Univerzitet u Beogradu
Matematički fakultet

Zorica Stanimirović

REŠAVANJE NEKIH DISKRETNIH LOKACIJSKIH PROBLEMA PRIMENOM GENETSKIH ALGORITAMA

Magistarski rad

B e o g r a d

2004.

Mentor: **Prof. dr Đorđe Dugošija**
Matematički fakultet u Beogradu

Članovi komisije: **Prof. dr Boško Jovanović**
Matematički fakultet u Beogradu

Prof. dr Dušan Tošić
Matematički fakultet u Beogradu

dr Jozef Kratica
naučni saradnik
Matematički institut SANU

Datum odbrane: _____

Rešavanje nekih diskretnih lokacijskih problema primenom genetskih algoritama

Rezime

U ovom radu opisan je genetski algoritam (GA) za rešavanje problema p-hab medijane (UMApHMP), p-hab centra sa višestrukim alokacijama (UMApHCP) i diskretno uređenog problema medijane (DOMP) neograničenih kapaciteta. Ovi NP-teški problemi nalaze veliku primenu u praksi. Primenjena je binarna reprezentacija rešenja, genetski operatori razvijeni u skladu sa prirodom problema i hibridizacija GA sa modifikovanom heuristikom zamene za rešavanje DOMP-a. Algoritam je testiran na odgovarajućim instancama iz literature. Za oba hab lokacijska problema GA dostiže optimalne vrednosti na instancama koje su do sada poznate u zadovoljavajućem vremenu računanja. U radu su data rešenja i za instance većih dimenzija (n=200, p=20) za koje optimalna rešenja nisu poznata. Značajni rezultati su dobijeni i pri rešavanju DOMP-a na dimenzijama problema n≤900, p≤200. Dobijeni rezultati na sva tri rešavana problema su uporedivi ili bolji od postojećih metoda.

Abstract

In this work a genetic algorithm (GA) for solving Uncapacitated Multiple Allocaton p-hub Median Problem (UMApHMP), Uncapacitated Multiple Allocaton p-hub Center Problem (UMApHCP) and Discrete Ordered Median Problem (DOMP) is described. These NP-hard problems have many applications in practice. Binary representation is used, genetic operators adopted to the problems are constructed and hybridization GA with modified interchange heuristic for solving DOMP is applied. Proposed algorithm is tested on the corresponding instances from the literature. For both hub location problems GA reaches all solutions that are proved to be optimal so far in a reasonable computational time, even for problem instances of higher dimensions. In this paper the solutions for the large-scaled problem instances (n=200, p=20) that are not reported in the literature yet are also presented. Significant results are also obtained on DOMP instances with dimensions n≤900, p≤200. For all problems GA solutions are comparable or better than ones obtained by existing methods.

1. UVOD

1.1 Lokacijski problemi

Može se reći da je prvi lokacijski problem na koji nailazimo u istoriji matematike sledeći: ako su date tri tačke u ravni, naći četvrtu tačku takvu da je suma njenih rastojanja do preostalih triju tačaka minimalna. Teško je ustanoviti ko ga je prvi formulisao. Prema nekim izvorima to je bio *Pierre de Fermat (1601-1665)*, prema drugim prvi ga je postavio i rešio italijanski matematičar *Battista Cavalieri (1598-1647)*. Nesumnjivo je da su se mnogi matematičari uporedo bavili ovim problemom u gore navedenom ili u nekom drugom, ekvivalentnom obliku.

Ipak, izvesno je da je *Alfred Weber* u svojoj poznatoj knjizi [Web09] prvi ukazao na njegov praktični značaj iskoristivši verziju ovog problema za ilustraciju problema minimizacije troškova transporta u industriji. Dve od tri fiksirane tačke on označava kao resurse sirovina, a treću kao lokaciju potrošača, pri čemu svaka tačka ima izvesnu "težinu". Pitanje je gde izgraditi industrijsko postrojenje tako da cena transporta bude minimalna. Weber je optimalnu lokaciju našao konstruktivnim putem i predložio isti postupak za rešavanje problema dimenzije veće od 3.

Ako su (x_i, y_i), i=1,..,n koordinate n fiksiranih tačaka (lokacije potrošača), w_i, i=1,...,n odgovarajuće težine (cene transporta do datog potrošača po jedinici dužine), Weber-ov problem nalaženja optimalne lokacije skladišta (x^*,y^*) može biti formulisan na sledeći način:

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

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

Naizgled jednostavan, Weber-ov problem je predmet proučavanja do današnjeg dana. Razvijene su brojne i raznovrsne metode i modeli za rešavanje raznih njegovih modifikacija, produženja i uopštavanja. Istorijat i ostali aspekti lokacijskih problema mogu se naći u stranoj literaturi: [Dea85], [Lov88], [Mir90], [Fra92], [Dre02].

U ovom radu pažnja je posvećena diskretnim lokacijskim problemima, koji su u poslednje četiri decenije doživeli pravu ekspanziju, najviše zahvaljujući svojoj širokoj primeni u praksi. Srećemo ih u svim segmentima života, recimo pri određivanju lokacija za izgradnju škola, bolnica, skladišta, industrijskih postrojenja, autobuskih stanica, aerodroma, tržnih centara i sličnih javnih objekata tako da budu zadovoljene potrebe korisnika a troškovi transporta svedeni na minimum. U nekim slučajevima

potrebno je da dodatni kriterijumi budu zadovoljeni, na primer: fiksiran broj snabdevača, minimalna udaljenost na kojoj se oni moraju nalaziti, snabdevanje svakog korisnika sa njemu najbliže lokacije, ograničeni kapaciteti korisnika i/ili snabdevača, fiksirani troškovi itd. Efikasno lokacijsko planiranje ima dugoročni uticaj na dobro funkcionisanje javnih službi i produktivnost kompanija, pa je razumljivo interesovanje za diskretne lokacijske probleme. U poslednjih par decenija lokacijsko planiranje predstavlja jednu od najprofitabilnijih oblasti primenjene matematike. Najpoznatiji diskretni lokacijski problemi su: maksimalno pokrivanje (maximal covering), problemi p-centra (p-center), p-medijane (p-median), p-disperzije (p-dispersion), kao i problemi fiksiranih troškova (fixed charge problem), lokacije habova (hub location problems), maksimalne sume (maxisum problem) itd. Njihova klasifikacija se može izvršiti na različite načine, a detaljnije o tome nalazi se u [Bra89], [Vuj96], [Dre02] i [Mla04].

Diskretni lokacijski problemi su uglavnom vrlo teški za rešavanje. Štaviše, većina spada u NP teške probleme ([Gar79] i [Cre97]). Takođe, ne postoji opšti matematički model koji dobro opisuje sve diskretne lokacijske probleme koji se sreću u praksi. Svaki model ima specifičnu strukturu (funkciju cilja, promenljive, ograničenja) zavisno od problema na koji se odnosi. Iz ovih razloga, egzaktne metode (na primer, branch-and-bound i branch-and-cut), u razumnom vremenu izvršavanja često ne mogu rešiti instance problema većih dimenzija. Zato je do sada razvijen čitav niz najraznovrsnijih heurističkih metoda koje su prilagođene karakteristikama i strukturi specijalnih vrsta problema, ali su zavisne od prirode problema. Neke od njih su proždrljiva heuristika (greedy heuristic) i brojne heuristike zamene (add/remove heuristic, 1-interchange, 2-interchange, fast interchange heuristics).

Poslednjih nekoliko decenija razvijaju se i neki opšti heuristički pristupi-metaheuristike koji su se pokazali vrlo efikasni u praksi. Najčešće korišćene metaheuristike su: tabu pretraživanje (tabu search) [Glo86], [Glo90], simulirano kaljenje (simulated annealing) [Kir83], Lagranžova relaksacija (Lagrangean relaxation) [Geo74], genetski algoritmi (genetic algorithms) [Hll75], [Kra00], [Toš04], metoda promenljivih okolina (variable neighborhood search) [Mla95], [Mla97], neuralne mreže (neural networks) [Fan90], mravlji sistemi (ant systems) [Dor96] i druge. Ove metode uglavnom daju dobra rešenja, neretko se dobijaju optimalna rešenja, iako se u većini slučajeva optimalnost ne može dokazati. Čak i u slučajevima kada problem ima više lokalnih ekstrema, ove heuristike nam pronalaze globalni optimum. Moguće je i njihovo međusobno kombinovanje u cilju korišćenja dobrih strana svake od njih, kao i hibridizacija sa egzaktnim metodama čime se povećava efikasnost pri nalaženju optimalnog rešenja. Većina ovih metoda se može i paralelizovati na višeprocesorskim računarima. Pregled metaheuristika sa primenama dat je u [Glo03].

U ovom radu predložen je genetski algoritam kao metaheuristika za efikasno rešavanje tri diskretna lokacijska problema: problem p-hab medijane i p-hab centra neograničenih kapaciteta sa višestrukim alokacijama (uncapacitated multiple allocation p-hub median/center problem), kao i diskretno uređen problem medijane (Discrete Ordered Median Problem - DOMP).

1.2 Genetski algoritmi

1.2.1 Opšte karakteristike GA

Pojam genetskog algoritma (GA) prvi je uveo Holland u svojoj knjizi iz 1975. godine [Hll75] koja se bavi teorijom adaptivnih sistema. Osnovna ideja je simuliranje procesa prirodne evolucije jedne populacije jedinki pod dejstvom genetskih operatora. Iako su i ranije postojali radovi sa sličnim idejama, Holland se smatra tvorcem ove metaheuristike i postavke iz njegovih najranijih radova još uvek važe. Danas se genetski algoritmi koriste za rešavanje široke klase problema kombinatorne optimizacije. Do sada je publikovan veliki broj radova koji se bave teorijskim razmatranjima genetskih algoritama. Neki od njih su: [DJo75], [Gol89], [BeD93a], [BeD93b], [Yur94], [Mic96], [Mit96] i [Müh97]. Koncept genetskih algoritama je izložen i u domaćoj literaturi: [Čan96], [Fil98], [Kra00] i [Toš04].

Genetski algoritam se primenjuje na konačnom skupu jedinki koji se naziva populacija. Svaka jedinka u populaciji je predstavljena nizom karaktera (genetskim kodom) i odgovara nekom rešenju u pretraživačkom prostoru. Kodiranje može biti binarno ili nad nekom drugom azbukom veće kardinalnosti. Kodiranje rešenja je važan korak GA jer neadekvatan izbor koda može dovesti do loših rezultata bez obzira na ostalu strukturu algoritma.

