta rešenja postaju tabu) u određenom broju narednih iteracija. Broj iteracija koliko je neko rešenje tabu se zove trajanje tabua (Tabu Tenure). Ne čuvaju se cela rešenja, već potezi koji dovode do njih. To se radi zbog toga što je potrebno previše vremena da se proveri da li je potencijalno naredno rešenje tabu. Na pimer, kod binarnih problema zabranjuje se promena vrednosti indeksa sa 1 na 0 i sa 0 na 1. Ako je u pitanju problem vezan za grafove, tabu se implementira tako što se zabrani dodavanje grane (ako je pre nije bilo) ili oduzimanje grane (ako je pre toga bila tu).

Iako teoretski ne postoji mogućnost da nastane ciklus, zbog svojstava tabua može doći do toga da blokiramo neke jako dobre poteze, čime smo oslabili proces pretraživanja. Zbog toga se uvodi postupak ukidanja tabua, tj. ukidanje statusa tabua potezima. Taj postupak se zove kriterijum aspiracije. Najčešće korišćeni kriterijum asporacije je ujedno i najjednostavniji i on se implementira tako da dopušta tabu poteze kada vode ka rešenju koje je bolje od trenutnog najboljeg rešenja. U nekim komplikovanijim implementacijama primenjuje se princip da se tabu može skroz zanemariti, ako ne može dovesti do formiranja ciklusa.

Tabui koji su obogaćeni kriterijum aspiracije menjaju okolinu nekog rešenja jer dopuštaju poteze koji inače ne bi bili dopušteni. Za skup rešenja kažemo da je dozvoljen kada se sastoji od svih rešenja iz okoline datog rešenja koja nisu tabu i svih rešenja iz iste okoline koja jesu tabu ali prestaju da budu tabu zbog kriterijuma aspiracije. Zahvaljujući kriterijumu aspiracije okolina rešenja može da

postane dinamična.

Tabu pretraživanje je sastavljeno do dve komponente. Prva je pretraga, a druga je adaptivna memorija. Pretraga je značajna jer možemo da pretpostavimo da loš strateški izbor rešenja može da da više korisnih informacija nego dobar slučajan izbor. U sistemu koji koristi memoriju, loš strateški izbor može da nas snabde informacijama kako je potrebno da menjamo strategiju koju smo do tada koristili. Adaptivna memorija omogućuje da tabu pretraživanje bude ekonomičnije i efikasnije.

Memorija tabu pretraživanja ima četiri bitne karakteristike (u nekim radovima one se zovu i dimenzija memorije), a to su bliska prošlost (Recency), učestalost (Frequency), kvalitet (Quality) i uticaj (Influence). Bliska prošlost i učestalost se odnose na sposobnost memorije da čuva različite tipove podataka. Informacije o bliskoj prošlosti se odnose na podatke o pretraživanju koje se odigralo neposredno pre tekućeg trenutka. Učestalost je sposobnost memorije da pamti učestalost nekog elementa tokom celog procesa pretrage. Kvalitet je sposobnost memorije da izdvoji dobra rešenja iz skupa loših na osnovu nekih informacija. Memorija uočava sličnosti između rešenja visokog kvaliteta ili puteva koji vode do takvih rešenja. Ta karakteristika memorije je jako važna jer ona navodi pretragu u predele sa dobrim rešenjima, a samim time odvlači pretragu od predela pretraživačkog prostora sa lošim rešenjima. Poslednja karakteristika, uticaj, se odnosi na uticaj koji napravljene odluke tokom tabu pretraživanja imaju na kvalitet i strukturu rešenja.

Memorija tabu pretraživanja može biti eksplicitna ili atributna. Eksplicitna memorija čuva cela rešenja. To je slučaj kada pretraga naiđe na elitno rešenje tokom procesa pretrage. Eksplicitna memorija može da se proširi tako da sadrži i susede elitnih rešenja koji još nisu posećeni. Sačuvana elitna rešenja, i njihovi susedi u slučaju proširene eksplicitne memorije, se mogu koristiti u svrhu proširenja tabu pretrage. Atributna memorija čuva informacije o atributima rešenja koji su se menjali prilikom prelaska sa jednog rešenja na drugo.

Po trajanju čuvanja podataka, memorija se može podeliti na kratkoročnu i dugoročnu memoriju. Kratkoročna memorija (Short-term memory) sadrži informacije iz bliske prošlosti procesa pretraživanja i u najvećem broju slučajeva beleži se samo ograničena količina informacija. U ovoj memoriji se najčešće čuvaju tabui, pa se ova memorija često zove tabu lista. Već smo spomenuli da tabui nisu cela rešenja već potezi do njih, tako da ova memorija spada u klasu atributnih memorija. Najvažniji parametar tabu liste je dužina. Dužina tabu liste se najčešće fiksira na tačno određenu vrednost koja zavisi od problema kombinatorne optimizacije koji se razmatra. Teoretski, dužina tabu liste je neograničena ali to ne bi bilo dobro jer bi porasla jako mnogo što bi usporavalo pretragu. Zaključujemo da se dužina tabu liste mora pažljivo birati, jer od nje zavisi brzina algoritma pretraživanja. Dužina tabu liste je ujedno i tabu mandant, tj. određuje koliko narednih iteracije će neki element biti u tabu listi. Novi tabu se dodaje na početak liste, a ako je lista puna pre dodavanje potrebno je izbaciti element sa kraja tabu liste, tj. element koji je najduže od svih u listi. Primetno je da je tabu lista struktura tipa LIFO (Last In – First Out). Ako je tabu lista predugačka, previše elemanata je eliminasano kao potencijalno rešenje. Ako je tabu lista previše kratka, može doći do pojave ciklusa koji su duži od dužine tabu liste, što bi dovelo do gubljenja osnovne funkcije tabu liste. Uobičajena je praksa da se tabu liste implementiraju sa fiksnim dužinama. Međutim, u određenim situacijama tabu liste fiksne dužine ne mogu uvek da spreče pojavljivanje ciklusa. Zbog toga je predložena implementacija tabu liste sa promenljivom dužinom [38, 39]. Druga mogućnost je da se proizvoljno generiše tabu mandant za svaki potez u okviru iteracije. Mora se izvršiti reorganizacija tabu pretraživanja tako da svaki tabu u sebi sadrži informaciju do kada je bio aktivan, tj. od kada je postao tabu.

Dugoročna memorija (Long-Term Memory) obezbeđuje informacije koje su komplementarne informacijama koje su u tabu listi. Dugoročna memorija je najčešće atributna memorija i čuva informacije o elementima koji su prisutni u tekućem rešenju tokom celog procesa pretrage. Dugoročna memorija može biti organizovana na dva načina. Prvi način je da se za svaki atribut čuva broj iteracija

u kojima je dati atribut bio prisutan u tekućem rešenju, što se može shvatiti kao postojanost (konstantnost) tog atributa. Drugi način organizovanja dugoročne memorije je da se čuva broj iteracija u kojima je atribut ulazio ili izlazio iz tekućeg rešenja. Taj podatak se zove tranzitivnost atributa. Dugoročna memorija je ključna stavka prilikom implementacija diversifikacije i intenzifikacije rešenja, o čemu će biti više detalja u nastavku teksta.

U narednoj sekciji ćemo opisati osnovni algoritam tabu pretraživanja. Prikazana verzija tabu pretraživanja je najčešće korišćena verzija tabu pretraživanja. Ova verzija algoritma istražuje celu dozvoljenu okolinu određenog rešenja i prelazi u optimum koji nalazi. Pseudokod tabu pretraživanja prikazan je Algoritmom 3.

Algoritam 3. Osnovni algoritam tabu pretraživanja

1. Konstrukcija početnog rešenja s_0;
2. $s = s_0$, $s^* = s_0$, $f^* = f(s_0)$, $T = \emptyset$;
3. while (! KriterijumZaustavljanja())
4. s' = NajboljeRešenjeOkoline($D(s)$);
5. $s = s'$;
6. if ($f(s) < f^*$)
7. $f^* = f(s)$;
8. $s^* = s$;
9. endif
10. AžuriranjeTabuListe (T);
11. endwhile
12. vratiti s*;

Kriterijum zaustavljanja je uslov koji mora biti ispunjen da bi se algoritam zaustavio u određenom trenutku. Da ne postoji kriterijum zaustavljanja, tabu pretraživanje bi moglo da se izvršava neograničeno dugo (sem u situaciji kada je optimalno rešenje istovremeno i početno rešenje). Kao kriterijum zaustavljanja se najčešće bira određen broj iteracija, određeno CPU vreme, istek unapred zadatog vremena, broj uzastopnih iteracija u kojima nije došlo do poboljšanja i unapred dostignuta vrednost koju treba da dostigne funkcija cilja. Kao najčešći kriterijum zaustavljanja bira se broj uzastopnih iteracija bez poboljšanja vrednosti funkcije cilja. Ako se tabu pretraživanje sastoji od više komponenti, pretraga se zaustavlja kada sa pretragom završi svaka od tih komponenti. Svaka komponenta može imati različite kriterijume zaustavljanja.

Do sada smo opisali osnovne koncepte i karatkeristike tabu pretraživanja. Veliki broj problema kombinatorne optimizacije može biti rešen standardnim imlpementacijama, međutim, veliki broj problema zahteva da se algoritam obogati dodatnim elementima kako bi se dobila rešenja visokog kvaliteta.

Jedan od tih dodatnih elemenata tabu pretraživanja može da bude intenzifikacija. Suština

intenzifikacije je menjanje pravila pomoću kojih biramo rešenja tako da se omogući izbor elemenata rešenja koji su se pokazali dobri kroz proces pretrage. Takođe, intenzifikacija može da odvede pretraživački proces u regione sa kvalitetnim rešenjima, kako bi ti regioni bili detaljnije ispitani. Uopšteno, svrha intenzifikacije je, kao što ime kaže, intenzivno pretraživanje određenih regiona pretraživačkog prostora. Intenzifikacija se temelji na srednjeročnoj memoriji u kojoj se za svaki element rešenja čuva maksimalan broj uzastopnih iteracija takvih da je taj element bio u tekućem rešenju na toj određenoj poziciji.

Najčešće korišćeni metod intenzifikacije je ponovno pretraživanje do tada najboljeg poznatog rešenja pri čemu se fiksiraju elementi koji su prograšeni dobrim. Tehnika koja može biti korisna je promena okoline pretrage, što dovodi do raznovrsnijih poteza. Intenzifikacija se koristi u mnogim implementacijama tabu pretraživanja, iako nije uvek neophodna. To je zato što postoje mnoge
situacije gde je normalan proces pretraživanja dovoljno temeljan. Drugi način implementacije intenzifikacije se zasniva na elitnim rešenjima. Intenzifikacija se u tom slučaju implementira tako da se pretraga tokom pretraživačkog procesa premesti u neko elitno rešenje, čime bi se ispitala okolina tog rešenja.