Početna populacija se obično generiše na slučajan način, što obezbeđuje raznovrsnost genetskog materijala. Moguće je korišćenje neke heuristike za generisanje početne populacije, ili jednog njenog dela, uz uslov da se ta heuristika relativno brzo izvršava i da značajno ne smanjuje raznovrsnost genetskog materijala. Svakoj jedinki u populaciji (u praksi ih je najviše do nekoliko stotina) se na određen način dodeljuje funkcija prilagođenosti koja je merilo kvaliteta jedinke (odnosno odgovarajućeg rešenja). Standardni pristup konceptu GA ([Hll75], [Gol89], [Toš04]) smatra da je cilj algoritma da se iz generacije u generaciju poboljšava prilagođenost svake jedinke u populaciji, kao i srednja prilagođenost cele populacije uzastopnom primenom genetskih operatora: selekcije, ukrštanja i mutacije. U radu [Hut02] izložena je hipoteza da srednja prilagođenost populacije generalno smanjuje raznovrsnost genetskog materijala, pa se predlaže potpuno suprotna strategija "uniformne selekcije".

Genetski operator selekcije vrši izbor jedinki iz populacije koje učestvuju u stvaranju nove generacije. Selekcija se primenjuje u skladu sa vrednostima funkcije prilagođenosti. Standardni pristup smatra da će bolje prilagođene jedinke preneti dobar genetski materijal na svoje potomke, pa smanjuje šansu prolaska loših jedinki u narednu generaciju, tako da one postepeno nestaju iz populacije.

Operator ukrštanja predstavlja postupak razmene delova genetskog koda dve (ili više) jedinke-roditelja i tako dobijaju kodovi novih (jedne ili više) jedinki-potomaka. Razmena genetskog materijala daje mogućnost da dobro prilagođene jedinke generišu još bolje potomke, ali i da neki dobri geni relativno lošijih jedinki dobiju svoju šansu za dalju reprodukciju.

Mutacijom se vrši promena koda jedinke zamenom pojedinih simbola koda nekim drugim simbolima azbuke kodiranja. Operator mutacije se koristi da bi se unela raznovrsnost među jedinkama populacije jer one vremenom mogu postati jako slične. Mutacija takođe sprečava gubitak dela genetskog materijala do kojeg može doći

višestrukom primenom operatora selekcije i ukrštanja. Svaki gen genetskog koda može mutirati sa datom malom verovatnoćom p_{mut}.

Operatori genetskog algoritma se uzastopno primenjuju do zadovoljenja nekog od kriterijuma zaustavljanja: maksimalan broj generacija, dostignut optimum, nepromenjen kvalitet rešenja posle unapred zadatog broja generacija itd.

Na slici 1.1 je dat shematski zapis osnovnih elemenata GA:

```
Unošenje_Ulaznih_Podataka();
Generisanje_Početne_Populacije();
while (! Kriterijum_Zaustavljanja_GA() )
  {
     for(i=0;i< Npop; i++) pi = Vrednosna_Funkcija();
     Funkcija_Prilagođenosti();
     Selekcija();
     Ukrštanje();
     Mutacija();
  }
Štampanje_Izlaznih_Podataka();
```

Slika 1.1 Shematski zapis GA

1.2.2 Prost GA

Najjednostavniji koncept genetskog algoritma – prost GA (simple genetic algorithm, SGA) sastoji se od proste rulet selekcije, jednopozicionog ukrštanja i proste mutacije. Ova varijanta GA može se naći u Hollandovoj knjizi [Hll75], a osobine navedenih genetskih operatora date su u narednom poglavlju.

Jedan od nedostataka SGA je mogućnost preuranjene konvergencije. Iz generacije u generaciju može doći do sve veće sličnosti između jedinki. Takva populacija može, usled preovlađivanja visoko prilagođenih jedinki u populaciji, odvesti algoritam u lokalni ekstrem, pri čemu su šanse za poboljšanje dobijenog rešenja do globalnog optimuma jako male. Preuranjena konvergencija je najčešće posledica primene proste rulet selekcije. Visoko prilagođene jedinke će vremenom istisnuti lošije jedinke iz populacije, iako one možda sadrže dobre i raznovrsne gene koji mogu dati kvalitetnije potomke. Na taj način dolazi do problema gubitka genetskog materijala koji ni mutacija ne može rešiti u praksi.

Drugi nedostatak SGA je spora konvergencija, koja se obično javlja u završnoj fazi GA. Dešava se da je srednja prilagođenost jedinki u populaciji velika, a razlika između najbolje jedinke i ostalih mala, tako da je gradijent funkcije prilagođenosti nedovoljan da bi genetski algoritam dostigao optimalno rešenje u doglednom vremenu.

1.2.3 Složeniji koncept GA

Koncept SGA se, zbog gore navedenih nedostataka, može uspešno primeniti samo na manji broj jednostavnijih problema kombinatorne optimizacije. Složeniji problemi zahtevaju korišćenje poboljšanih verzija GA. Razvijeniji koncepti GA uključuju različite vrste kodiranja i funkcija prilagođenosti, a razvijen je i čitav spektar genetskih operatora selekcije, ukrštanja i mutacije koji se koriste u zavisnosti od karakteristika rešavanog problema. Iako postoje brojne i raznovrsne implementacije genetskih operatora, njihov razvoj i usavršavanje su i dalje aktuelni.

1.2.3.1 Kodiranje i funkcija prilagođenosti

Način kodiranja i definisanje funkcije prilagođenosti su važni činioci efikasnosti genetskog algoritma. Oni su u tesnoj vezi sa prirodom problema koji rešavamo. Za uspešnu primenu genetskog algoritma uglavnom je najpogodnije binarno kodiranje, koje svakom rešenju dodeljuje kod unapred zadate dužine nad binarnom azbukom {0,1}. Moguće je i kodiranje nad azbukama veće kardinalnosti, recimo celim ili realnim brojevima. Neki matematičari u teorijskim razmatranjima favorizuju binarno kodiranje [Gol89], dok drugi daju prednost ne-binarnim reprezentacijama [Ant89], [BeD93b]. Eksperimentalna poređenja binarnog i drugih načina kodiranja mogu se naći u [Jan91].

U literaturi nailazimo na različite načine računanja funkcije prilagođenosti: *direktno preuzimanje, linearno skaliranje, skaliranje u jedinični interval, sigma odsecanje* (videti [Kra00]). Specifičnosti problema upućuju na izbor odgovarajućeg načina, a moguće je i njihovo kombinovanje.

Iako je GA najuspešniji ako je funkcija prilagođenosti neprekidna i glatka, prednosti GA dolaze do izražaja tek kada to nije ispunjeno, posebno u slučajevima kada klasične metode rešavanja nije moguće primeniti.

Za GA je najpogodnije uspostaviti bijektivnu vezu između rešenja problema i njihovih genetskih kodova. Ako to nije moguće, kodiranje ne mora biti bijektivno, čak ne mora biti preslikavanje. Međutim, tada postoji mogućnost da se tokom rada GA generišu kodovi koji ne odgovaraju nijednom rešenju, tzv. nekorektne jedinke. Nekorektne jedinke mogu se izbaciti iz populacije postavljajući im vrednost funkcije prilagođenosti na nulu, popraviti do korektnih primenom neke metode ili smanjiti njihovu prilagođenost za određenu vrednost pomoću kaznenih funkcija. Jedan od uspešnih metoda u praksi sastoji se u tome da se početna populacija generiše korektno, a zatim primenjuju genetski operatori koji čuvaju korektnost. Detaljnije o načinima rešavanja problema nekorektnih jedinki nalazi se u [Lvi93a], [SmA93] i [Orv93].

1.2.3.2 Selekcija

Pošto je selekcija u direktnoj vezi sa funkcijom prilagođenosti, osnovni način implementacije ovog genetskog operatora je prosta rulet selekcija. Ovaj metod koristi distribuciju u kojoj je verovatnoća selekcije jedinke proporcionalna njenoj prilagođenosti. Sa tim šansama jedinke učestvuju na ruletu i u skladu sa njima (ne) prolaze u proces stvaranja nove generacije. Nedostatak proste rulet selekcije je mogućnost preuranjene konvergencije usled postepenog preovlađivanja visoko prilagođenih jedinki u populaciji koje ne odgovaraju globalnom optimumu.

Da bi se izbegao ovaj problem može se koristiti *selekcija zasnovana na rangiranju* genetskih kodova prema njihovoj prilagođenosti. Funkcija prilagođenosti jedinke jednaka je nekom rangu iz unapred zadatog niza rangova, pa zavisi samo od pozicije jedinke u populaciji. Može se koristiti linearno, ali i drugi vidovi rangiranja.

Još jedan oblik selekcije je *turnirska selekcija*, kod koje se na slučajan način generišu podskupovi od po N jedinki (N je unapred zadat broj), a zatim se u svakom podskupu, po principu turnira, bira najbolja jedinka koja učestvuje u stvaranju nove generacije. Obično je problem izabrati N tako da se smanje nepovoljni stohastički efekti, kako bi što bolji i raznovrsniji genetski materijall prošao u sledeću generaciju. U slučajevima kada je pogodno da veličina turnira ne bude ceo broj, uspešno se pokazala *fino gradirana turnirska selekcija* (*FGTS*). Detaljni opis ove i ostalih tipova selekcije i njihovi teorijski aspekti mogu se naći u [Fil98]. Primena fino gradirane turnirske selekcije i poređenje u praksi sa drugim operatorima selekcije dati su u [Fil00], [Fil01], [Fil03].

1.2.3.3 Ukrštanje

Razmena genetskog materijala jedinki-roditelja može se sprovesti pomoću jednopozicionog, dvopozicionog, višepozicionog ili uniformnog ukrštanja, ali postoje i drugi, složeniji vidovi ovog genetskog operatora [Müh97].

U SGA implementirano je *jednopoziciono ukrštanje*, kod kojeg se na slučajan način biraju parovi jedinki-roditelja iz populacije i broj $k \in \{0,..., n-1\}$ koji predstavlja tačku ukrštanja (n je dužina genetskog koda). Svi geni od počevši od pozicije k+1 do poslednje pozicije n–1 u genetskim kodovima jedinki-roditelja uzajamno menjaju mesta, stvarajući pritom dve nove jedinke-potomka. Kod *dvopozicionog ukrštanja* slučajno se biraju dve tačke ukrštanja k_1 i k_2 i razmenjuju delovi genetskih kodova roditelja, od gena na poziciji k_1+1 do gena na poziciji k_2.

Operator *uniformnog ukrštanja* za svaki roditeljski par generiše binarni niz iste dužine kao genetski kod jedinki, tzv. "masku". Jedinke-roditelji razmenjuju gene na svim pozicijama na kojima maska ima vrednost 0, dok na mestima gde maska uzima vrednost 1 roditelji zadržavaju svoje gene. Karakteristike uniformnog ukrštanja mogu se naći u [Spe91].

Osobine problema koji rešavamo utiču na izbor operatora ukrštanja. Za razliku od kodiranja i selekcije kod kojih korišćenje različitih pristupa može dovesti do drastičnih razlika u performansama GA, kod ukrštanja su razlike mnogo manje, ali su ipak primetne. U slučajevima kada je pogodno delimično sačuvati strukturu genetskog koda, zbog velike međuzavisnosti gena, korisno je jednopoziciono ukrštanje. Ako želimo da u većoj meri razbijemo i izmešamo blokove u genetskom kodu, možemo

izabrati dvopoziciono ili višepoziciono ukrštanje. Ukoliko su geni skoro, ili potpuno nezavisni, najbolje rezultate daje uniformno ukrštanje.

1.2.3.4 Mutacija

Izbor operatora mutacije može da značajno utiče na performanse GA. Ako GA koristi binarno kodiranje i populacija nema nekorektnih jedinki, obično se implementira operator *proste mutacije* koji prolazi kroz gene genetskog koda jedinke i za svaki od njih proverava da li je mutiran ili ne. Verovatnoća mutacije gena p_{mut} je unapred zadata mala veličina, najčešće uzeta iz intervala [0.001, 0.01]. Prosta mutacija se ponekad primenjuje preko binarnog niza – maske koja se slučajno generiše za svaku jedinku i nosi informaciju o tome na kojoj poziciji u genetskom kodu dolazi do promene gena.

Korišćenjem binomne ili normalne raspodele može se ubrzati realizacija operatora proste mutacije. *Mutacija pomoću binomne raspodele* koristi činjenicu da slučajna promenljiva X_{ind}=broj mutiranih gena jedinke ima binomnu raspodelu $B(n, p_{mut})$, gde je n dužina genetskog koda, a p_{mut} nivo mutacije. Neka je F(k) njena funkcija raspodele. Pomoću F^{-1} nalazimo n_{mut}= broj gena koje treba mutirati u datom genetskom kodu. Pozicije gena koji se mutiraju biraju se uniformno iz skupa {0,...,n-1}. U slučaju dugačkog genetskog koda može doći do greške izračunavanja broja n_{mut}, pa je tada pogodnije koristiti *mutaciju pomoću normalne raspodele*. Ako $n \to \infty$ slučajna promenljiva X_{ind} se može aproksimirati normalnom raspodelom $N(n*p_{mut}; n*p_{mut}*(1-p_{mut}))$, uz uslov da je n dovoljno veliko a p_{mut} dovoljno malo. Kao i u prethodnom slučaju, koristimo inverznu funkciju funkcije normalne raspodele da bismo izračunali broj gena koje treba mutirati, a i njihove pozicije određujemo na isti način. Razvijena je varijanta ovog operatora koji se primenjuje na celu populaciju, jer se pri mutaciji genetski kodovi svih jedinki mogu posmatrati kao jedna celina. Detaljnije informacije o gore navedenim konceptima mutacije mogu se pročitati u [Kra00] i [Toš04].

Ukoliko mutacija nije uniformna, što se dešava kada geni genetskog koda nisu ravnopravni, već je neke delove koda jedinke potrebno mutirati sa manjom ili većom verovatnoćom, obično se koristi *normalna mutacija* (primenjuje se u skladu sa normalnom raspodelom), *eksponencijalna mutacija* (broj mutiranih gena u kodu eksponencijalno opada) i sl.

Ako GA koristi kodiranje celim ili realnim brojevima (sa pokretnim zarezom) razvijeni su drugi koncepti mutacije: zamena gena slučajno izabranim brojem (*random replacement*), dodavanje ili oduzimanje male vrednosti (*creep*), množenje sa brojem bliskim jedinici (*geometric creep*) itd. Za oba creep operatora mutacije potrebne vrednosti su slučajne i mogu imati uniformnu, eksponencijalnu, Gausovu ili binomnu raspodelu (videti [BeD93a] i [BeD93b]).

U literaturi često nalazimo mutaciju i ukrštanje zavisne od prirode problema, što se obično javlja u slučajevima kada su nekorektne jedinke prisutne u velikom procentu u populaciji. To može biti posledica specifičnog načina kodiranja ili postojanja velikog broja ograničenja (constrained problems) koja se ne mogu sva implementirati u genetski kod. Ovakvi operatori mutacije i ukrštanja čuvaju korektnost jedinki na koje se primenjuju. Pri rešavanju lokacijskih problema nailazimo na neke od njih: *hipermutacija* (*Hypermutation*) u [Cor01], *N4H mutacija* u [Alk04*], Modified-Basic Operator (MBO), String-of-Change Operator (SOC), Template Operator (TEMP), Backward Crossover Operator (BACK)* u [Boz02] i drugi.

1.2.3.5 Kriterijum zaustavljanja

Genetski algoritmi su u osnovi stohastičke metode pretrage dopustivog prostora rešenja, tako da mogu raditi beskonačno dugo, ukoliko im ne nametnemo kriterijum zaustavljanja. Postoji nekoliko kriterijuma završetka GA: maksimalni broj generacija, sličnost jedinki u populaciji, najbolja jedinka je ponovljena maksimalni broj puta, algoritam je dostigao optimalno rešenje (ukoliko je ono unapred poznato), dokazana optimalnost najbolje jedinke (ukoliko je to moguće), ograničeno vreme izvršavanja GA, prekid od strane korisnika itd. Svaki od navedenih kriterijuma ima dobre i loše aspekte, tako da se u praksi najbolje pokazalo njihovo kombinovanje, jer se tako smanjuje mogućnost loše procene prekida GA.

1.2.3.6 Ostali aspekti GA

Za uspešnu primenu GA važno je dobro odrediti *politiku zamene generacija*:

- *generacijski GA* u svakoj generaciji vrši promenu svih jedinki u populaciji,
- *stacionarni GA* u svakoj generaciji generiše samo deo populacije, dok se preostale jedinke prenose iz prethodne populacije,
- *elitistički GA* u svaku generaciju propušta određen broj elitnih jedinki, bez primene genetskih operatora i računanja njihove funkcije prilagođenosti, čime se skraćuje vreme izvršavanja algoritma i obezbeđuje čuvanje dobrih rešenja. Pored uštede procesorskog vremena, i kvalitet rešenja se može bolje predvideti, tako da elitistička strategija ima prednost nad ostalim, mada je moguće i njihovo uspešno kombinovanje.

Implementacija GA podrazumeva korišćenje raznih parametara: veličina početne populacije, nivo mutacije, nivo ukrštanja itd. Nedostatak genetskog algoritma je u tome što ne postoji jedinstvena kombinacija parametara koja je najbolja za sve probleme, ili bar za različite instance istog problema, već ih u svakom konkretnom slučaju moramo podesiti eksperimentalnim putem.

Parametre možemo menjati i u toku izvršavanja GA:

- *fiksna promena parametara* unapred zadaje smer promene (povećavanje ili smanjivanje) linearno ili eksponencijalno.
- *adaptivna promena parametara* (ne)menja parametar u zavisnosti od toga koliko je dati operator do tada bio uspešan i kakve je rezultate dao.

Detaljnije o podešavanju parametara GA može se naći u [BrmL91], [Bëc92a], [Bëc93] i [Sri94].

Genetski algoritmi se mogu kombinovati sa drugim heuristikama. Heuristike (jedna ili više) se mogu koristiti ne samo za poboljšavanje početne populacije, već i svake naredne generacije. Možemo ih primenjivati na celu populaciju, jedan njen deo ili na pojedinačnu (često najbolju) jedinku. Značajan doprinos poboljšanju performansi GA predstavlja paralelizacija, kao i keširanje – videti u [Kra00].

1.3 Primena GA u rešavanju lokacijskih problema

Jedna od važnih primena genetskih algoritama je rešavanje NP-kompletnih i ostalih diskretnih lokacijskih problema. Pregled svih radova koji se bave praktičnim aspektima GA u ovoj oblasti prevazilazi obim ovog rada, pa ćemo pomenuti samo nekoliko primena GA na neke od poznatijih NP-kompletnih lokacijskih problema.

U radovima [DD93], [Dre03] i [Cor01] su predloženi različiti koncepti GA, kao i rezultati njihove primene na instancama problema p-medijane. Korišćene su razne hibridizacije genetskog algoritma sa nekim heuristikama za poboljšavanje rešenja i razvijeni genetski operatori koji zavise od prirode p-medijan problema: [Alp03], [Boz02], [Dv99], [Alk04].

[Mih03] rešava problem p-centra, a [Lor00] klastering probleme pomoću GA. Neki od hab lokacijskih problema su takođe rešavani primenom genetskog algoritma: [AbH98], [AbH99], [Ern04a]. I u domaćoj literaturi nalazimo uspešne primene unapređenih verzija GA na prost lokacijski problem, problem dizajniranja mreže, selekcije indeksa i bi-povezanosti grafova: [Kra98], [Kra00], [Lju00], [Lju00a], [Kra01], [Kra02] i [Kra03]. Hibridizacija GA sa egzaktnom metodom Branch-and-Cut-and-Price je data u [Lju04].

Iako je teorijski moguće primeniti koncept GA na svaki lokacijski problem, GA nije podjednako uspešan za sve probleme. Kod problema sa mnogo ograničenja mogu nastati teškoće u pronalaženju adekvatnog načina kodiranja. Ispostavilo se da je genetski algoritam najefikasniji za rešavanje problema sa malo ograničenja i relativno složenom funkcijom cilja [Ree93].