Jedan od najčešćih problema algoritama kojima je osnova lokalno pretraživanje je taj da postoji tendencija da budu previše ograničene. Ovaj problem se javlja i kod tabu pretraživanja, bez obzira što je tabu pretraživanje, na neki način, poboljšanje lokalne pretrage. Većinu vremena koju utroši procedura pretrage provede na ograničenom prostoru. Još jedan negativni aspekt toga je da najzanimljiviji delovi pretraživačkog prostora mogu ostati neistraženi, čime bi se dobila nekvalitetna rešenja. Diversifikacija je mehanizam koji rešava ovaj problem tako što pretraživanje navodi u regione pretraživačkog prostora koji nisu prethodno bili istraženi. Način implementacije diversifikacije se zasniva na dugoročnoj memoriji, na primer frekventna memorija. U memoriji se za svaki atribut čuva ukupan broj iteracija od početka pretraživačkog procesa tako da je taj atribut bio prisutan u tekućem rešenju. Ovde izdvajamo dva najznačajnija načina da se implementira diversifikacija. Prva je diversifikacija restarta. Diversifikacija restarta neke elemente koji su retko korišćeni stavlja u tekuće ili do tada najbolje rešenje, zatim restartuje pretragu iz te tačke. Drugi metod diversifikacije se zove kontinualna diversifikacija. Kontinualna diversifikacija integriše rezultate diversifikacije u sam proces tabu pretraživanja. Integracija se postiže promenom funkcije cilja tako što se funkcija cilja sabere sa vrednošću koja zavisi od frekvencije elemenata rešenja. Za tu novu vrednost funkcije cilja se kaže da je penalizovana. Formula je sledeća :

$$f' = f + div * pen,$$

gde je f vrednost funkcije cilja, f' je penalizovana vrednost funkcije cilja i pen je vrednost penala i predstavlja funkciju frekvencija elemenata rešenja. Promenljiva div je parametar diversifikacije. U slučaju minimizacije, vrednost penala je manja kod onih rešenja čiji elementi imaju manju frekvenciju. Tako se obezbeđuje da će pretraga razmatrati rešenja koja u sebi sadrži retko korišćene elemente. Rešenja koja se koriste ne moraju nužno biti visokog kvaliteta, ali će zbog dodavanja penala na funkciju cilja možda biti i bolja od nekih prethodnih visoko kvalitetnih rešenja. Dalje će to proces pretraživanja da odvede u regione koji u ranijim segmentima pretrage nisu bili istraženi. Mera diversifikacije rešenja direktno je proporcionalna sa parametrom diversifikacije [98].

U nekim situacijama izračunavanje funkcije cilja može biti preskupo. Nekada računanje vrednosti funkcije cilja zahteva rešavanje potproblema ili računanje funkcija koje čine funkciju cilja, pa to umanjuje efikasnost samog algoritma. Jedan od načina da se reši ovaj problem je da nastavi pretraživanje sa izmenjenom funkcijom cilja. Izmenjena funkcija cilja je najčešće funkcija koja je na neki način povezana sa originalnom funkcijom cilja, ali je, logično, lakša za izračunavanje. Uz pomoć

izmenjene funkcije cilja dobija se mali broj potencijalno dobrih kandidata. Ti kandidati mogu da daju najbolje vrednosti za izmenjenu funkciju cilja. Zatim, se originalna funkcija cilja izračunava na osnovu skupa poteza i najbolje od novih rešenja se uzima kao tekuće rešenje.

Ograničavanje pretraživačkog prostora je problem koji može da se reši na nekoliko načina. Jedna od mogućnosti je relaksacija ograničenja. Relaksacija ograničenja proširuje prostor pretraživanja koji se može pretražiti jednostavnijim okolinama nego što je to bio slučaj pre relaksacije. Relaksacija ograničenja je implementira tako što se izbace neka ograničenja (što manje uslova ima, problem je prostiji za rešavanje - relaksiran) tokom definisanja pretraživačkog prostora i dodavanjem penala na funkciju cilja za narušavanje ograničenja. Proces određivanja penala za funkciju cilja može biti komplikovan. Jedan od načina da se reši ovaj problem je da se parametri penalizacije dinamički podešavaju na osnovu podataka iz skorijeg pretraživanja. Penal se povećava ako su rešenja u nekoliko iteracija nedopustiva, a smanjuje se ako su skorašnja rešenja dopustiva. Ponovo, razmatranje nedopustivog rešenja ne mora nužno biti traćenje vremena. Pretraga okoline nedopustivog rešenja može proces pretrage da odvede u regione sa jako dobrim dopustivim rešenjima.

Još jedan način da se određuju penali funkcije cilja je da se sistematski modifikuju tako da se pretraga namerno usmeri u regione sa lošim rezultatima, čime se aktivira proces diversifikacije. Poslednja opisana tehnika poznatija je kao strateško oscilovanje [37]. Strateško oscilovanje je metoda navođenja procesa pretraživanja u odnosu na kritičnu granicu vrednosti, granicu oscilovanja. Granica oscilovanja je najčešće granica dopustivosti. U opštem slučaju, proces pretraživanja se zaustavlja kada dođe u zonu oko kritične granice. Međutim, kod kritičkog oscilovanja se pravi izuzetak i pravila se menjaju tako da pređe ta granica. Proces pretrage se dalje nastavlja, sa druge strane kritične granice, do određene dubine. Dubina do koje prelazimo crvenu liniju se može odrediti na razne načine. Dubina može biti broj iteracija koje pretraživanje napravi nakon prelaska granice oscilovanja ili dubina može da bude maksimalna vrednost penala nedopustivosti rešenja. Nakon što se dostigne određena dubina, pretraga se okreće, prelazi granicu oscilovanja (sada je ponovo u dopustivom delu prostora) i kreće se sve dok ne dostigne granicu oscilovanja "sa druge strane" i prelazi je do određene dubine. Generalno, pretraga naizmenično prelazi granicu oscilovanja. Pretraga se neće vraćati istim putem zbog implementacije tabu algoritma. Nivo oscilacije je mera nedopustivosti koja određuje koliko je pretraga daleko otišla preko granice oscilovanja.

Moguće je da skup dozvoljenih rešenja bude veoma veliki što može da uspori pretraživački proces. Strategija liste kandidata ima zadatak da smanji broj elemenata u skupu dozvoljenih rešenja. Tabu pretraživanje može da se koristi i kao metoda za rešavanje teških problema koji u sebi sadrže razne potprobleme ili više njih. U tim slučajevima algoritam mora pored glavnog kombinatornog problema da rešava i njegove potprobleme, što se može ispostaviti kao veoma komplikovan i skup proces. Stvara se potreba za metodom koja može da izdvoji delove regiona pretraživačkog prostora koji sadrže kvalitetna rešenja i da detaljno ispita ta izdvojena rešenja. Takva strategija se zove strategija liste kandidata koja kvalitetna rešenja ili poteze, u zavisnosti od problema, čuva u listi. Elementi sačuvani u listi se mogu kasnije detaljno ispitati. Strategija liste kandidata istovremeno povećava brzinu pretrage i povećava kvalitet pronađenih rešenja.

Postoji puno načina da se implementira strategija liste kandidata. Dva najpoznatija načina su strategija liste elitnih kandidata i strategija P aspiracije.

Strategija P aspiracije ili strategija plus aspiracije obezbeđuje graničnu vrednost za kvalitet poteza na osnovu iskustva stečenog tokom pretraživačkog procesa. Zadatak strategije P aspiracije je da izračunava kvalitet svakog poteza, što podrazumeva kvalitet i cenu poteza, sve do pronalaženja poteza koji je u okviru granične vrednosti. Nakon pronalaženja takvog poteza, strategija izračunava još dodatnih P poteza, zatim se bira najbolji od P + 1 poteza za izvršavanje. Vrednost broja P, tj. za koliko poteza se izračunava njihov kvalitet, se nalazi između dve vrednosti koje su tako odabrane da broj P ne

bude ni preveliki ni premali. Neka je F redni broj poteza koji prvi zadovoljava graničnu vrednost. Ako su vrednosti Max i Min nepoznate, ukupno će biti razmatrano F + P poteza. Ako su Max i Min navedeni, i ako važi da je F + P < Min, vrši se izračunavanje Min poteza. Analogno, ako je F + P > Max vrši se izračunavanje ukupno Max poteza.

Linija aspiracije je granična vrednost kvaliteta poteza i ta vrednost može da bude dinamička. Ako imamo niz kvalitetnih uzastopnih poteza koji su doneli napredak ciljnoj funkciji, može se pretpostaviti da će i naredni potez biti istih ili sličnih karatkeristika, pa se linija aspiracije može pomeriti (podići). Slično, za vreme perioda loših (manje kvalitetnih) poteza koji ne donose poboljšanje vrednosti funkcije cilja, linija aspiracije se može spustiti. Specijalni slučaj strategije P aspiracije je kada je vrednost broja P nula i ta strategija se zove strategija prvog poboljšanja (First Improving Strategy). U ovom specijalnom slučaju linija aspiracije će prihvatiti prvi potez koji donosi poboljšanje, dok se parametri Max i Min ignorišu. Prethodno opisani primer je u stvari pretraga sa prvim poboljšanjem. Najvažnije kod implementacije strategije P aspiracije je da se potezi u tekućoj iteraciji razlikuju od poteza iz prethodnih iteracija. Specifičnost vezana za tabu algoritme je da se ne razmatraju potezi koji su inverzni prethodno razmatranih poteza, jer bi inače bili dobijeni ciklusi.

Strategija liste elitnih kandidata prvo generiše master listu. Master lista se dobija analizom svih ili skoro svih poteza i izborom *k* najboljih. Sledeće se u svakoj narednoj iteraciji izvršava najbolje rangirani potez iz master liste. Potezi iz master liste se primenjuju ako potez zadovoljava uslov da mu je kvalitet iznad određenog nivoa (sve dok potez ima status elite) ili ako je izvršeno manje iteracija nego što je unapred zadato. Suštinska ideja strategije liste elitnih kandidata je da dobar potez zadržava status elitnog kroz višestruko mnogo iteracija, dok potez koji nije elitni može da menja status izvršavanjem iteracija. Dovoljno dobri potezi se izvršavaju, ali to može da promeni statuse drugih elemenata master liste. Ako relativno veliki broj poteza master liste tokom iteracija izgubi status kvalitetnog poteza, generiše se nova master lista.

4.2. Implementacija algoritma tabu pretraživanja za rešavanje problema HMLC

4.2.1 Kodiranje i generisanje početnog rešenja

Predloženi algoritam tabu pretraživanja za rešavanje problema HMLC sastoji se od dva algoritma koji rešavaju potprobleme originalnog problema. Problem HMLC razdvajamo na njegova dva potproblema, lokacijski potproblem (location subproblem) i potproblem dodeljivanja (assigment subproblem).

Prvi korak algoritma je inicijalizacija populacije i pronalaženje početnog rešenja. Validno početno rešenje se traži pomoću pohlepnog algoritma (Greedy Algorithm). Proverava se da li su podaci validtni, tj. da li zadovoljavaju sva ograničenja koja su zadata u matematičkoj formulaciji problema. Ako nisu validni podaci, vrši se ponovna inicijalizacija. Bira se početno rešenje na proizvoljan način. Početno stanje je da je skup habova prazan, pa se habovi dodaju jedan po jedan dok se ne dobije validno rešenje. Drugi korak je primena iteracije tabu pretraživanja na lokacijski potproblem. Treći korak je primena lokalne pretrage na dvadeset najboljih skupova habova koji su rezultati prethodnih koraka.

U tabu pretrazi rešenje je predstavljeno kao skup indeksa lokacija na kojima su uspostavljeni

habovi. Za taj skup habova, povezivanje sa ne-hab čvorovima se vrši pomoću pohlepnog algoritma. Pohlepni algoritam jedan po jedan ne-hab čvor povezuje sa habom tako se minimizuju slučajevi neodgovarajućih veza. Ako pohlepni algoritam ne nađe pogodnu alokaciju u okolini rešenja, taj kandidat se prograšava nepogodnim.