2. PROBLEM P-HAB MEDIJANE NEOGRANIČENIH KAPACITETA SA VIŠESTRUKIM ALOKACIJAMA

Mreže habova (hub networks) imaju široku primenu u modernim transportnim i telekomunikacijskim sistemima. Umesto direktnog snabdevanja svakog korisnika od njemu pridruženog snabdevača, u hab mrežama je dozvoljen transport snabdevač-korisnik preko izabranog skupa habova. Habovi predstavljaju centre konsolidacije i kolekcije protoka u mreži između dve lokacije. Svaki čvor je pridružen jednom ili više habova, tako da se protok od jednog do drugog čvora realizuje (ali ne uvek obavezno) preko skupa odgovarajućih habova, koji može biti i jednočlan. Koristeći habove kao tačke preusmeravanja protoka i povećavajući transport između habova, kapacitet mreže se može iskoristiti dosta efikasnije. S obzirom da je cena transporta između habova po jedinici količine niža, ukupni troškovi transporta u mreži se na ovaj način smanjuju.

Hab lokacijski problemi se intenzivno koriste za dizajniranje transportnih mreža: železničkih i drumskih sistema, poštanskih mreža, sistema brze isporuke, prevoženja putnika i robe u avio saobraćaju itd. Protok između čvorova u mreži se obično definiše kao broj putnika ili količina robe koju treba transportovati od snabdevača (početni čvor) do korisnika (krajnji čvor). Habovi u ovim sistemima su recimo transportni terminali, centri selekcije, sakupljanja ili preusmeravanja robe i putnika i sl.

Telekomunikacijski sistemi se takođe mogu konfigurisati kao mreže habova. Informacije u kompjuterskim, telefonskim, satelitskim i drugim komunikacijskim sistemima se transmituju preko odgovarajućih linkova (telefonskim ili optičkim kablovima, satelitskim kanalima), što je često vrlo skupo. U cilju smanjivanja cene protoka podataka u ovim mrežama takođe se koriste hab-čvorovi (prekidači, koncentratori, portovi).

U literaturi nalazimo razne varijante hab lokacijskih problema sa različitim ograničenjima i funkcijama cilja. Među njima se mogu uočiti dva osnovna alokacijska koncepta:

- *šema jednostruke alokacije (single allocaton scheme)*, kod koje je svaki snabdevač/korisnik pridružen tačno jednom habu, tako da se sav transport od/do tog čvora obavlja isključivo preko određenog haba.
- *šema višestruke alokacije (multiple allocaton scheme)*, koja dopušta svakom ne-hab čvoru da komunicira sa jednim ili više habova.

Neki od hab lokacijskih problema uključuju razna ograničenja kapaciteta u mreži (*capacitated hub problems*), kao što su: limitirana količina robe koja dolazi ili prolazi kroz hab čvor, ograničen protok između habova kao i između habova i pridruženih korisnika/snabdevača, ograničeni kapaciteti ne-hab čvorova i sl. Čvorovi u mreži habova mogu imati fiksirane troškove *(fixed costs hub problems)* ili se zahteva da broj hab lokacija koje treba uspostaviti bude tačno *p* (*p-hub location problems*). Pregled hab lokacijskih problema i njihova klasifikacija dati su u [Cam96] i [Cam02].

2.1 Matematička formulacija

U literaturi postoje različite formulacije problema p-hab medijane neograničenih kapaciteta sa višestrukim alokacijama (Uncapacitated Multiple Allocation p-hub Median Problem-UMApHMP). U ovom pogavlju koristimo mešovitu celobrojnu linearnu formulaciju datu u [Bol04].

Neka je *I={1,...,n}* skup različitih čvorova mreže, pri čemu svaki čvor označava lokaciju korisnika/snabdevača ili potencijalnu lokaciju haba. Rastojanje od *i*-tog do *j*-tog čvora je C_{ij}, i može se pretpostaviti da važi nejednakost trougla [Cam02]. Količina robe koju treba transportovati od i-tog snabdevača do j-tog korisnika je označena sa W_{ij}. Broj habova koje treba locirati je fiksiran na p. Promenljive H_j, Z_{ik}, Y_{kl}^i i X_{lj}^i su korišćene u formulaciji problema na sledeći način:

- H_j = 1ako je hub lociran na j-tom čvoru, 0 ako nije
- Z_{ik} = količina robe koja polazi od i-tog čvora a sakuplja se u habu k
- Y_{kl}^i = količina robe koja polazi od i-tog čvora, sakuplja se u habu k i distribuira preko haba l
- X_{lj}^i = količina robe koja kreće od čvora i, čije je odredište čvor j, a transportuje se preko haba l.

Protok od čvora-snabdevača do čvora-korisnika sastoji se od tri komponente: transfer od snabdevača do prvog haba, transport između habova i distribucija od poslednjeg haba do korisnika, pri čemu se podrazumeva šema višestruke alokacije. Parametri χ i δ redom označavaju troškove (cenu) kolekcije i distribucije robe po jedinici količine, dok $1-\alpha$ predstavlja koeficijent uštede za transport između habova. Koristeći gornju notaciju, UMApHMP se matematički može zapisati kao:

$$\min \sum_i [\chi \sum_k C_{ik} Z_{ik} + \alpha \sum_k \sum_l C_{kl} Y_{kl}^i + \delta \sum_l \sum_j C_{lj} X_{lj}^i] \tag{2.1}$$

uz uslove

$$\sum_j H_j = p \tag{2.2}$$

$$\sum_k Z_{ik} = \sum_j W_{ij} \text{ , za svako i} \tag{2.3}$$

$$\sum_l X_{lj}^i = W_{ij} \text{ , za svako i, j} \tag{2.4}$$

$$\sum_l Y_{kl}^i + \sum_j X_{kj}^i - \sum_l Y_{lk}^i - Z_{ik} = 0 \text{ , za svako i, k} \tag{2.5}$$

$$Z_{ik} \le \sum_j W_{ij} H_k \text{ , za svako i, k} \quad (2.6)$$

$$\sum_i X_{lj}^i \le \sum_i W_{ij} H_l \text{ , za svako l, j} \quad (2.7)$$

$$X_{lj}^i, Y_{lk}^i, Z_{ik} \ge 0 , \quad H_k \in \{0,1\} \text{ , za svako i, j, k, l} \quad (2.8)$$

Funkcija cilja (2.1) minimizuje sumu transportnih troškova snabdevač-hab, hab-hab i hab-korisnik, pomnoženih sa koeficijentima χ, α i δ respektivno. Uslov (2.2) fiksira broj uspostavljenih habova na p, dok (2.3)-(2.5) predstavljaju jednačine divergencije protoka u mreži za svaki čvor i. Ograničenja (2.6) i (2.7) ne dopuštaju direktnu komunikaciju između ne-hab čvorova, a (2.8) označava ne-negativnu i/ili binarnu reprezentaciju promenljivih H_j, Z_{ik}, Y_{kl}^i i X_{lj}^i.

UMApHMP je NP-kompletan problem, sa izuzetkom specijalnih slučajeva koji se mogu rešiti u polinomijalnom vremenu (na primer, kada je matrica protoka W_{ij} retka). Ako je skup habova unapred izabran, problem je takođe polinomijalno rešiv koristeći algoritam najkraćeg puta (shortest-path algorithm) u $O(n^2p)$ vremenu izvršavanja.

2.2 Postojeći načini rešavanja

U poslednje vreme u literaturi nalazimo dosta radova koji se bave hab lokacijskim problemima, ali se većina odnosi na hab probleme sa šemom jednostruke alokacije. Najčešći pristup hab lokacijskim problemima je celobrojno programiranje, pri čemu se koriste dva osnovna tipa formulacije ovih problema. Prvi koristi $O(n^4)$ promenljivih i zahteva $O(n^3)$ ograničenja [Ham04]. Pošto dimenzije matrice protoka rapidno rastu sa dimenzijom problema, teško je rešiti LP relaksaciju za veći broj čvorova, pa ovaj tip formulacija nije pogodan za branch-and-cut i branch-and-price algoritme. Ali, zbog dobrih ocena donje i gornje granice rešenja, ove fomulacije mogu se koristiti za generisanje početnog rešenja za dual ascent algoritam ili metodu Lagranžove relaksacije. Drugi tip formulacija [Ern98] redukuje broj promenljivih i ograničenja na O(n) i granice rešenja su slabije, ali LP relaksacija se može rešiti znatno brže, iako je veličina branch-and-bound drveta dosta veća.

Campbell u [Cam96] daje formulaciju hab lokacijskih problema sa višestrukom alokacijom koja je slična prvoj gore pomenutoj formulaciji. On koristi slabiji oblik uslova (4) i predlaže jednu heuristiku zamene (exchange heuristic) za UMApHMP. Aykin u [Ayk95] predstavlja metod prebrojavanja (enumeration method) i "proždrljivi" metod zamene (greedy interchange method) za UMApHMP i neke njegove varijante.

U [Soh98] predloženi su formulacija UMApHMp i egzaktni metod za njegovo rešavanje. Autori relaksiraju binarna ograničenja postavljena na promenljivim H_j i rešavaju dobijenu LP relaksaciju, koja daje dosta dobre granice rešenja problema i u većini slučajeva vodi celobrojnom rešenju. Ako dobijeno rešenje nije celobrojno, pomoću drveta implicitnog prebrojavanja (sa samo nekoliko čvorova) lako se stiže do optimuma.

Ernst i Krishnamoorthy [Ern98] koriste metod najkraćeg puta (shortest path method) za prebrojavanje svih mogućih lokacija habova u UMApHMP. Ovaj algoritam je eksponencijalan po p i polinomijalan po n, tako da na instancama problema sa relativno malim brojem habova koje treba uspostaviti ($p \le 5$) daje tačna rešenja u razumnom vremenu računanja. Na instancama UMApHMP većih dimenzija, metod najkraćeg puta se može uspešno kombinovati sa branch-and-bound algoritmom.

Skup čvorova mreže se deli u klastere, a zatim se računaju sve moguće alokacije tačno p habova u svim klasterima. Metod najkraćeg puta obezbeđuje dobijanje donjih granica koje se dalje koriste u branch-and-bound šemi za dobijanje egzaktnog rešenja.