4.2.2 Definicija okoline rešenja i poteza

Okolina rešenja se generiše sledećim potezima. Potez dodavanja (adding move) služi uspostavljanju novog hab čvora. Potez uklanjanja (removing move) uklanja određeni hab iz skupa hab čvorova. I potez razmene (swapping move) koji uklanja jedan hab čvor i uspostavlja drugi hab čvor na drugoj lokaciji. Primetno je da prva dva poteza menjaju kardinalnost skupa habova, dok poslednji potez čuva kardinalnost tog skupa.

4.2.3 Parametri algoritma

Na osnovu stečenog iskustva u izračunavanju, postavljene su vrednosti ostalih bitnih parametara tabu pretrage. Kriterijum zaustavljanja je ispunjen ako algoritam izvrši sto pedeset (150) iteracija bez poboljšanja tekućeg rešenja. Dužina tabu liste je šest, tj. ona čuva poteze koji su doveli do poslednjih šest rešenja. Da bi se razmatrala dobra rešenja koja su posledica tabu poteza, primenjen je kriterujm aspiracije. Tabu potez će biti razmotren ako dovodi do rešenja koje je bolje od do tada najboljeg rešenja.

4.2.4 Potproblem dodeljivanja ne-hab čvorova habovima

U sledećem koraku će biti rešen potproblem dodeljivanja. Tokom tog dela algoritma, algoritam lokalne pretrage se primenjuje na dvadeset najboljih skupova habova koje je generisao prethodni korak. Dat je skup hab čvorova a rešenje je predstavljeno vezama ne-hab čvorova sa hab čvorovima. Lokalna pretraga poboljšava sumu transportnih troškova menjanjem alokacije jednog ili najviše dva ne-hab čvora. Primenjuje se strategija prve promene (First Improvement Strategy), tako da prvo rešenje iz okoline koje je bolje od tekućeg rešenja postaje novo rešenje. Kriterijum zaustavljanja lokalne pretrage je ispunjen kada u okolini rešenja ne postoji rešenje koje dovodi do poboljšanja. Za dato rešenje, okolina se generiše primenom sledećih koraka. Prvi korak je promena hab čvora koji je dodeljen ne-hab čvoru. Drugi potez je zamena habova koji su dodeljni dvema ne-hab čvorova. Na primer ako je ne-hab čvor *i* dodeljen hab čvoru *k* i ne-hab čvor *j* je dodeljen hab čvoru *l,* onda se menjaju habovi pa je ne-hab čvor *i* dodeljen hab čvoru *l* i ne-hab čvor *j* je dodeljen hab čvoru *k*. Algoritam 4 prikazuje način na koji je implementirana tabu pretraga za problem HMLC.

Algoritam 4. Koncept tabu pretraživanja za rešavanje problema HMLC

1. *PočetniSkupHabova* = ∅;
2. *PočetniSkupHabova* ← *KonstrukcijaPočetnogRešenjaPohlepnimAlgoritmom();*
3. *SkupHabova* ← *TabuPretraživanje(PočetniSkupHabova);*
4. *ElitniHabovi* ← *SortiranjeDvadesetNajboljih(SkupHabova)*
5. vratiti *LokalnaPretraga(ElitniHabovi)*

4.3. Osnove genetskog algoritma

Genetski algoritmi su jedni od najpoznatijih metaheuristika široke primene. Genetski algoritmi su naučnoj javnosti predstavljeni u radu [43] iz 1975. godine. Genetski algoritmi su prvobitno bili kreirani da simuliraju proces genetske evolucije populacije jedinki pod dejstvom neposrednog okruženja i genetskih operatora. Jedinke koje bi uspele da se prilagode uslovima okruženja, međusobno su se dalje reprodukovale i stvarale novu generaciju jedinki koje su bile bolje od svojih roditelja. Ovaj proces se ponavlja generacijama, pri čemu se prilagođenost neposrednom okruženju poboljšava sa svakom novom generacijom. Postupak se zaustavlja kada je ispunjen jedan ili više kriterijuma zaustavljanja, a najbolji član tekuće populacije predstavlja rešenje genetskog algoritma. Svaka jedinka u populaciji se predstavlja genetskim kodom nad određenom konačnom azbukom. Najščešće se koristi binarno kodiranje, gde se genetski kod sastoji od niza jedinica i nula. Prednost binarnog kodiranja je lako implementiranje. Međutim, prilikom rešavanja specifičnih problema bolje je koristiti azbuke veće kardinalnosti. Kodiranje rešenja je bitan deo genetskog algoritma jer se pogrešnim izborom koda može doći do loših rezultata.

Generisanje početne populacije se obavlja slučajno. Slučajna metoda je najbolji način da se postigne raznovrsnost genetskog materijala. Populacije mogu biti raznih veličina, najveće populacije imaju najviše nekoliko stotina članova. Svakoj jedinki populacije pridružena je funkcija prilagođenosti (fitness function) koja određuje kvalitet te jedinke. Nakon što se svakoj jedinici u populaciji izračuna kvalitet, primenjuje se operator selekcije. Operator selekcije uzima bolje jedinke i formira se nova generacija. Nakon toga nad nekim članovima populacije primenjuju se genetski operatori ukrštanja i mutacije. Genetski operator selekcije ima zadatak da prenosi dobra svojstva jedinki populacije na jedinke sledeće populacije, tj. na generaciju potomaka. Selekcijom se, kako sam naziv kaže, biraju jedinke koje će se reprodukovati i dati novu generaciju, time prenoseći svoje kvalitetne osobine na svoje potomke. Operator selekcije se primenjuje u skladu sa vrednostima funkcije prilagođenosti.

Genetski operator ukrštanja je postupak u kome učestvuju dve ili više jedinki koje ćemo dalje zvati roditeljima. Ukrštanjem roditelja, kao i u prirodi, nastaje jedna ili više jedinki koje nazivamo potomcima. Postoje različiti načini da se izvrši ukrštanje roditeljskog genetskog materijala, a to su jednopoziciono, dvopoziciono i uniformno ukrštanje. Genetskim operatorom ukrštanja se omogućuje prenos genetskog materijala sa roditelja na potomke. Ako su roditelji imali dobar genetski materijal, verovatno će sličan takav imati i potomci. Postoji mogućnost da se ukrštanjem dobije lošiji potomak od roditelja.

Genetski operator mutacije se primenjuje na svaku jedinku populacije potomaka sa unapred

zadatom verovatnoćom mutacije. Verovatnoća mutacije je kao i u prirodnim procesima jako mala, najčešće reda 10^{-3}. Operator mutacije se koristi da bi se povremeno unela raznovrsnost među jedinke populacije koje su međusobno veoma slične. Treba napomenuti da mutacija ne mora dovesti do loših rešenja, kao što ni sve mutacije u prirodi ne moraju da budu maliciozne (plave oči i crvena kosa su primeri benignih genetskih mutacija kod ljudi). Operatori genetskog algoritma se izvršavaju nad populacijom sve dok se ne ispuni jedan ili više kriterijuma zaustavljanja. Kriterijumi zaustavljanja genetskog algoritma mogu biti maksimalan broj generacija, dostignut optimum i razni drugi.

Genetski algoritmi se koriste za rešavanje velikog broja problema kombinatorne optimizacije, a neke od primena mogu se naći u radovima [24, 6, 7]. Primena genetskih algoritama za rešavanje nekoliko hab lokacijskih problema može se naći u [101].

Osnovni koncept genetskog algoritma prikazan je Algoritmom 5.

Algoritam 5. Osnovni koncept genetskog algoritma

```
1. ÚčitavanjeUlaznihPodataka();
2. pop ← GenerisanjePočetnePopulacije();
3. while (!KriterijumZaustavljanja())
4.       for i ← od 1 do veličinaPopulacije
5.              IzračunajFunkcijaCilja(i);
6.       izlazak iz for petlje
7.       Prilagođenost();
8.       OperatorSelekcije();
9.       OperatorUkrštanja();
10.      OperatorMutacije();
11. izlazak iz while petlje
12. vratiti rešenje;
```

Prost genetski algoritam (simple genetic algorithm) je najjednostavnija vrsta genetskog algoritma. Prost genetski algoritam se sastoji od proste rulet selekcije, jednopozicionog ukrštanja i proste mutacije.

Prosta ruletska selekcija može dovesti do preuranjene konvergencije prostog genetskog algoritma. Preuranjena konvergencija se dešava ako jedna ili više jedinki koje nisu optimalne dominiraju populacijom i proces konvergira ka lokalnom ekstremu. Jedinke koje imaju visok stepen prilagođenosti će vremenom istisnuti lošije jedinke iz populacije. Time može doći do toga da budu istisnute jedinke koje, iako su lošije, sadrže kvalitetne gene koji se mogu preneti potomcima. Zaključak je da dolazi do gubitka genetskog materijala. Spora konvergencija je još jedna mana prostog genetskog algoritma. Može da se dogodi da je prosečna prilagođenost jedinki u populaciji velika, da je razlika između optimalne jedinke i ostalih mala, zbog čega genetski algoritam ne može da dostigne optimalno rešenje u razumnom vremenu. Detaljan prikaz prostog genetskog algoritma dat je u [43].

Prost genetski algoritam ima previše nedostataka da bi mogao da služi za rešavanje složenih problema kombinatorne optimizacije. Zbog toga se genetski algoritam obogaćuje raznim algoritmima i metodama kao što su razne vrste kodiranja, razne funkcije prilagođenosti, napredne operatore selekcije, ukrštanja i mutacije.

4.3.1. Kodiranje i funkcija prilagođenosti

U praksi se pokazalo da je kod implementacije genetskog algoritma najbolje koristiti binarno kodiranje. Binarno kodiranje svakom rešenju dodeljuje kôd unapred određene dužine nad azbukom simbola {0, 1}. U nekim situacijama se za kodiranje može koristiti azbuka sa više simbola. Prikazano je detaljno poređenje binarnog kodiranja i drugih načina kodiranja u radu [47].

Funkcija prilagođenosti se može implementirati na razne načine, a neki od načina su direktno preuzimanje, skaliranje u jediničnom intervalu, linearno skaliranje i sigma odsecanje. Ovi načini se mogu koristi samostalno ili se mogu međusobno kombinovati.

Moguće je tokom genetskog algoritma dobiti jedinke koji ne odgovaraju ni jednom rešenju i te jedinke nazivamo nekorektnim jedinkama. Nekorektni kodovi mogu da se ignorišu tako što im se dodeli nula kao vrednost funkcije prilagođavanja. Time se postiže njihovo izbacivanje iz populacije u narednoj generaciju. Primer iz prirode bi bio umiranje, bez reprodukcije, jedinke usled nemogućnosti prilagođavanja. Sledi da u populaciji ostaju samo korektni genetski kodovi, odnosno oni kodovi koji imaju odgovarajuće rešenje. Praksa je pokazala da se ovaj postupak može primeniti samo kada postoji najviše do 70 % nekorektnih kodova. Postoji i mogućnost uključivanja nekorektnih kodova u genetski algoritam. U tom slučaju se računa vrednost svake jedinke u populaciji, s time da se nekorektnim jedinkama računa kaznena funkcija prilagođenosti. Nakon toga, pomoću dobijenih vrednosti računamo prilagođenost svake jedinke i dalje se nastavlja kao u klasičnom genetskom algoritmu. Ovim postupkom se sprečava gubitak genetskog materijala, ali se kaznom obeshrabruje napredak nekorektnih jedinki.