U literaturi postoje i brojne heuristike za rešavanje UMApHMP. Skorin-Kapov i saradnici [Sko94] razvili su dvofazni tabu search algoritam (TS). Posle izbora skupa od p habova, u prvoj fazi korisnici/snabdevači se pridružuju njima najbližim habovima, čime se dobija početno rešenje za tabu search algoritam. TS ispituje različite mogućnosti pridruživanja svakog ne-hab čvora nekoj od hab-lokacija iz prethodno izabranog skupa. Kasniji rad Skorin-Kapova [Sko96] dokazuje da predloženi TS algoritam nalazi optimalno rešenje za sve testirane instance ovog problema.

Boland i saradnici u [Bol04] kombinuju heuristički pristup i egzaktan branch-and-bound metod za hab probleme sa višestrukom alokacijom koristeći pre-procesiranje i algoritme "sečenja" (cutting algorithms). Oni heuristikom dobijaju dobre gornje granice problema koje se dalje koriste za "sečenje" branch-and-bound drveta, čime se smanjuju njegove dimenzije, a time i procesorsko vreme izračunavanja.

Neki rezultati i poređenja postojećih metoda za UMApHMP sa genetskim algoritmom predloženim u ovom radu dati su u 2.4.

2.3 Predloženi GA

2.3.1 Reprezentacija i funkcija prilagođenosti

U ovoj GA implementaciji primenjeno je binarno kodiranje jedinki. Svako rešenje je predstavljeno binarnim stringom dužine n. Jedinica u genetskom kodu označava da je uspostavljen hab na datoj lokaciji, dok nula pokazuje da nije. Pošto snabdevači/korisnici mogu biti pridruženi isključivo uspostavljenim hab-lokacijama, iz genetskog koda se dobija samo niz H_j, dok se vrednosti promenljivih Z_{ik}, Y_{kl}^{i} i X_{lj}^{i} računaju tokom evaluacije funkcije cilja, koja je u ovoj implementaciji identički jednaka funkciji prilagođenosti.

Za fiksiran skup habova (H_j) koristi se modifikacija poznatog Floyd-Warshall algoritma najkraćeg puta, opisanog u [Ah93] i [Ern98]. Posle nalaženja najkraćih puteva za svaki par čvorova u mreži lako se računa funkcija cilja jednostavnim sumiranjem najkraćih rastojanja snabdevač-hab, hab-hab i hab-korisnik pomnoženih odgovarajućim parametrima χ, α i δ.

2.3.2 Genetski operatori

2.3.2.1 Selekcija

Predloženi koncept GA koristi fino-gradiranu turnirsku selekciju (FGTS), opisanu u [Fil98], koja predstavlja unapređenje standardnog operatora turnirske selekcije. Umesto celobrojnog parametra N_{tour}= veličina turnirske grupe, FGTS zavisi od parametra F_{tour}= željena srednja veličina turnira koji uzima realne vrednosti. FGTS operator realizuje dva tipa turnira: prvi tip se sprovodi k_1 puta i njegova veličina je $[F_{tour}]+1$, dok se drugi tip realizuje k_2 puta sa $[F_{tour}]$ jedinki-učesnika turnira. Pošto je

vrednost F_{tour}=5.4 korišćena u ovoj GA implementaciji, odgovarajuće vrednosti k_1 i k_2 (za 50 ne-elitnih jedinki) su 20 i 30 respektivno.

Vreme izvršavanja FGTS operatora je $O(n*F_{tour})$. U praksi, parametar F_{tour} se smatra konstantnim (ne zavisi od n), što daje O(n) vremensku složenost. Detaljnije informacije o FGTS mogu se naći u [Fil98] i [Fil03].

2.3.2.2 Ukrštanje

Posle selekcije parova jedinki-roditelja operator ukrštanja se primenjuje na njih stvarajući dve jedinke-potomka. Osnovni tip ukrštanja razmenjuje delove genetskih kodova roditelja počevši od slučajno izabrane tačke ukrštanja. Prosta zamena segmenata genetskog koda može proizvesti nekorektne potomke za UMApHMP (broj jedinica u kodu može postati različit od p), iako su roditelji imali tačno p jedinica u svojim genetskim kodovima. Da bi se ovaj problem prevazišao, osnovni operator ukrštanja je modifikovan u ovoj GA implementaciji. Operator istovremeno prolazi kroz genetske kodove obe jedinke-roditelja s desna na levo tražeći poziciju *i* na kojoj prvi roditelj ima 1 a drugi 0. Jedinke razmenjuju gene na nađenoj poziciji *i* (koja predstavlja tačku ukrštanja), a zatim se sličan proces sprovodi počevši od leve strane genetskog koda na desno. Operator sada traži poziciju *j* na kojoj stoji 0 u kodu prvog roditelja, a 1 u kodu drugog. Posle zamene gena na *j*-toj poziciji broj uspostavljenih habova ostaje nepromenjen. Opisani proces (korak) se ponavlja do ispunjenja uslova $j \geq i$.

Ukrštanje se realizuje sa verovatnoćom p_{cross}=0.85, što znači da oko 85% parova jedinki učestvuju u stvaranju jedinki-potomaka.

2.3.2.3 Mutacija

Jedinke-potomci generisane operatorom ukrštanja podležu mutaciji sa zaleđenim bitovima. Operator mutacije se realizuje promenom slučajno izabranog gena u genetskom kodu jedinke (0 u 1, 1 u 0), sa osnovnim nivoom mutacije od 0.4/n za nezaleđene i 1.0/n za zaleđene bitove. Oba nivoa mutacije su konstantna tokom svih generacija genetskog algoritma.

U svakoj mutiranoj jedinki operator mutacije prebrojava i upoređuje brojeve promenjenih nula i jedinica genetskog koda. U slučaju da se ti brojevi razlikuju, neophodno je dodatno mutirati odgovarajuće gene (nule ili jedinice), da bi ih izjednačili. Na ovaj način operator mutacije održava p jedinica u genetskom kodu i čuva korektnost jedinki.

Tokom izvršavanja GA može se desiti da (skoro) sve jedinke u populaciji imaju isti gen na određenoj poziciji. Takvi geni (može ih biti više) se nazivaju " zaleđenim". Ako je broj zaleđenih bitova *l*, pretraživački prostor postaje 2^l puta manji i mogućnost preuranjene konvergencije rapidno raste. Operatori selekcije i ukrštanja ne mogu promeniti vrednost nijednog zaleđenog bita, a osnovni nivo mutacije često nije dovoljno veliki da bi povratio izgubljene regione prostora pretrage. Ako se osnovni nivo mutacije značajno poveća, genetski algoritam postaje slučajna pretraga (random search). Iz ovih razloga, nivo mutacije se povećava samo na zaleđenim genima , ali ne više od nekoliko puta. U ovoj implementaciji zaleđeni bitovi se mutiraju sa 2.5 puta većom verovatnoćom u odnosu na nezaleđene bitove (1.0/n umesto 0.4/n).

2.3.2.4 Politika zamene generacija

Početna populacija, koja broji 150 jedinki, se generiše na slučajan način. Ovaj pristup obezbeđuje maksimalnu raznovrsnost genetskog materijala i veći gradijent funkcije cilja. Jedna trećina populacije se zamenjuje u svakoj generaciji, dok najboljih 100 jedinki direktno prolaze u narednu generaciju, čuvajući visoko prilagođene gene populacije. Funkcije cilja elitnih jedinki se računaju samo u prvoj generaciji, što obezbeđuje uštedu vremena izračunavanja. Ovaj koncept je u literaturi poznat pod nazivom *stacionarni GA sa elitističkom strategijom* ([Kra00], [Kra01]).

U cilju dobijanja što većeg broja korektnih jedinki u početnoj populaciji, verovatnoća generisanja jedinica u genetskom kodu je postavljena na *p/n*. Jedinke koje imaju *k≠p* jedinica u kodu se popravljaju do korektnih dodajući/brišući |*p-k*| jedinica na/sa kraja genetskog koda. Implementirani genetski operatori čuvaju fiksiran broj habova, tako da se nekorektne jedinke ne pojavljuju u narednim generacijama.

Jedinke koje se više puta pojavljuju u populaciji se uklanjaju u svakoj generaciji postavljajući im fitnes na nulu, tako da operator selekcije onemogućava njihov prolazak u sledeću generaciju. Ovo je vrlo efikasan način za očuvanje raznovrsnosti genetskog materijala i sprečavanje preuranjene konvergencije GA. Jedinke sa istom funkcijom cilja, ali različitim genetskim kodovima u nekim slučajevima mogu dominirati u populaciji. Ako su još njihovi kodovi slični, genetski algoritam može završiti u lokalnom optimumu. Zato je korisno ograničiti njihovo pojavljivanje u populaciji na neku konstantu N_{rv} (N_{rv}=40 u ovoj GA implementaciji).

2.3.2.5 Keširanje GA

Vreme izvršavanja GA se može poboljšati keširanjem [Kra99], [Kra01]. Izračunate vrednosti funkcije cilja jedinki se čuvaju u heš-red tabeli koristeći CRC kodove pridružene njihovim genetskim kodovima. Kada se tokom primene GA ponovo dobije isti genetski kod , vrednost funkcije cilja se uzima iz heš-tabele preko CRC koda. Za keširanje GA primenjena je tehnika najstarijeg korišćenog člana (Least recently used strategy-LRU). U ovom konceptu GA broj keširanih vrednosti funkcije cilja je ograničen na N_{cache}=5000.

2.4 Rezultati

U ovoj sekciji predstavljeni su rezultati predloženog GA i poređenja sa nekim drugim algoritmima za rešavanje UMApHMP. Sva testiranja su izvedena na računaru sa AMD K7/1.33 GHz procesorom sa 256 MB memorije. Algoritam je kodiran u programskom jeziku C. Korišćena su dva tipa ORLIB instanci [Bea96] problema za testiranje algoritma:

CAB (Civil Aeronautics Board) instance, zasnovane na podacima o protoku putnika u avio saobraćaju između gradova SAD. Ovaj skup sadrži 60 instanci sa najviše 25 čvorova (gradova) i najviše 4 haba (terminala). Pretpostavlja se da su parametri χ i δ jednaki 1, dok α uzima vrednosti od 0.2 do 1. Rastojanja između

gradova zadovoljavaju nejednakost trougla i matrica protoka je simetrična. Detaljnije infomacije o CAB instancama mogu se naći u [Bea96] i [Cam96].