4.3.2. Operator selekcije

Operator selekcije bira jedinke iz trenutne populacije koje će biti upotrebljene za stvaranje nove generacije. Prema Darvinovoj teoriji evolucije, jedinke koje su se najbolje prilagodile sredini u kojoj borave bi trebalo da prežive i reprodukuju se. Osnovni način selekcije bira paran broj p, koji je manji ili jednak veličini populacije. Zatim se p puta vrši biranje jedinki iz populacije u zavisnosti od vrednosti funkcije prilagođenosti. Treba napomenuti da neke jedinke, logično, kvalitetnije, mogu biti izabrane više puta.

Operator selekcije ima više varijatni. Jedna od njih je selekcija zasnovana na principu ruleta [64]. Na ruletskom točku se nalaze kodovi date populacije. Svaki kod zauzima deo točka čija je veličina direktno proporcionalna vrednosti njegove funkcije prilagođenosti. Ruletski točak se zavrti i baca se loptica, koja će sigurno pasti na deo točka i time sigurno odabrati neki kod. Očekivano, loptica će se češće zaustavljati na onim kodovima čija je vrednost funkcije prilagođenosti veća.

Glavni nedostatak ruletske selekcije je što može doći do toga da se često bira kod koji ima preveliku vrednosti funkcije prilagođenosti. Problem je što želimo da izbegnemo jednoličnost generacije potomaka, a ako je jedan kod previše dominantan previše često će biti odabran. Zbog toga se uvodi selekcija na osnovu ranga (Rank Based Selection - RBS) i to na sledeći način. Kod sa najgorom vrednošću funkcije prilagođenosti ima rang 1, a kod sa najboljom vrednošću funkcije prilagođenosti ima rang *max*. Rang kodova zavisi samo od poretka kod u populaciji. Primenom rangiranja izbegli smo preuranjeno konvergiranje pretrage i sporu konvergenciju.

Turnirska selekcija je jedna od najboljih verzije genetskog operatora selekcije. Glavni parametar turnirske selekcije je broj jedinki koji u turniru učestvuju, označimo ga sa N_t . Broj jedinki koje će

učestvovati u ovom obliku selekcije se zadaje unapred. Prvo se biraju proizvoljni podskupovi jedinki tako da je svaki podskup veličine N_t. U svakom od tih podskupova se održava turnir i u narednu generaciju prolazi jedinka sa najboljom vrednošću funkcije prilagođenosti. Najveći problem kod turnirske selekcije je odabrati N_t, tako da se smanje nepovoljni efekti, tako da što kvalitetniji genetski materijal prođe u sledeću generaciju. Metoda turnirske selekcije koja je najbolje rešila taj problem je fino-gradirana turnirska selekcija [30, 31]. Kod ove varijante selekcije željena veličina turnira nije ceo broj, već realan. Od ovog parametra zavisi celokupan proces selekcije, pa bi prosečna veličina turnira trebalo da bude što bliža njegovoj vrednosti. Element populacije se bira tako da ima bolju vrednost funkcije prilagođenosti od svojih konkurenata. Za razliku od obične turnirske selekcije, kod fino-gradirane turnirske selekcije tokom jednog koraka postoje različite vrednosti turnira.

4.3.3. Operator ukrštanja

Ukrštanje je postupak kojim se na slučajan način razmenjuju geni dva roditelja, pri čemu se dobijaju dve nove jedinke, potomci tih roditelja. Ako roditelji imaju visoku vrednost funkcije prilagođenosti, to će verovatno biti i slučaj sa vrednostima funkcija prilagođenosti potomaka. Može se dogoditi da ukrštanjem kombinuju loši geni oba roditelja, pa se dobijaju potomci koji su lošiji od roditelja.

Kod jednopozicionog ukrštanja bira se jedna tačka ukrštanja. Ukrštanje se realizuje tako što prvi potomak uzima gene prvog roditelja do tačke ukrštanja i gene drugog roditelja od tačke ukrštanja, dok drugi potomak uzima obrnuto, gene prvog roditelja od tačke ukrštanja i gene drugog roditelja do tačke ukrštanja.

Kod dvopozicionog ukrštanja biraju se dve tačke ukrštanja. Potomci dobijaju nasledni materijal roditelja na sličan način kao i kod jednopozicionog ukrštanja. Prvi potomak uzima gene prvog roditelja između dve tačke ukrštanja i gene drugog roditelja koji se u kodu nalaze pre prve tačke ukrštanja i posle druge tačke ukrštanja. Drugi potomak uzima gene prvog roditelja koji se nalaze pre prve tačke ukrštanja i posle druge tačke ukrštanja i gene drugog roditelja koji se nalaze između dve tačke ukrštanja.

Uniformno ukrštanje koristi slučajno biranje roditelja za svaki gen koji se prenosi na potomstvo. Roditeljski par generiše filter (maska) pomoću kojeg se određuje čiji gen će dobiti potomak na određenoj poziciji. Filter je binarni niz koji je iste dužine kao i roditeljski kod. Roditelji razmenjuju gene na svim pozicijama na kojima filter ima vrednost 0, dok na mestima gde je vrednost 1 roditelji zadržavaju svoje gene.

4.3.4. Operator mutacije

Mutacija je genetska operacija koja menja sadržaj hromozoma jedinke slučajnom zamenom pojedinih simbola toga koda nekim drugim iz iste azbuke. Genetski operator mutacije se primenjuje nad svakim delom koda sa određenom verovatnoćom, tj. to je verovatnoća koja određuje da li će se mutacija uopšte dogoditi. Verovatnoća je veoma mala, pošto se i u prirodi mutacije jako retko dešavaju. Takođe verovatnoća događaja mutacije ne sme biti ni previše mala, jer se onda nikada neće dogoditi, što dalje povlači da populacija iz generacije u generaciju postaje sve sličnija svojoj prethodnoj populaciji. Nedostatak mutacije povećava mogućnost lokalne konvergencije, dok prečesta mutacija usporava pretragu.

Matematički gledano, govoreći za binarno kodiranje, mutacija je invertovanje jednog bita. Ako je bit imao vrednost jedan, posle mutacije će imati vrednost nula i obrnuto. Ova vrsta mutacije se još zove prosta mutacija. Ako razmatramo kod koji nije binarni, mutacije može da se implementira kao menjanje mesta dvema karaktera u okviru koda. Karatker *i* dobije vrednost karatkera *j*, i obrnuto, karakter *j* dobije vrednost *i.*

Treba još spomenuti genetske operatore mutaciju pomoću binomne raspodele i mutaciju pomoću normalne raspodele, koje su opisane u radovima [61, 105].

4.3.5. Kriterijum zaustavljanja

Genetski algoritmi mogu raditi beskonačno dugo, ukoliko nemaju kriterijum zaustavljanja. Kriterijumi zaustavljanja koji se najčešće koriste su dostignut maksimalan broj generacija, nivo sličnosti jedinki u populaciji, ponavljanje optimalne jedinke određen broj puta, dostizanje optimalnog rešenja ako je ono unapred poznato, ograničeno vreme trajanja genetskog algoritma. Najbolje je koristiti kombinaciju više različitih kriterijuma zaustavljanja, jer se tako sprečava neadekvatno zaustavljanje genetskog algoritma.

4.3.6. Parametri genetskog algoritma

Parametri igraju veoma važnu ulogu prilikom implementacije genetskog algoritma. Parametri imaju fiksno postavljene vrednosti, ako su podešeni pre početka izvršavanja genetskog algoritma. U suprotnom, imamo adaptivne parametre. Vrednosti adaptivnih parametera sa menjaju na osnovu toga kako utiču na rad genetskog algoritma. Veličina parametara zavisi od veličine problema koji se rešava genetskim algoritmom. Praksa je pokazala da se fiksno postavljeni parametri skoro i ne koriste. Najčešće korišćene vrednosti parametara su:

- $0{,}005 \leq$ *verovatnoća mutacije* $\leq 0{,}01$
- $0{,}8 \leq$ *verovatnoća ukrštanja* $\leq 0{,}95$
- 20 (nekad 50) $\leq$ *veličina populacije* $\leq$ 30 (150)

Ipak, treba napomenuti da je za najbolje performanse algoritma za konkretan problem neophodno izvršiti preliminarne testove i utvrditi vrednosti parametara za koje genetski algoritam daje najbolje rezultate.

4.3.7. Politika zamene generacija

Politika zamene generacija je jedan od najbitnijih delova genetskog algoritma. Postoji nekoliko ideja o zameni generacije, gde svaka ima prednosti i mane. Politike zamene generacija koje se najčešće koriste su generacijska (Generational), stacionarna (Steady-State) i elitistička politika (Elitist strategy). Genetski algoritam sa generacijskom strategijom zamene generacija u svakoj generaciji vrši promenu čitave populacije potomcima, tj. svi roditelji su zamenjeni svojim potomcima. Odmah upada u oči da je moguć scenario u kojem lošiji potomak istiskuje boljeg roditelja, što nije dobro za dalji rad genetskog algoritma. Stacionarni genetski algoritam generiše samo deo populacije u svakoj novoj generaciji, a preostale jedinke se prenose iz prethodne generacije. Mana ovog algoritma je što se ne proveravaju

jedinke koje se prenose iz jedne u drugu generaciju. Elitistička strategija zamene generacija omogućava genetskom algoritmu da jedna ili više elitnih jedinki prođe direktno u narednu generaciju. Na elitne jedinke se ne primenjuju genetski operatori selekcije, ukrštanja i mutacije. Takođe, za njih se ne računa vrednost funkcije prilagođenosti. Očigledno je da je elitistička strategija mnogo bolja od generacijske i stacionarne strategije, jer obezbeđuje da se najbolje jedinke uvek uzimaju u obzir kao kandidat za najbolje rešenje u datom trenutku.

Genetski algoritmi se koriste za rešavanje problema za čije rešavanje egzaktnim metodama je potrebno previše vremena. Ti problemi su NP-teški problemi za koje se ne zna postoji li algoritam koji ih rešava u polinomijalnom vremenu. Genetski algoritmi su doživeli veliku popularnost zbog mogućnosti paralelizacije, što je dobilo na značaju tek sa razvojem paralelnog programiranja. Još jedna prednost genetskih algoritama je jednostavna implementacija. Glavna mana genetskih algoritama je dugotrajno vreme izvršavanja u poređenju sa velikim brojem poznatih heuristika. Za ubrzavanje genetskog algoritma može da se koristi keširanje, o čemu će biti više reči kasnije.

4.4. Implementacija genetskog algoritma za rešavanje HMLC problema

4.4.1. Kodiranje

Svako rešenje je predstavljeno strukturom dužine N. Ta struktura se sastoji od redom poređanih promenljivih x_i, y_j *i* u^l_{ij}. Na odgovarajućoj poziciji promenljive y_j je vrednost nula ili jedan, u zavisnosti da li je hab uspostavljen ili nije. Na odgovarajućoj poziciji promenljive x_i nalazi se podatak o količini protoka koju fabrika *i* šalje lokaciji *j*. Na odgovarajućoj poziciji promenljive u^l_{ij} nalazi se podatak o broju modula *l* koji su uvedeni na putanji od fabrike *i* do lokacije *j*. Primetno je da je struktura nehomogena, tj. da nije samo binarna. Funkcija cilja se računa posebnom metodom koja koristi podatke iz gorenavedenog koda i podatke koji su učitani iz instance.