AP (Australian Post) instance, dobijene iz studije o australijskom poštanskom sistemu isporuke. Instanca najveće dimenzije iz ovog skupa uključuje 200 čvorova (regione poštanskih brojeva), dok se manje instance sa 10, 20, 25, 50, 100 čvorova mogu dobiti iz najveće agregacijom skupa čvorova. Broj habova (centara sortiranja/konsolidacije poštanskih pošiljki) u testiranim instancama je najviše 20. Matrica protoka W_{ij} nije simetrična i $W_{ii}\neq 0$, pošto polazni i odredišni poštanski region pošiljke može biti isti (sa istim poštanskim brojem). Skup AP instanci se takođe može preuzeti iz [Bea96].

Parametri GA o kojima je bilo reči u prethodnoj sekciji, a koji su se pokazali robusnim i pogodnim za UMApHMP su korišćeni u ovoj implementaciji algoritma. Maksimalni broj generacija je N_{gen}=500 za manje instance i N_{gen}=5000 za instance problema većih dimenzija. Algoritam se takođe zaustavlja ako je najbolja jedinka ili najbolja vrednost funkcije cilja ostala nepromenjena tokom N_{rep}=200, odnosno N_{rep}=2000 uzastopnih generacija. Na svim testiranim instancama, ovi kriterijumi zaustavljanja su obezbeđivali konvergenciju GA ka visoko kvalitetnim rešenjima problema. Samo mala poboljšanja finalnog rešenja GA mogu se očekivati u slučaju produžavanja rada algoritma, što se može videti iz Tabela 2.1-2.3. Glavni cilj u ovom radu bio je nalaženje rešenja koja odgovaraju najboljim rešenjima UMApHMP poznatim u literaturi do sada, tako da su vremena izvršavanja algoritma bila u drugom planu razmatranja.

U Tabeli 2.1 prikazani su rezultati GA pristupa na CAB instancama, dok Tabela 2.2 i Tabela 2.3 sadrže rezultate dobijene na manjim/većim AP instancama respektivno. Da bi rezultati mogli biti korektno upoređeni, GA je izvršen 20 puta na svim instancama, sa izuzetkom AP instanci većih dimenzija sa n≥100 čvorova, na kojima je algoritam primenjen 10 puta jer računanje vrednosti funkcije cilja kod većih instanci zahteva relativno mnogo procesorskog vremena.

U prvoj koloni Tabela 2.1-2.3 date su dimenzije testiranih instanci (n, p i eventualno α). Druga kolona sadrži optimalno rešenje tekuće instance, ako je ono poznato. U suprotnom, "-" je upisana u odgovarajuće polje. Najbolje rešenje GA dato je u sledećoj koloni, sa oznakom *opt* u slučajevima kada GA dostiže unapred poznato optimalno rešenje. Prosečno vreme potrebno algoritmu da dobije najbolju vrednost je dato u *t[s]* koloni, dok t_{tot}*[s]* predstavlja ukupno vreme rada GA za svih 500/5000 generacija. U proseku, najbolje rešenje algoritam je našao posle *gen* generacija.

Kvalitet rešenja u svih 20/10 izvršavanja se računa kao procentualno odstupanje (*gap*) u odnosu na $OPT_{reš.}$ (unapred poznato optimalno rešenje) ili $GA_{najb.}$ (najbolja vrednost GA) sa standardnom devijacijom srednjeg gapa σ (kolone *od[%]* i *σ[%]* respektivno). Poslednje dve kolone u tabelama se odnose na keširanje: *eval* predstavlja prosečan broj neophodnih izračunavanja, dok *kеš[%]* prikazuje uštedu vremena (u procentima) ostvarenu primenom tehnike keširanja. U proseku, umesto 25 000/250 000 izračunavanja funkcije cilja, iskorišćeno je između 42% i 98.5% vrednosti iz heš-tabele.

Tabela 2.1 Rezultatati GA na CAB instancama

n p α	$OPT_{reš.}$	$GA_{najb.}$	t[s]	t_{tot}[s]	gen	od[%]	σ[%]	eval	kеš[%]
20 2 0.2	972.251	opt	0.006	0.046	209	0.0	0.0	296	97.2
20 2 0.4	1013.358	opt	0.004	0.045	210	0.0	0.0	296	97.2
20 2 0.6	1046.895	opt	0.006	0.046	210	0.0	0.0	296	97.2
20 2 0.8	1075.301	opt	0.003	0.044	201	0.0	0.0	297	97.1
20 2 1.0	1090.628	opt	0.004	0.045	204	0.0	0.0	296	97.1
20 3 0.2	712.090	opt	0.013	0.066	213	0.0	0.0	950	91.2
20 3 0.4	803.810	opt	0.014	0.067	213	0.0	0.0	938	91.3
20 3 0.6	884.636	opt	0.016	0.068	215	0.0	0.0	944	91.3
20 3 0.8	948.415	opt	0.009	0.064	208	0.0	0.0	949	91.0
20 3 1.0	975.532	opt	0.013	0.068	209	0.0	0.0	948	91.1
20 4 0.2	568.505	opt	0.027	0.101	226	0.0	0.0	1656	85.5
20 4 0.4	694.557	opt	0.018	0.099	210	0.0	0.0	1603	85.0
20 4 0.6	788.594	opt	0.024	0.102	215	0.0	0.0	1596	85.4
20 4 0.8	870.076	opt	0.022	0.102	215	0.0	0.0	1595	85.4
20 4 1.0	934.083	opt	0.023	0.103	216	0.0	0.0	1586	85.5
25 2 0.2	996.022	opt	0.003	0.053	201	0.0	0.0	410	96.0
25 2 0.4	1072.489	opt	0.003	0.052	201	0.0	0.0	407	96.0
25 2 0.6	1137.081	opt	0.002	0.054	201	0.0	0.0	409	96.0
25 2 0.8	1180.020	opt	0.003	0.054	201	0.0	0.0	410	96.0
25 2 1.0	1206.620	opt	0.003	0.053	201	0.0	0.0	410	96.0
25 3 0.2	752.907	opt	0.022	0.095	218	0.0	0.0	1237	88.8
25 3 0.4	859.636	opt	0.017	0.093	209	0.0	0.0	1236	88.4
25 3 0.6	949.230	opt	0.017	0.094	209	0.0	0.0	1246	88.3
25 3 0.8	1020.037	opt	0.017	0.095	209	0.0	0.0	1249	88.2
25 3 1.0	1062.144	opt	0.021	0.099	213	0.0	0.0	1250	88.4
25 4 0.2	618.483	opt	0.048	0.153	233	0.0	0.0	2028	82.8
25 4 0.4	754.489	opt	0.045	0.153	228	0.0	0.0	1982	82.9
25 4 0.6	866.445	opt	0.023	0.145	209	0.0	0.0	1892	82.2
25 4 0.8	951.755	opt	0.027	0.152	210	0.0	0.0	1910	82.1
25 4 1.0	1006.657	opt	0.029	0.161	210	0.0	0.0	2021	81.1

Tabela 2.2 Rezultati GA na AP instancama

n p	$OPT_{reš.}$	$GA_{najb.}$	t[s]	t_{tot}[s]	gen	od[%]	σ[%]	eval	keš[%]
10 2	163603.94	opt	0.001	0.037	201	0.000	0.000	156	98.5
10 3	131581.79	opt	0.001	0.038	201	0.000	0.000	268	97.4
10 4	107354.73	opt	0.004	0.040	204	0.000	0.000	351	96.6
10 5	86028.88	opt	0.003	0.042	201	0.000	0.000	384	96.2
10 6	72427.73	opt	0.002	0.042	201	0.000	0.000	341	96.6
10 7	63466.81	opt	0.002	0.041	202	0.000	0.000	273	97.3
10 8	54628.75	opt	0.002	0.041	202	0.000	0.000	196	98.1
20 2	168599.79	opt	0.004	0.045	201	0.000	0.000	297	97.1
20 3	148048.30	opt	0.012	0.065	210	0.000	0.000	883	91.7
20 4	131665.43	opt	0.017	0.091	213	0.000	0.000	1461	86.5
20 5	118934.97	opt	0.020	0.119	210	0.000	0.000	1809	83.0
20 6	107005.85	opt	0.045	0.161	226	0.000	0.000	2239	80.4
20 7	97697.75	opt	0.031	0.184	209	0.000	0.000	2301	78.4
20 8	91454.83	opt	0.060	0.211	227	0.000	0.000	2313	79.8
25 2	171298.10	opt	0.003	0.051	201	0.000	0.000	411	96.0
25 3	151080.66	opt	0.016	0.088	209	0.000	0.000	1130	89.3
25 4	135638.58	opt	0.028	0.139	212	0.000	0.000	1851	82.8
25 5	120581.99	opt	0.051	0.208	223	0.000	0.000	2464	78.2
25 6	110835.82	opt	0.094	0.277	246	0.000	0.000	2913	76.5
25 7	103880.23	opt	0.139	0.374	257	0.000	0.000	3461	73.2
25 8	97795.59	opt	0.155	0.453	252	0.000	0.000	3750	70.5
40 2	173415.96	opt	0.025	0.102	211	0.000	0.000	783	92.7
40 3	155458.61	opt	0.073	0.245	226	0.000	0.000	2062	82.0
40 4	140682.74	opt	0.131	0.452	234	0.000	0.000	3265	72.5
40 5	130384.74	opt	0.358	0.788	299	0.024	0.060	4667	68.8
50 2	174390.03	opt	0.061	0.173	228	0.000	0.000	1078	90.6
50 3	156014.72	opt	0.240	0.511	288	0.000	0.000	3069	78.6
50 4	141153.38	opt	0.384	0.885	285	0.000	0.000	4419	69.0
50 5	129412.60	opt	0.571	1.282	301	0.015	0.036	5280	65.0