4.4.2. Operator selekcije

Predloženi genetski algoritam koristi fino-gradiranu turnirsku selekciju. Parametar ove turnirske selekcije je realna promenljiva koja predstavlja srednju veličinu turnira *TourLen* (Tournament Length). Element populacije je izabran ako ima bolju funkciju prilagođenosti od svojih proizvoljno odabranih konkurenata . Operator fino-gradirane turnirske selekcije koristi dva tipa turnira. Prvi turnir se održava *l1* puta i veličina tog turnira je [*TourLen*] + 1. Drugi tip turnira se odigrava *l2* puta i veličina tog tipa turnira je *[TourLen]*. U ovoj implementaciji *TourLen* ima vrednost 5.4 *(TourLen = 5,4)* a vrednosti za *l1* i *l2* su 20 i 30. Detaljniji opis fino-gradirane turnirske seleckije prikazan je u radu [30].

4.4.3. Operator ukrštanja

U implementaciji ovog algoritma koristi se jednopoziciono ukrštanje. Verovatnoća ukrštanja je 0.80, što znači da će u 80% slučajeva doći do operacija ukrštanja (*p = 0.80*). U preostalih 20%

slučajeva, kada se ne dogodi ukrštanje, prelazi se odmah na genetski operator mutacije. Prilikom ove implementacije ukrštanja bira se N / 2 parova jedinki, gde je N veličina populacije. Posle toga se za svaki par jedinki bira proizvoljna tačka ukrštanja i vrši se ukrštanje genetskog materijala kao što je opisano u uvodnom delu.

4.4.4. Operator mutacije

Jedinke dobijene nakon primene genetskog operatora ukrštanja podležu operatoru mutacije. U ovoj implementaciji koristi se posebna vrsta mutacije, mutacija sa zaleđenim bitovima. Tokom izvršavanja genetskog algoritma može doći do situacije da puno (u ekstremnom slučaju sve) jedinki ima istu vrednost na istoj poziciji (zaleđena pozicija, zaleđeni bit). To znači da operatori selekcije i ukrštanja na toj poziciji ne mogu da promene vrednost promenljive, što znači da će pretraživački prostor biti znatno smanjen. Primenom mutacije na tim pozicijama proširujemo pretraživački prostor. To se postiže time što se verovatnoća mutacije na tim pozicijama povećava nekoliko puta. Bitno je napomenuti da se verovatnoća mutacije ne sme povećati za sve pozicije, jer bi posledica toga bila da se algoritam pretvori u običan algoritam slučajne pretrage (random search). U ovoj implementaciji verovatnoća mutacije na standardanoj poziciji u kodu je 0.05 (p = 0,05), a na zaledjenim pozicijama je 0.20 (p = 0,20).

4.4.5. Politika zamene generacija

Početna populacija ima 160 jedinki, slučajno je generisana obezbeđujući time maksimalnu raznovrsnost genetskog materijala. Implementirana je elitistička strategija zamene generacija. Polovina populacije se menja, dok druga polovina prelazi direktno iz generacije u generaciju. Time se štedi na vremenu u svakoj sledećoj generaciji i doprinosi se prenošenju kvalitetnog genetskog materijala u sledeću generaciju. Ovo je veoma efikasna metoda za čuvanje raznolikosti genetskog materijala i sprečava preuranjenu konvergenciju .

4.4.6. Keširanje

Genetski algoritmi imaju duže vreme izvršavanja u odnosu na druge metode kombinatorne optimizacije, pa bi bilo dobro da se oni na neki način ubrzaju. Za te svrhe koriste algoritam keširanja jer doprinosi velikoj uštedi vremena. Genetski algoritam veći deo vremena potroši na izračunavanje vrednosti funkcije cilja datog problema, što u zavisnosti od problema može biti jako komplikovano. Izračunata vrednost se zatim smešta u memoriju. Keširanje funkcioniše tako što se pre smeštanja vrednosti u memoriju proverava da li se ona nalazi u keš memoriji. Ako se nalazi u keš memoriji, vrednost funkcije cilja se ne računa ponovo već se poziva iz keš memorije. Ako se vrednost ne nalazi u keš memoriji, vrednost funkcije cilja se izračunava i zajedno sa genetskim kodom smešta u keš memoriju.

Keš memorija sadrži blokove. Svaki blok ima funkciju skladištenja jedinke i, eventualno, direktno očitavanje vrednosti te jedinke. Veličina keš memorije, tj. broj blokova od kojih se ona sastoji je ograničen. Ako se keš memorija napuni izbacuje se najstariji nekorišćeni element i skladišti se novi (Least Recently Used Strategy - LRUS).

4.4.7. Kriterijum zaustavljanja

Kriterijum zaustavljanja algoritma je ispunjen ako je ispunjen jedan od sledeća dva uslova. Prvi uslov je prekoračenje broja generacija. U slučaju ovog problema maksimalan broj generacija je postavljen da bude deset hiljada. Drugi uslov je ponavljanje najboljeg rešenja u poslednjih 200 generacija.

4.5. Predloženi memetski algoritam za rešavanje problema HMLC

Suštinska razlika genetskog algoritma i algoritma tabu pretraživanja je što su genetski algoritmi populacijski algoritmi, a tabu pretraživanje radi nad jednim rešenjem. Glavna mana tabu pretraživanja, šire gledano lokalne pretrage, je što može doći do ignorisanja atraktivnih delova pretraživačkog prostora. Pošto genetski algoritmi razmatraju čitave populacije rešenja, oni nemaju problem ignorisanja zanimljivih delova pretraživačkog prostora. Nedostatak genetskih algoritama je što u određenim sitacijama ne mogu da dostignu optimalno rešenje. Takođe, porastom dimenzije problema koji rešava genetski algoritam, raste i vreme izvršavanja koje može da bude nedopustivo veliko.

Zbog navedenih mana genetskog algoritma i algoritma tabu pretraživanja, intuitivno se nameće njihova hibridizacija kao način da se reše ti nedostaci. Mogući pristup hibridizaciji je poboljšanje kvaliteta jedinki populacije primenom tabu pretraživanja. Nakon generisanja svake generacije algoritam tabu pretraživanja može se iskoristiti kao operator genetskog algoritma. Tabu algoritam može da se iskoristi kao genetski operator mutacije. Jedinke nove populacije koriste se kao početna rešenja, pa se unapređuju tabu pretragom [67]. Prilikom hibridizacije genetskog algoritma i tabu pretraživanja vreme izvršavanja tabu pretraživanja se ograničava jer bi u suprotnom došlo do značajnog povećanja vremena izvršavanja celokupnog algoritma. Još jedna primena tabu pretraživanja kao operatora genetskog algoritma je poboljšanje svih jedinki nove populacije [103].

U prethodnom odeljku je opisano kako se tabu pretraživanje može iskoristiti u algoritmu čiji je nosilac genetski algoritam. Moguća je i obrnuta slika, da tabu pretraživanje čini kostur hibridnog algoritma, ali se takva hibridizacija retko koristi. Jedan od načina implementacije takvog algoritma je da se intenzifikacija tabu pretraživanja implementira kao genetski algoritam. Početna populacija tog algoritma je skup elitnih rešenja koja su posećena tokom pretrage. Bitno je da takva intenzifikacija, odnosno u ovom slučaju genetski algoritam, ima male zahteve za vremenskim i memorijskim resursima. U supotrom može doći do značajnog smanjenja efikasnosti tabu pretraživanja. Cilj ovog algoritma je dobijanje hibridizacije koja ne bi imala manu tabu pretraživanja da eventualno zaobiđe čitave delove pretraživačkog prostora.

U ovom radu korišćen je koncept memetskog algoritma za hibridizaciju predloženih heuristika tabu pretraživanja i genetskog algoritma opisanih u prethodnim podsekcijama. Memetski algoritam koristi tabu pretraživanje za poboljšanje kvaliteta rešenja. Tabu pretraživanje se izvršava u svakoj generaciji algoritma, pre primene genetskih operatora selekcije, ukrštanja i mutacije. Tabu pretraživanje se primenjuje samo na najbolju jedinku i to samo ako se ta jedinka promenila u poslednjoj iteraciji genetskog algoritma. Ako se nije promenila u prethodnoj iteraciji genetskog algoritma, znači da je tabu pretraživanje već primenjeno tu jedinku u prošloj iteraciji genetskog algoritma, stoga ne mora ponovo. Tabu pretraživanje je deterministička metaheuristika, tako da ako ne može da poboljša najbolju jedinku u tekućoj generaciji, onda ne može dati nikakvo poboljšanje.

Implementacija memetskog algoritma za problem HMLC se realizuje na sledeći način. U

svakoj generaciji, pre nego što se primene genetski operatori selekcije, ukrštanja i mutacije, algoritmom tabu pretraživanja ćemo pokušati da poboljšamo rešenje.Rešenje se poboljšava tako što se najboljoj jedinki na nekom čvoru promeni status. Konkretno, ako je na određenoj poziciji hab čvor on postaje ne-hab čvor i obrnuto. Posle svake promene statusa algoritam računa vrednost funkcije cilja. Ako je vrednost funkcije cilja bolja nego pre promene statusa čvora, nastavljamo dalje dok god jedinka može da se poboljša. Ako se vrednost funkcije cilja nije promenila, nastavlja se dalje sa genetskim algoritmom. Algoritmom 6 prikazan je pseudokod memetskog algoritma.

Algoritam 6. Memetski algoritam za rešavanje HMLC problema

1. *UčitavanjeUlaznihPodataka();*
2. *pop* ← *GenerisanjePočetnePopulacije();*
3. while (!*KriterijumZaustavljanjaGenetskogAlgoritma()*)
4. for *i* ← od 1 do *veličinaPopulacije*
5. *IzračunajFunkcijuCilja(i);*
6. *TabuPretraživanje(i);*
7. izlazak iz for petlje
8. *Prilagođenost()*;
9. *OperatorSelekcije();*
10. *OperatorUkrštanja();*
11. *OperatorMutacije();*
12. izlazak iz while petlje
13. vratiti rešenje;

U opštem slučaju genetski algoritam propušta dobre jedinke u sledeću generaciju da bi se njihovim ukrštanjem dobile bolje jedinke. Međutim, može se desiti da dobre jedinke dominiraju populacijom tako što izbacuju jedinke koje imaju lošiju vrednost funkcije cilja, ali koje imaju dobre gene. Na taj način će doći do višestrukih pojava dobrih jedinki, što bismo hteli da izbegnemo zarad raznovrsnosti genetskog materijala. Fizička eliminacija višestrukih jedinki se obavlja relativno sporo, pa se zbog toga izbegava. Rešenje je markirati jedinke u tekućoj populaciji, zatim ih ukloniti u sledećoj generaciji. Markiranje se vrši dodeljivanjem nulte vrednosti za prilagođenost jedinke. Pri izboru jedinki za sledeću generaciju, operatero selekcije ih uopšte neće izabrati jer imaju nisku (u ovom slučaju nultu) prilagođenosti.

5. Eksperimentalni rezultati

Za rešavanje datog problema napisana su tri programa. Prvi program je implementacija tabu algoritma. Drugi program je implementacija genetskog algoritma. Treći program je implementacija memetskog algoritma. Sva tri programa napisana su u programskom jeziku C.

Svi eksperimenti su izvršeni na personalnom računaru koji koristi dualni procesor "AMD Athlon(tm) X2 250" sa 2,9 GHz i 3 GB RAM memorijom. Operativni sistem je Linux 10.04 (lucid). Predloženi programi koji rešavaju problem HMLC testirani su uz pomoć instanci AP, koje su generisane za ovaj problem. Instance AP (Australian Post) su zasnovane na realnim podacima dobijenih iz australijskog poštanskog sistema. Prvi put su korišćene od strane autora Ernst i Krishnamoorthy 1996. godine. Najveće instance imaju do dve stotine (200) čvorova.