Tabela 2.3 Rezultati GA na AP instancama većih dimenzija

n p	$OPT_{reš.}$	$GA_{najb.}$	t[s]	t_{tot}[s]	gen	od[%]	σ[%]	eval	keš[%]
40 6	122171.26	opt	0.247	4.834	2039	0.000	0.000	23349	77.1
40 7	-	116036.38	0.467	6.002	2086	0.000	0.000	25307	75.8
40 8	-	109971.92	0.579	7.655	2085	0.000	0.000	28348	72.9
40 9	-	104212.42	0.884	9.010	2127	0.000	0.000	29598	72.2
40 10	-	99452.67	0.779	9.491	2085	0.000	0.000	27863	73.3
50 6	121671.76	opt	1.537	9.150	2284	0.000	0.000	31036	72.9
50 7	-	115911.64	4.872	15.725	2851	0.000	0.000	46503	67.4
50 8	-	109926.60	4.294	17.188	2591	0.000	0.000	44043	66.0
50 9	-	104968.27	2.869	17.252	2298	0.000	0.000	38930	66.2
50 10	100508.95	opt	4.333	21.136	2412	0.000	0.000	42743	64.7
50 11	-	96186.22	5.294	24.675	2454	0.000	0.000	44999	63.4
50 12	-	93171.96	3.714	23.870	2267	0.000	0.000	39458	65.2
50 13	-	90409.79	4.255	27.221	2281	0.000	0.000	41079	64.1
50 14	-	87654.61	3.972	29.098	2238	0.000	0.000	40315	64.1
50 15	-	85032.89	7.463	35.493	2456	0.000	0.000	45615	62.9
50 20	-	73490.33	2.824	39.859	2094	0.000	0.000	38133	63.6
100 2	176245.38	opt	0.639	2.736	2089	0.000	0.000	4088	96.1
100 3	157869.93	opt	2.195	13.227	2207	0.000	0.000	21017	81.0
100 4	143004.31	opt	9.007	32.848	2652	0.000	0.000	44346	66.6
100 5	133482.57	opt	20.067	54.389	3097	0.000	0.000	60475	60.9
100 6	-	126107.56	58.421	99.973	4350	0.000	0.000	94424	56.6
100 7	-	120165.15	45.945	100.118	3553	0.011	0.024	80659	54.6
100 8	-	114295.92	77.750	125.793	3891	0.228	0.355	87852	54.8
100 9	-	109448.87	54.651	126.037	3409	0.002	0.005	77693	54.6
100 10	-	104794.05	63.355	146.263	3421	0.001	0.002	79849	53.4
100 15	-	88882.05	150.193	270.956	4004	0.093	0.162	93755	53.1
100 20	-	79191.02	195.747	377.160	3828	0.139	0.152	96737	49.5
200 2	178093.99	opt	8.123	35.686	2129	0.000	0.000	10048	90.6
200 3	159725.11	opt	43.393	174.900	2520	0.000	0.000	40939	67.6
200 4	-	144508.20	172.663	376.815	3585	0.001	0.002	78983	56.0
200 5	-	136761.83	357.326	562.245	4231	0.096	0.092	103391	51.2
200 6	-	129560.60	393.868	681.338	4281	0.046	0.062	111529	47.9
200 7	-	123609.44	460.543	766.016	4219	0.051	0.070	112515	46.7
200 8	-	117709.98	566.177	879.377	4237	0.213	0.189	115253	45.8
200 9	-	112380.66	869.886	1096.180	4809	0.066	0.146	131684	45.3
200 10	-	107846.82	847.216	1157.049	4591	0.090	0.189	127817	44.4
200 15	-	92669.64	1246.186	1750.105	4699	0.397	0.275	135060	42.6
200 20	-	83385.94	1935.840	2425.588	4924	0.169	0.232	142223	42.3

Metode za rešavanje UMApHMP predložene u [OK96], [Sko94] i [Soh98] daju rezultate dobijene samo na skupu CAB instanci ($n \leq 25$, $p \leq 5$). Na ovim istancama GA dostiže optimalna rešenja za manje od 0.17 sekundi, tako da je jako teško (skoro nemoguće) napraviti poređenja u korišćenim kompjuterskim okruženjima. Iz istog razloga, teško je izvesti jasne zaključke iz poređenja sa [Ern98] i [Bol04] na instancama manjih dimenzija, ali ovi radovi daju rezultate predloženih metoda i na većim AP instancama, tako da je opšti pregled rezultata na nekoliko izabranih AP instanci prikazan u Tabeli 2.4.

Prva kolona u Tabeli 2.4 odnosi se na dimenzije izabrane AP instance (n i p), dok naredne kolone sadrže:

- odstupanje (gap) vrednosti $GA_{najb.}$ od optimalnog rešenja i prosečno AMD K7/1.33 GHz vreme izvršavanja,

- rezultate heuristike najkraćeg puta (odstupanje najboljeg rešenja od optimalnog i vreme izvršavanja) koju su dali Ernst i Krishnamoorthy u [Ern98], testirane na računaru DEC 3000/700 sa alpha 200 MHz procesorskim okruženjem (označena sa EK-SP-heur u tabeli),
- rezultate egzaktnog branch-and–bound metoda (broj čvorova branch-and-bound drveta i vreme izvršavanja), takođe predloženog u [Ern98] i testiranog na DEC 3000/700 (EK-BnB-opt u tabeli),
- rezultate egzaktnog branch-and-bound metoda II (odstupanje početnog rešenja dobijenog heuristikom, broj čvorova drveta i vreme izvršavanja) koji su dali Boland i saradnici u radu [Bol04], a koji je testiran na DEC personalnom računaru sa alpha procesorom na 500 MHz (oznaka: BKEE-metod-II).

Tabela 2.4 Poređenja GA sa drugim metodama

n p	GA		EK-SP-heur		EK-BnB-opt		BKEE-metod-II		
	od[%]	t_{AMD}[s]	od[%]	t_{DEC}[s]	br.čv.	t_{DEC}[s]	$od_{poč.}$[%]	br.čv.	t_{DEC}[s]
25 4	0.00	0.139	0.77	0.40	708	1.68	4.09	43	245.07
50 5	0.00	1.282	0.07	10.95	10187	143.48	3.46	89	27597
50 10	0.00	21.136	0.14	66.71	2125737	57243	-	-	-
100 5	0.00	54.389	0.00	168.49	153266	7688	-	-	-
200 3	0.00	174.900	0.00	632.70	25349	3636.6	-	-	-

Kao što se vidi iz Tabela 2.1-2.3, GA dostiže sva do sada poznata optimalna rešenja. Tabela 2.4 pokazuje da je vreme izvršavanja predloženog genetskog algoritma slično vremenima ostalih heuristika, dok je kvalitet dobijenih rešenja nešto bolji. Egzaktne metode zahtevaju znatno više procesorskog vremena i pošto eksponencijalno rastu sa dimenzijom problema, ne mogu se testirati na instancama problema velikih dimenzija (n=200, p=20). Bilo bi zanimljivo porediti postojeće heuristike za UMApHMP na instancama ovih dimenzija, ali, na žalost, ti rezultati se do sada ne mogu naći u literaturi.

3. PROBLEM P-HAB CENTRA NEOGRANIČENIH KAPACITETA SA VIŠESTRUKIM ALOKACIJAMA

U ovom poglavlju razmatra se još jedna varijanta hab lokacijskih problema, u literaturi poznata pod nazivom *Uncapacitated multiple allocation p-hub center problem (UMApHCP).* Cilj ovog problema je uspostaviti p habova na nekim od n potencijalnih lokacija tako da se minimizuje maksimalno rastojanje, odnosno troškovi ili vreme transporta između bilo koja dva čvora u mreži preko jednog ili više habova. UMApHCP model se može uspešno primeniti u hab mrežama kod kojih je rastojanje snabdevač-korisnik u najgorem slučaju ekstremno veliko. U tim slučajevima p-medijan formulacija istog problema, sa istim pretpostavkama i ograničenjima, može dovesti do nezadovoljavajućih rezultata. UMApHCP ima značajnu primenu u dizajniranju sistema isporuke, službi hitne pomoći (vatrogasci, policija, medicinska pomoć), transporta putnika itd. U ovom, a i drugim p-centar hab problemima, korišćen je *minmax* kriterijum u cilju minimizacije maksimalnog vremena isporuke pošiljki (posebno onih koje je zbog kvarljivosti ili osetljivosti potrebno hitno dostaviti korisniku), vremena čekanja putnika u saobraćaju ili ljudi kojima je neophodna hitna pomoć različite vrste.

3.1 Postojeći načini rešavanja i formulacija problema

Sudeći po zastupljenosti u literaturi, p-centar hab lokacijski problemi su manje proučavani u poređenju sa p-medijan hab problemima. Dve najstarije definicije p-centar hab problema nalaze se u [OK91] i [Cam94]. U poslednjem, Campbell daje formulaciju koristeći $O(n^4)$ promenljivih i $O(n^3)$ ograničenja, ali kasnije predlaže preformulaciju problema u vidu linearnog programa. Dobijena LP relaksacija je i dalje teška za rešavanje u slučaju većeg skupa čvorova u mreži.

Jedini rad koji nalazimo u literaturi do sada, koji proučava UMApHCP, je [Ern04b]. Autori navode dve celobrojne formulacije UMApHCP, a jedna od njih je korišćena u ovom radu. Oni takođe predlažu dve heuristike za rešavanje p-centar hab lokacijskog problema sa jednostrukom/višestrukom alokacijskom šemom, kao i branch-and-bound algoritam (BnB), zasnovan na metodi najkraćeg puta. Rezultati dobijeni testiranjem na standardnim ORLIB (CAB i AP instancama) pokazuju da obe heuristike brzo dolaze do rešenja, ali da gap u nekim slučajevima prelazi 10% (na primer, AP instanca n=25, p=3).

Neka je ponovo $I=\{1,...,n\}$ skup od n različitih čvorova mreže (lokacije snabdevača, korisnika i potencijalnih habova). Označimo sa C_{ij} rastojanje između

i-tog i *j*-tog čvora pri čemu važi $C_{ij} = C_{ji}$ i nejednakost trougla takođe može biti zadovoljena [Cam02].

Protok u mreži koji polazi od snabdevača *i* se usmerava ka odgovarajućem habu, zatim se realizuje preko jednog ili više hab čvorova i na kraju distribuira korisniku *j.* U UMApHCP treba locirati tačno p hab čvorova. Binarne promenljive H_k, X_j^{ik} i Y_{lj} se koriste u formulaciji na sledeći način:

- H_k = 1 akko je hab uspostavljen na lokaciji *k*
- X_j^{ik} = 1 akko je snabdevač *i* pridružen habu *k* u uređenom paru snabdevač-korisnik (*i*, *j*)
- Y_{lj}^{i} = 1 akko je korisnik *j* pridružen habu *l* u uređenom paru snabdevač-korisnik (*i*, *j*).