U Tabeli 1 prikazani su rezultati predložene heuristike tabu pretraživanja na skupu AP instanci. Rezultati su prezentovani na sledeći način. U prvoj koloni se unosi naziv testirane instance. Na svakoj instanci algoritam je testiran 20 puta. Naziv instance dat je u obliku *AP_BrC_X_Y_Z_T_U_V*, pri čemu AP označava tip instance, *BrC* predstavlja dimenziju instance, a ostali podaci se odnose na vrednosti fiksnih cena. U drugoj koloni unosimo vrednosti fje cilja *Opt* koje odgovaraju optimalnom rešenju, koje je dobijeno primenom strateške oscilacije (Strategic Oscilation) u radu [18]. U trećoj koloni se nalazi vrednost f_{best} koja predstavlja najbolju vrednost funkcije cilja od dvadeset dobijenih vrednosti funkcije cilja. U četvrtoj koloni je prikazano vreme t_{best}. Vreme t_{best} predstavlja prosečno vreme (izraženo u sekundama) potrebno da bi se dobio najbolji rezultat tabu algoritma. Prosečno ukupno vreme izvršavanja programa je označeno sa t_{total} (u sekundama) i ono je prikazano u petoj koloni tabele. U poslednjoj, šestoj koloni, nalazi se prosečna relativna greška *Gr* (izražena u procentima) koja se računa u odnosu na optimalno rešenje.

U Tabeli 2 su prikazani eksperimentalni rezultati genetskog algoritma. Raspored kolona u sledećoj tabeli isti je kao raspored kolona u Tabeli 1. Program koji implementira genetski algoritam će se izvršavati nad istim instancama nad kojima je izvršavan program koji implementira algoritam tabu pretraživanja. Kao i u prethodnoj glavi, program će biti pokrenut dvadeset puta za svaku instancu.

U Tabeli 3 su prikazani eksperimentalni rezultati memetskog algoritma. Raspored kolona u sledećoj tabeli isti je kao raspored kolona u Tabeli 1 i Tabeli 2. Program koji implementira memetski algoritam će se izvršavati nad istim instancama nad kojima je izvršavan program koji implementira algoritam tabu pretraživanja i genetski algoritam. Kao i u slučaju prethodnih algoritama, memetski algoritam je testiran dvadeset puta za svaku instancu.

Tabela 4 predstavlja poređenje algoritma tabu pretraživanja, genetskog algoritma i memetskog algoritma predloženih za rešavanje problema HMLC. U prvoj koloni je navedeno ime instance u skraćenom obliku. Instance su navedene istim redom kao u prve tri tabele. U drugoj koloni unosimo vrednosti fje cilja *Opt* koje odgovaraju optimalnom rešenju. U trećoj koloni se nalazi vrednost f_{best} za algoritam tabu pretraživanja. U četvrtoj koloni se nalazi t_{best} za algoritam tabu pretraživanja (izraženo u sekundama). U petoj koloni se nalazi f_{best} za genetski algoritam. U šestoj koloni se nalazi t_{best} za genetski algoritam (izraženo u sekundama). U sedmoj koloni se nalazi f_{best} za memetski algoritam i u osmoj koloni je t_{best} za memetski algoritam (izraženo u sekundama).

Tabela 1. Rezultati dobijeni metodom tabu pretraživanja

Instanca	*Opt*	f_{best}	t_{best} (s)	t_{total} (s)	*Gr* (%)
AP_10_700_50_60_8_1_60	243 854	243 854	0,005	0,011	0
AP_10_800_60_60_6_1_69	245 513	245 513	0,009	0,0018	0,15
AP_20_700_50_60_8_1_60	393 788	393 788	0,0043	0,003	0,33
AP_20_700_69_40_8_1_50	417 187	417 187	0,007	0,021	0,22
AP_20_800_60_80_8_1_80	368 951	368 951	0,015	0,031	0,59
AP_45_600_80_89_8_1_60	536 953	536 960	0,028	0,062	0,28
AP_60_500_60_69_60_1_50	580 361	580 363	0,075	0,202	0,38
AP_60_800_69_50_80_1_60	656 894	656 894	0,038	0,093	0,24
AP_70_600_60_69_60_1_69	607 509	605 511	0,041	0,101	0,44
AP_85_500_60_69_60_1_50	772 714	772 715	0,050	0,123	0,75
AP_90_600_60_69_60_1_69	810 722	810 723	0,061	0,159	0,42
AP_95_600_60_69_60_1_69	807 075	807 076	0,057	0,117	0,38
AP_120_500_60_69_60_1_50	1 061 782	1 061 832	0,890	2,290	0,24
AP_130_600_60_69_60_1_69	1 126 476	1 126 499	0,771	1,921	0,38
AP_155_900_69_50_89_1_60	1 769 092	1 769 145	0,950	2,859	0,34
AP_190_600_60_69_60_1_69	821 318	821 386	5,006	18,274	0,34
AP_195_600_60_69_60_1_69	713 616	713 635	6,525	22,562	0,11
AP_200_800_69_50_80_1_60	763 349	763 386	7,589	19,033	0,22

Tabela 2. Rezultati dobijeni primenom genetskog algoritma

Instanca	*Opt*	f_{best}	t_{best} (s)	t_{total} (s)	*Gr* (%)
AP_10_700_50_60_8_1_60	243 854	243 854	0,001	0,006	0
AP_10_800_60_60_6_1_69	245 513	245 513	0,001	0,007	0
AP_20_700_50_60_8_1_60	393 788	393 788	0,001	0,021	0
AP_20_700_69_40_8_1_50	417 187	417 187	0,001	0,025	0
AP_20_800_60_80_8_1_80	368 951	368 951	0,001	0,037	0
AP_45_600_80_89_8_1_60	536 953	536 953	0,012	0,123	0
AP_60_500_60_69_60_1_50	580 361	580 361	0,011	0,301	0
AP_60_800_69_50_80_1_60	656 894	656 925	0,026	0,479	0,005
AP_70_600_60_69_60_1_69	607 509	610 590	0,049	1,423	0,005
AP_85_500_60_69_60_1_50	772 714	773 851	0,102	2,466	0,505
AP_90_600_60_69_60_1_69	810 722	811 024	0,101	2,465	0,037
AP_95_600_60_69_60_1_69	807 075	808 845	0,258	4,218	0,219
AP_120_500_60_69_60_1_50	1 061 782	1 062 951	0,091	4,312	0,110
AP_130_600_60_69_60_1_69	1 126 476	1 129 845	0,427	9,876	0,298
AP_155_900_69_50_89_1_60	1 769 092	1 772 102	1,745	19,876	0,170
AP_190_600_60_69_60_1_69	821 318	824 399	4,132	39,650	0,373
AP_195_600_60_69_60_1_69	713 616	717 451	5,136	47,401	0,534
AP_200_800_69_50_80_1_60	763 349	766 872	6,988	61,300	0,460

Tabela 3. Rezultati dobijeni primenom memetskog algoritma

Instanca	*Opt*	f_{best}	t_{best} (s)	t_{total} (s)	*Gr* (%)
AP_10_700_50_60_8_1_60	243 854	243 854	0,001	0,012	0
AP_10_800_60_60_6_1_69	245 513	243 854	0,002	0,012	0
AP_20_700_50_60_8_1_60	393 788	393 788	0,001	0,040	0
AP_20_700_69_40_8_1_50	417 187	417 187	0,002	0,048	0
AP_20_800_60_80_8_1_80	368 951	368 951	0,001	0,072	0
AP_45_600_80_89_8_1_60	536 953	536 953	0,016	0,247	0
AP_60_500_60_69_60_1_50	580 361	580 362	0,018	0,610	0,11
AP_60_800_69_50_80_1_60	656 894	656 894	0,036	0,898	0
AP_70_600_60_69_60_1_69	607 509	607 509	0,079	1,423	0
AP_85_500_60_69_60_1_50	772 714	772 716	0,151	2,466	0,10
AP_90_600_60_69_60_1_69	810 722	810 722	0,144	2,465	0
AP_95_600_60_69_60_1_69	807 075	807 076	0,447	4,218	0,09
AP_120_500_60_69_60_1_50	1 061 782	1 061 785	0,201	4,312	0,25
AP_130_600_60_69_60_1_69	1 126 476	1 126 488	0,891	9,876	0,39
AP_155_900_69_50_89_1_60	1 769 092	1 769 098	3,075	19,876	0,06
AP_190_600_60_69_60_1_69	821 318	821 322	5,334	39,650	0,17
AP_195_600_60_69_60_1_69	713 616	713 625	6,596	47,401	0,28
AP_200_800_69_50_80_1_60	763 349	763 358	8,173	61,300	0,09

Tabela 4. Poređenje rezultata

Instance	Opt	TS f_{best}	TS t_{best} (s)	TS GR(%)	GA f_{best}	GA t_{best}(s)	GA GR(%)	MA fbest	MAt_{best}(s)	MA GR(%)
AP_10a	243 854	243 854	0,005	0	243 854	0,001	0	243 854	0,001	0
AP_10b	245 513	245 513	0,009	0,15	245 513	0,001	0	243 854	0,002	0
AP_20a	393 788	393 788	0,0043	0,33	393 788	0,001	0	393 788	0,001	0
AP_20b	417 187	417 187	0,007	0,22	417 187	0,001	0	417 187	0,002	0
AP_20c	368 951	368 951	0,015	0,59	368 951	0,001	0	368 951	0,001	0
AP_45	536 953	536 960	0,028	0,28	536 953	0,012	0	536 953	0,016	0
AP_60a	580 361	580 363	0,075	0,38	580 361	0,011	0	580 362	0,018	0,11
AP_60b	656 894	656 894	0,038	0,24	656 925	0,026	0,005	656 894	0,036	0
AP_70	607 509	605 511	0,041	0,44	610 590	0,049	0,005	607 509	0,079	0
AP_85	772 714	772 715	0,050	0,75	773 851	0,102	0,505	772 716	0,151	0,10
AP_90	810 722	810 723	0,061	0,42	811 024	0,101	0,037	810 722	0,144	0
AP_95	807 075	807 076	0,057	0,38	808 845	0,258	0,219	807 076	0,447	0,09
AP_120	1 061 782	1 061 832	0,890	0,24	1 062 951	0,091	0,110	1 061 785	0,201	0,25
AP_130	1 126 476	1 126 499	0,771	0,38	1 129 845	0,427	0,298	1 126 488	0,891	0,39
AP_155	1 769 092	1 769 145	0,950	0,34	1 772 102	1,745	0,170	1 769 098	3,075	0,06
AP_190	821 318	821 386	5,006	0,34	824 399	4,132	0,373	821 322	5,334	0,17
AP_195	713 616	713 635	6,525	0,11	717 451	5,136	0,534	713 625	6,596	0,28
AP_200	763 349	763 386	7,589	0,22	766 872	6,988	0,460	763 358	8,173	0,09

Iz Tabele 1 mogu da se izvuku korisni zaključci. Može se primetiti da je algoritam dostigao optimum u većini slučajeva. U slučajevima gde algoritam nije dostigao optimum, rezultat je bio dovoljno blizu optimuma. Na osnovu prosečnog vremena za dostizanje najboljeg posećenog rešenja zaključujemo da algoritam radi na efikasan način, i stiže brzo do najboljeg posećenog rešenja.

Na osnovu eksperimentalnih rezultata može se videti da preciznost algoritma generalno ne opada sa porastom dimenzija. U nekim instancama je zabeležena je velika nekonzistencija (instanca AP_85_500_60_69_60_1_50), a mogući uzrok je veliki uticaj diversifikacije. Negativno svojstvo diversifikacije je da može da pogorša kvalitet tekućeg rešenja. Primenjena je diversifikacija restarta.

Kao što se može videti iz Tabele 2, za veliki broj instanci program postiže optimalno ili skoro pa optimalno rešenje. Najmanje prosečno ukupno vreme imaju najmanje instance, dok najveće prosečno ukupno vreme imaju najveće instance. Prosečna relativna greška u dva slučaja dostiže vrednost veću od 0,5 %, ostale su manje od toga, što govori u prilog konzistentsnosti rešenja. Ako uporedimo prosečno ukupno vreme izvršavanja algoritma tabu pretraživanja i genetskog algoritma, zaključujemo da je ta vrednost kod nekih instanci i po nekoliko puta veća kod genetskog nego kod tabu algoritma.

Kao što se može videti iz Tabele 3, za veliki broj instanci program postiže optimalno ili skoro pa optimalno rešenje. Najmanje prosečno ukupno vreme imaju najmanje instance, dok najduže prosečno ukupno vreme imaju najveće instance. Prosečna relativna greška skoro nikad nije veća od 0,25%, sem u jednoj situaciji gde je njena vrednost skoro 0,40%.

Zaključci do kojih se dolazi na osnovu Tabele 4 je da su sva tri algoritma postigli optimalne ili skoro optimalne rezultate za zadovoljavajuće vreme. Vreme dostizanja najboljeg rešenja je najveće kod memetskog algoritma, jer memetski algoritam u sebi sadrži genetski algoritam i tabu pretraživanje.

6. Zaključak

U ovom master radu opisani su algoritmi tabu pretraživanja, genetski algoritam i memetski algoritam za rešavanje problema HMLC, pošto je problem HMLC težak za rešavanje, pa postojeće egzaktne metode ne postižu egzaktna rešenja sem kod problema malin dimenzija.

Algoritam tabu pretraživanja je predložen za rešavanje problema HMLC. Problem je podeljen na lokacijski potproblem i potproblem dodeljivanja. Lokacijski potproblem rešen je primenom tabu pretraživanja. Rezultat rada tabu pretraživanja je sortirani skup habova. Nad najboljim skupovima hab čvorova izvršena je lokalna pretraga koja se zasniva na strategiji prvog poboljšanja (First Improvement Strategy).

Drugi algoritam koji je predložen za rešavanje problema HMLC je genetski algoritam koji koristi operatore fino-gradirane turnirske selekcije, operator jednopozicionog ukrštanja, kao i specijalni oblik operatora mutacije sa zaleđenim genima. Primenjena je elitistička politika zamene generacija koja prenosi veliki deo populacije u narednu generaciju.

Memetski algoritam je treći algoritam koji je upotrebljen za rešavanje problema HMLC. Predloženi memetski algoritam je hibiridizacija tabu pretraživanja i genetskog algoritma. Hibiridizacija algoritma tabu pretraživanja i genetskog algoritma uzima kao kostur genetski algoritam, dok se tabu pretraživanje koristi kao sastavni deo tog genetskog algoritma. Tabu pretraživanje se primenjuje u svakoj generaciji pre primene genetskih operatora selekcije, ukrštanja i mutacije.

Implementacije sva tri predložena algoritma za rešavanje problema HMLC testirane su nad istim instancama koje pripadaju grupi instanci koje sadrže podatke australijskog poštanskog sistema. Svi algoritmi su postigli optimalno ili rešenje blisko optimalnom u relativno kratkom vremenu uzvršavanja.

Analizom ekperimentalnih rezultata može se zaključiti da je implementacija memetskog algoritma dala bolje rezultate od tabu pretraživanja i genetskog operatora.

Buduća istraživanja se mogu odvijati u više pravaca :

- Paralelizacija algoritma na računarima sa više jezgara, dodatna pogodnost je korišćene turnirske selekcije
- Memetski algoritam može da iskoristi neki drugi genetski algoritam kao kostur. Genetski algoritmi nikada nisu jedan algoritam, već porodica algoritama, tako da se mogu razmatrati razne varijacije.
- Opisani memetski algoritam može se modifikovati tako da rešava slične probleme kombinatorne optimizacije

Literatura

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

[2] Abdinnour-Helm S., Venkataramanan M., “Solution approaches to hub location problems“, Annals of Operations Research 78, 31-50, (1998).

[3] Alumur, S., Kara, B.Y., "Network hub location problems: The state of the art", European Journal of Operational Research (2007).

[4] Aykin T., "Lagrangean relaxation based approaches to capacitated hub-and-spoke network design problem", European Journal of Operational Research 79 (3), 501–523 (1994).

[5] Bambha N.K., Bhattacharyya S.S., Teich J., Zitzler E.: "Systematic integration of parameterized local search into evolutionary algorithms", IEEE Transactions on Evolutionary Computation 8(2), 137–155 (2004).

[6] Beasley D., Bull D.R., Martin R.R., "An Overview of Genetic Algorithms, Part 1, Fundamentals", University Computing, Vol. 15, No. 2, pp. 58-69 (1993).

[7] Beasley D., Bull D.R., Martin R.R., "An Overview of Genetic Algorithms, Part 2, Research Topics", University Computing, Vol. 15, No. 4, pp. 170-181 (1993).

[8] Berretta R., Cotta C., Moscato P.:"Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm: Results on the number partitioning problem. In: Resende,M., Pinho de Sousa, J. (eds.) Metaheuristics: Computer-Decision Making, pp. 65–90. Kluwer Academic Publishers, Boston (2003).

[9] Boland N., Ebery J., Ernst A., Krishnamoorthy M., "The capacitated multiple allocation hub location problem: formulation and algorithms". European Journal of Operational Research 120:614–31 (2000).

[10] Buriol L., Franca, P., Moscato P.: "A new memetic algorithm for the asymmetric traveling salesman problem", Journal of Heuristics 10(5), 483–506 (2004).

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

[12] Canovas L., Garcia S., Marin A., “Solving the uncapacitated multiple allocation hub location problem by means of dual-ascent technique“, Working paper no 1, Departamento de Estadısticae Investigacion Operativa (2004).

[13] Carello G., Della Croce F., Ghirardi M., Tadei R., "Solving the hub location problem in telecommunication network design:a local search approach", (2004).

[14] Chen J. F., “A hybrid heuristic for the uncapacitated single allocation hub location problem“, OMEGA The Internacional Journal of Menagement Science, 35: 211-220, (2007).

[15] Cobb H., Grefenstette J.: "Genetic algorithms for tracking changing environments", Forrest, S. (ed.) ICGA, pp. 529–530. Morgan Kaufmann, San Mateo (1993).

[16] Contreras I., Dias J., Fernandez E., “Lagrangean relaxation for the capacitated hub location problem with single assigment”, OR Spectrum 31: 483-505, (2009).

[17] Contreras I., "Location Science", Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, QC, Canada, (2015).

[18] Corberan A., Peiro J., Campos V., Glover F., Marti R., "Strategic Oscillation for the capacitated hub location problem with modular links", (2014).

[19] Correia I., Captivo M.E., "A Lagrangean heuristic for a modular capacitated location problem". Annals of Operations Research 122, 141–161 (2003).

[20] Correia I., Gouveia L., Saldanha-da-Gama F., "Discretized reformulations for a capacitated networ k loading problem arising in a facility location context", Europeatn Journal of Operational Research (2010).

[21] Costa M. da G., Captivo M.E., Climaco J., "Capacitated single allocation hub location problem - A bi-criteria approach", Computers & Operations Research 32:3671-3695, (2008).

[22] Cotta C., Fernandez A., "Memetic algorithms in planning, scheduling, and timetabling". Dahal, K., Tan, K., Cowling, P. (eds.) Evolutionary Scheduling. Studies in Computational Intelligence, vol. 49, pp. 1–30. Springer, Heidelberg (2007).

[23] Davidor Y., Ben-Kiki O., "The interplay among the genetic algorithm operators: Information theory tools used in a holistic way", Conference: Parallel Problem Solving from Nature 2, pp. 75–84 (1992).

[24] De Jong K.E., "An analysis of the behavior of a class of genetic adaptive systems", PhD thesis, University of Michigan (1975).

[25] Dias J., Captivo M., Climaco J., "A memetic algorithm for multi-objective dynamic location problems". Journal of Global Optimization 42(2), 221–253 (2008).

[26] Drezner, Z. "The p-cover problem", European Journal of Operational Research, 26:312-313 (1986)

[27] Ernst A.T., Jiang H., Krishnamoorthy M., "Reformulations and computational results for uncapacitated single and multiple allocation hub covering problems", (2005).

[28] Ernst A.T., Krishnamoorthy M., “An exact solution approach based on shortestpaths for p-hub median problems“, INFORMS Journal on Computing, 10 (2):149-162, (1998).

[29] Ernst A.T., Krishnamoorthy M., "Solution algorithms for the capacitated single allocation hub location problem", Annals of Operations Research 86, 141–159, (1999).

[30] Filipović V. "Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama", Magistarski rad, Univerzitet u Beogradu, Matematički fakultet (1998).

[31] Filipović V., Kratica J., Tošić D., Ljubić I., "Fine Grained Tournament Selection for the Simple Plant Location Problem", Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications - WSC5, pp. 152-158, (2000).

[32] Franca, P.M., Mendes, A., Moscato, P.: A memetic algorithm for the total tardiness single machine scheduling problem. European Journal of Operational Research 132(1), 224–242 (2001).

[33] Freisleben B., Merz P., "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems". Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, Nagoya, Japan, pp. 616–621. IEEE Press, Los Alamitos (1996).

[34] Gallardo J.E., Cotta C., Fernandez A.J., "Solving the multidimensional knapsack problem using an evolutionary algorithm hybridized with branch and bound", (2005).

[35] Gallardo J.E., Cotta C., Fernandez A.J., "On the hybridization of memetic algorithms with branch-and-bound techniques", IEEE Transactions on Systems, Man and Cybernetics, part B 37(1), 77–83 (2007).

[36] Gallard J.E., Cotta C., Fernandez A.J., "A memetic algorithm with bucket elimination for the still life problem", Gottlie J., Raidl G.R. (eds.) EvoCOP 2006. LNCS, vol. 3906, pp. 73–85. Springer, Heidelberg (2006).

[37] Glover, F., "Heuristics for integer programming using surrogate constraints", Journal of Applied Mathematics and Decision Sciences 8, 156–166 (1977).

[38] Glover F., "Tabu search – Part I", ORSA Journal on Computing 1, 190–206 (1989).

[39] Glover F., "Tabu search – Part II", ORSA Journal on Computing 2, 4–32 (1990).

[40] Glover F., "Future paths for integer programming and links to artificial intelligence". Computers & Operations Research 13, 533–549 (1986).

[41] Glover F., Taillard E. D., de Werra D., "A user's guide to tabu search", Annals of Operations Research 41, 3-28, (1993).

[42] Hart W., Krasnogor N., Smith J.E., "Memetic evolutionary algorithms", (2005).

[43] Holland J.H., "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor (1975).

[44] Ilić A., Urošević D., Brimberg J., Mladenović N., “Agenral variable neighborhood search for solving te uncapacitated single allocation p-hub median problem“, European Journal of Operational Research, 206: 289-300, (2010).

[45 Ishibuchi H., Murata T., "Multi-objective genetic local search algorithm", Fukuda, T., Furuhashi, T. (eds.) 1996 International Conference on Evolutionary Computation, pp. 119–124. IEEE Press, Nagoya (1996).

[46] Ishibuchi H., Murata T., "Multi-objective genetic local search algorithm and its application to flowshop scheduling", IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews 28(3), 392–403 (1998).

[47] Janikow C.Z., Michalewicz Z., "An Experimental Comparasion of Binary and Floating Point Representations in Genetic Algorithms", in: Proceedings of the Fourth International Conference on Genetic Algorithms - ICGA '91, Morgan Kaufmann, San Mateo, Calif., pp. 37-44 (1991).

[48] Jaszkiewicz A., "Genetic local search for multi-objective combinatorial optimization". European Journal of Operational Research 137, 50–71 (2002).

[49] Jaszkiewicz A., "A comparative study of multiple-objective metaheuristics on the biobjective set covering problem and the pareto memetic algorithm". Annals of Operations Research 131(1-4), 135–158 (2004).

[50] Karaoglu B., Topcuoglu H., Gurgen F., "Evolutionary algorithms for location area management". Rothlauf F., Branke J., Cagnoni S., Corne D.W., Drechsler R., Jin Y., Machado P., Marchiori E., Romero J., Smith G.D., Squillero G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 175–184. Springer, Heidelberg (2005).

[51] Karimi H., Bashiri M., "Hub covering location problems with different coverage types", Scienta Iranica E 18 (6), 1571-1578, (2011).

[52] Klau G., Ljubić I., Moser A., Mutzel P., Neuner P., Pferschy U., Raidl G., Weiskircher R., "Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem". pp. 1304–1315 (2004).

[53] Klincewicz J.G., "Heuristics for the p-hub location problem", European Journal of Operational Research, 53, 25-37, (1991).

[54] Klincewicz J.G., “A dual algorithm for the uncapacitated hub location problem“, Location Science, 4 (3): 173-84, (1996).

[55] Knowles J., Corne D., "A Comparison of Diverse Aproaches to Memetic Multiobjective Combinatorial Optimization". Wu, A.S. (ed.) Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program, pp. 103–108 (2000).

[56] Knowles J., Corne D., "M-PAES: A memetic algorithm for multiobjective optimization", (2000).

[57] Krasnogor N., "Self-generating metaheuristics in bioinformatics: The protein structure comparison case", Genetic Programming and Evolvable Machines 5(2), 181–201 (2004).

[58] Krasnogor N., Blackburne B., Burke E., Hirst J. "Multimeme algorithms for proteine structure prediction", (2002).

[59] Krasnogor N., Gustafson S., "A study on the use of “self-generation” in memetic algorithms", Natural Computing 3(1), 53–76 (2004).

[60] Krasnogor N., Smith J., "Memetic algorithms: The polynomial local search complexity theory perspective", Journal of Mathematical Modelling and Algorithms 7(1), 3–24 (2008).

[61] Kratica J., "Paralelizacija genetskih algoritama za rešavanje nekih NPkompletnih problema", Doktorska disertacija, Matematički fakultet, Beograd (2000).

[62] Labbe M, Yaman H, Gourdin E., "A branch and cut algorithm for the hub location problems with single assignment", Mathematical Programming, (2004).

[63] Lin S., Kernighan B., "An Effective Heuristic Algorithm for the Traveling Salesman Problem", Operations Research 21, 498–516 (1973).

[64] Lipowski A., Lipowska D., "Roulette-wheel selection via stochastic acceptance", Adam Mickiewicz University, Poland, (2011).

[65] Love R.F., Morris J.G., Wesolowsky O. G., "Facilities Location: Models and Methods", Publication in Operation Research vol. 7, North Holland, New York, (1988).

[66] Lozano M., Herrera F., Krasnogor N., Molina D., "Real-coded memetic algorithms with crossover hill-climbing". Evolutionary Computation 12(3), 273–302 (2004).

[67] Mak K. L., Sun, D., "A new hybrid Genetic Algorithm and Tabu Search method for Yard Cranes Scheduling with Inter-crane Interference", World Congress on Engineering 2009 Vol I, (2009) .

[68] Marin A., Canovas L., Landete M., “New formulation for the uncapacitated multiple allocation hub location problem“, European Journal of Operational Research, 172 (1): 274-292, (2006).

[69] Marić M., Stanimirović Z., Đenić A., Stanojević P., „Memetic Algorithm for Solving the Multilevel Uncapacitated Facility Location Problem”, Informatica, Vol. 25, No.3, pp. 439-466, (2014).

[70] Marić M., Stanimirović Z., Stanojević P., "An efficient memetic algorithm for the uncapacitated single allocation hub location problem", Soft Computing, 17:445–466 (2013).

[71] Mayer G., Wagner B., “Hublocater: An exact solution method for the multiple allocation hub location problem“, Computers & OR, 29: 715-739, (2002).

[72] Mendes A., Cotta C., Garcia V., Franca P., Moscato P., "Gene ordering in microarray data using parallel memetic algorithms", Skie, T., Yang, C.S. (eds.) 2005 International Conference on Parallel

Processing Workshops, pp. 604–611. IEEE Press, Oslo (2005).

[73] Mendes A.S., Franca P.M., Moscato P., "Fitness landscapes for the total tardiness single machine scheduling problem". Neural Network World 2(2), 165–180 (2002).

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

[75] Molina D., Herrera F., Lozano M., "Adaptive local search parameters for real-coded memetic algorithms", (2005).

[76] Molina D., Lozano M., Herrera F., "Memetic algorithms for intense continuous local search methods", (2008).

[77] Moscato P., Cotta C., "A gentle introduction to memetic algorithms", (2003).

[78] Moscato P., Cotta C., "Memetic algorithms", Gonzalez, T. (ed.) Handbook of Approximation Algorithms and Metaheuristics, ch. 22. Taylor & Francis, Abington (2006).

[79] Moscato P., Cotta C., "A modern introduction to memetic algorithms", Gendrau M., Potvin J.Y. (eds.) Handbook of Metaheuristics, 2nd edn. International Series in Operations Research and Management Science, vol. 146, pp. 141–183. Springer, New York (2010).

[80] Moscato P., Cotta C., Mendes A., "Memetic algorithms", (2004).

[81] Moscato P., Berretta R., Cotta C., "Memetic algorithms", Wiley Encyclopedia of Operations Research and Management Science. Wiley, Chichester (2011).

[82] Moscato P., Norman M., "A competitive and cooperative approach to complex combinatorial search", Tech. Rep. 790, Caltech Concurrent Computation Program (1989).

[83] Moscato P., "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms", Tech. Rep. 826, California Institute of Technology (1989).

[84] Nguyen Q.H., Ong Y.S., Krasnogor N., "A study on the design issues of memetic algorithm", (2007).

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

[86] Osman I. H., Kelly J. P., "Meta-Heuristics: Theory and Applications", Kluwer, Boston, MA., (1996).

[87] Perez M., Almeida Rodriguez F., Moreno Vega J.M., “On the use of the path relinking for the p-hub median problem“, Lecture Notes in Computer Science 3004, pp. 155-164, (2004).

[88] Perez-Perez M., Almeida Rodriguez F., Moreno Vega J.M., "A Hybrid GRASP Path Relinking Algorithm for the Capacitated p-Hub Median Problem", Lecture Notes in Computer Science, Vol. 3636, pp. 142-153, (2005).

[89] Pardalos P. M., Resende M. G. C., "Handbook of Applied Optimization", Oxford University Press, New York., (2002) .

[90] Puchinger J., Raidl G., "Cooperating memetic and branch-and-cut algorithms for solving the multidimensional knapsack problem", Proceedings of the 2005 Metaheuristics International Conference, Vienna, Austria, pp. 775–780 (2005).

[91] Puchinger J., Raidl G.R., Koller G., "Solving a real-world glass cutting problem", Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2004. LNCS, vol. 3004, pp. 165–176. Springer, Heidelberg (2004).

[92] Rechenberg I., "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", Frommann-Holzboog Verlag, Stuttgart (1973).

[93] Ribeiro C. C., Hansen P., "Essays and Surveys in Metaheuristics", Kluwer, Norwell, MA, (2002).

[94] Schnecke V., Vornberger O., "Hybrid genetic algorithms for constrained placement problems", IEEE Transactions on Evolutionary Computation, 1(4):266-277, (1997).

[95] Schwefel H.P., "Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of natural evolution", Annals of Operations Research 1, 165–167 (1984).

[96] Skorin-Kapov D., Skorin-Kapov J., O'Kelly M., "Tight linear programming relaxations of uncapacitated p-hub median problems", European Journal of Operational Research 94, 582-593, (1996).

[97] Skorin-Kapov D., Skorin-Kapov J., "On tabu search for the location of interacting hub facilities", European Journal of Operational Research 73, pp. 502-509, (1994).

[98] Soriano P., Gendreau M., "Diversification strategies in tabu search algorithms for the maximum clique problem", Annals of Operations Research 63, 189–207, (1996).

[99] Stanimirović Z., "Genetski algoritmi za rešavanje nekih NP-Teških hab lokacijskih problema", Doktorska Disertacija, Univerzitet u Beogradu, Matematicki fakultet, (2007).

[100] Stanimirović Z., "Solving the capacitated single allocation hub location problem using genetic algorithm", C.H.Skiadas (Ed.), Recent Advances in StochasticModelling and Data Analysis, World Scientific Publishing Co. Pte Ltd., pp. 464-471, (2007).

[101] Stanimirović Z., Kratica J., Filipović V., Tošić D., „Evolutivni pristup za rešavanje hab lokacijskih problema", Zavod za udžbenike i nastavna sredstva, Beograd (2011).

[102] Sudholt D., "Memetic algorithms with variable-depth search to overcome local optima", (2008).

[103] Thamilselvan R., Balasubramanie P., "Integration of Genetic Algorithm with Tabu Search for Job Shop Scheduling with Unordered Subsequence Exchange Crossover", Journal of Computer Science 8 (5), 681-693, (2012).

[104] Topcuoglu H., Corut F., Ermis M., Yilmaz G., "Solving the Uncapacitated Hub Location Problem Using Genetic Algorithms", Computers & Operations Research, Vol. 32, pp. 967-984, (2005).

[105] Tošić D., Mladenović N., Kratica J., Filpović V., "Genetski algoritmi", Matematički institut SANU, Beograd (2004).

[106] Vazirani V. "Approximation Algorithms", Springer, Berlin (2001).

[107] Watson-Gandy C., "Heuristic procedures for the m-partial cover problem on a plane", European Journal of Operational Research (1982).

[108] Whitley D., "Using reproductive evaluation to improve genetic search and heuristic discovery", (1987).

[109] Wu L.-Y., Zhang X.-S., Zhang J.-L., "Capacitated facility location problem with general setup cost", Computers and Operations Research 33, 1226–1241, (2006).

[110] Yaman H., "Star p-hub median problem with modular arc capacities", Computers and Operations Research, 35 (9), 3009–3019. (2008).

[111] Yaman H., Elloumi S.,"Star p-hub center problem and star p-hub median problem with bounded path lengths", Computers and Operations Research, 39 (11), 2725–2732, (2012).

[112] Yaman H., Carello G., "Solving the hub location problem with modular link capacities", Computers & Operations Research 32, 3227 – 3245 (2005).