Parametar α uzima vrednosti iz intervala [0, 1], tako da $1-\alpha$ predstavlja faktor uštede za transport između habova, kao u UMApHMP. Definišimo $C_{max} = max \{C_{ij} \mid i, j \in I\}$ i neka je *z* slobodna promenljiva koja predstavlja funkciju cilja. Uzimajući u obzir gornju notaciju, matematički zapis UMApHCP je:

$$min\ z \tag{3.1}$$

uz uslove:

$$\sum_{k=1}^{n} H_k = p \tag{3.2}$$

$X_j^{ik} \le H_k$ za svako $i, j, k \in I$ (3.3)

$Y_{lj}^{i} \le H_l$ za svako $i, j, l \in I$ (3.4)

$$\sum_{k=1}^{n} X_j^{ik} = 1 \quad \text{za svako } i, j \in I \tag{3.5}$$

$$\sum_{l=1}^{n} Y_{lj}^{i} = 1 \quad \text{za svako } i, j \in I \tag{3.6}$$

$$z \ge \sum_{k=1}^{n} (C_{ik} + \alpha \cdot C_{kl}) \cdot X_j^{ik} + \sum_{m=1}^{n} C_{mj} \cdot Y_{mj}^{i} - \alpha \cdot (1 - Y_{lj}^{i}) \cdot C_{max} \quad \text{za svako } i, j, l \in I \tag{3.7}$$

$H_k, X_j^{ik}, Y_{lj}^{i} \in \{0, 1\}$ za svako $i, j, k, l \in I$ (3.8)

Cilj UMApHCP je minimizacija maksimalnih troškova transporta između bilo koja dva čvora u mreži (3.1). Uslov (3.2) određuje tačan broj habova koje treba uspostaviti - *p*. Ograničenja (3.3)-(3.6) obezbeđuju da svaki korisnik/snabdevač bude pridružen tačno jednom, prethodno lociranom habu. Donja granica za promenljivu z funkcije cilja je data uslovom (3.7). Konačno, (3.8) ukazuje na binarnu reprezentaciju promenljivih H_k, X_j^{ik} i Y_{lj}^{i}. UMApHCP je NP-težak, što je dokazano u [Ern04b].

3.2 Predloženi genetski algoritam

GA implementacija u ovom radu za rešavanje UMApHCP takođe koristi binarnu reprezentaciju jedinki. Genetski kod je dužine n i svaki gen uzima vrednost 1 ako je na toj poziciji uspostavljen hab, a 0 ako nije. Niz (H_k) se jedini dobija iz genetskog koda, dok se vrednosti promenljivih X_j^{ik} i Y_{lj}^{i} dobijaju tokom računanja funkcije cilja (jer korisnici/snabdevači mogu biti pridruženi isključivo uspostavljenim habovima).

Ako se fiksira podskup habova H_k, ispostavlja se da UMApHCP postaje ekvivalentan sa n^2 problema najkraćih puteva, koji se mogu rešiti u $O(n^2*p)$ vremenu, što se vidi iz rada [Ern98a].

Predloženi GA koristi iste genetske operatore i GA parametre opisane u 2.3. Takođe je primenjena ista politika zamene generacija, kao i keširanje. Algoritam je kodiran u programskom jeziku C i testiran na dva skupa modifikovanih ORLIB instanci (CAB i AP), preuzetih iz [Ern04a] i [Ern04b]. Svi testovi su izvedeni na računaru sa AMD K7/1.33 Ghz procesorom sa 256 MB memorije. Maksimalan broj generacija GA je N_{gen}=1000 za manje instance i N_{gen}=10 000 za instance problema većih dimenzija. Algoritam se takođe zaustavlja ako je najbolja jedinka ili najbolja vrednost funkcije cilja ostala nepromenjena tokom N_{rep}=400, odnosno N_{rep}=4000 uzastopnih generacija. Koristeći ove kriterijume GA dostiže optimalna rešenja koja su do sada poznata u literaturi na testiranim instancama.

3.3 Rezultati GA

U Tabeli 3.1 prikazani su rezultati predložene GA implementacije na CAB instancama UMApHCP, dok Tabela 3.2 i Tabela 3.3 sadrže rezultate dobijene na skupu AP instanci manjih/većih dimenzija respektivno. GA je testiran 20 puta na svakoj instanci iz Tabela 1-2, sem instanci većih dimenzija u Tabeli 3 koje je GA rešavao samo 10 puta, zbog relativno velikog vremena izračunavanja funkcije cilja u slučaju većeg broja čvorova.

U prvoj koloni Tabela 3.1-3.3 date su dimenzije odgovarajuće instance (n, p i eventualno α). Druga kolona sadrži optimalno rešenje tekuće instance, ukoliko je unapred poznato (ako nije, upisana je "-"). Najbolja vrednost koja je dobijena primenom GA nalazi se u sledećoj koloni, sa oznakom *opt* u slučajevima kada je GA dostigao poznato optimalno rešenje. U koloni *t[s]* nalazi se prosečno vreme potrebno algoritmu da nađe najbolju vrednost, a t_{tot}*[s]* kolona sadrži ukupno vreme za koje GA prolazi kroz svih 1000/10 000 generacija. Najbolja vrednost GA je dostignuta posle *gen* generacija u proseku.

Kvalitet rešenja dobijenog u svih 20/10 izvršavanja algoritma se ocenjuje preko procentualne vrednosti odstupanja, označenog sa *od[%],* u odnosu na optimalno rešenje $OPT_{reš.}$ ili $GA_{najb.}$ (u zavisnosti od toga da li je optimalno rešenje unapred poznato ili ne). Standardna devijacija prosečne vrednosti gapa σ nalazi se u istoj koloni. Poslednje dve kolone u tabelama se odnose na keširanje i sadrže: *eval* - prosečan broj neophodnih izračunavanja, *kеš[%]* - uštedu vremena (u procentima) ostvarenu primenom tehnike keširanja. U proseku, umesto 50000/5000000 izračunavanja funkcije cilja, iskorišćeno je između 36.8% i 99.2% vrednosti iz heš-tabele.

Tabela 3.1 Rezultatati GA na CAB instancama

n p α	$OPT_{reš.}$	$GA_{najb.}$	t[s]	t_{tot}[s]	gen	od[%]	σ[%]	eval	keš[%]
20 2 0.2	1892.99	opt	0.005	0.087	407	0.000	0.000	297	98.5
20 2 0.4	2027.69	opt	0.005	0.087	413	0.000	0.000	297	98.6
20 2 0.6	2248.13	opt	0.005	0.087	410	0.000	0.000	297	98.6
20 2 0.8	2335.99	opt	0.002	0.083	401	0.000	0.000	298	98.5
20 2 1.0	2600.08	opt	0.002	0.084	401	0.000	0.000	298	98.5
20 3 0.2	1551.25	opt	0.014	0.110	416	0.000	0.000	1020	95.1
20 3 0.4	1738.32	opt	0.006	0.104	406	0.000	0.000	1048	94.9
20 3 0.6	1916.16	opt	0.011	0.111	406	0.000	0.000	1101	94.6
20 3 0.8	2195.22	opt	0.012	0.111	416	0.000	0.000	1058	94.9
20 3 1.0	2600.08	opt	0.004	0.110	401	0.000	0.000	1134	94.4
20 4 0.2	1287.78	opt	0.015	0.148	409	0.000	0.000	1960	90.5
20 4 0.4	1472.71	opt	0.030	0.158	433	0.000	0.000	2042	90.6
20 4 0.6	1808.70	opt	0.013	0.149	406	0.000	0.000	1986	90.3
20 4 0.8	2128.11	opt	0.021	0.155	417	0.000	0.000	1969	90.6
20 4 1.0	2600.08	opt	0.005	0.190	401	0.000	0.000	2951	85.4
25 2 0.2	2049.48	opt	0.011	0.097	416	0.000	0.000	416	98.0
25 2 0.4	2402.55	opt	0.010	0.095	409	0.000	0.000	416	98.0
25 2 0.6	2558.74	opt	0.009	0.098	409	0.000	0.000	418	98.0
25 2 0.8	2714.93	opt	0.006	0.096	410	0.000	0.000	418	98.0
25 2 1.0	2739.22	opt	0.003	0.098	401	0.000	0.000	418	97.9
25 3 0.2	1911.60	opt	0.007	0.144	401	0.000	0.000	1554	92.3
25 3 0.4	2064.67	opt	0.040	0.166	484	0.833	0.627	1578	93.3
25 3 0.6	2243.77	opt	0.014	0.146	407	0.000	0.000	1426	93.1
25 3 0.8	2515.58	opt	0.019	0.146	413	0.000	0.000	1398	93.3
25 3 1.0	2725.79	opt	0.004	0.149	401	0.000	0.000	1653	91.8
25 4 0.2	1619.48	opt	0.061	0.233	466	2.635	1.109	2671	88.4
25 4 0.4	1774.45	opt	0.076	0.255	467	0.000	0.000	3070	86.9
25 4 0.6	2127.13	opt	0.103	0.292	537	0.000	0.000	3421	87.0
25 4 0.8	2437.71	opt	0.034	0.220	420	0.000	0.000	2498	88.2
25 4 1.0	2725.79	opt	0.009	0.304	401	0.000	0.000	4027	80.1

Tabela 3.2 Rezultati GA na AP instancama

n	p	$OPT_{reš.}$	$GA_{najb.}$	t[s]	t_{tot}[s]	gen	od[%]	σ[%]	eval	keš[%]
10	2	39922.11	opt	0.001	0.079	401	0.000	0.000	155	99.2
10	3	32713.94	opt	0.001	0.071	401	0.000	0.000	268	98.7
10	4	31577.96	opt	0.001	0.074	401	0.000	0.000	354	98.2
10	5	30371.32	opt	0.002	0.077	401	0.000	0.000	378	98.1
20	2	45954.15	opt	0.005	0.086	408	0.000	0.000	297	98.6
20	3	40909.59	opt	0.012	0.108	412	0.000	0.000	1076	94.8
20	4	38320.25	opt	0.026	0.149	423	0.000	0.000	1876	91.2
20	5	37868.15	opt	0.007	0.182	401	0.000	0.000	2318	88.5
20	10	37868.15	opt	0.013	0.726	401	0.000	0.000	6900	65.8
25	2	51533.30	opt	0.012	0.097	419	0.000	0.000	417	98.0
25	3	45552.50	opt	0.038	0.154	449	0.000	0.000	1547	93.1
25	4	45552.50	opt	0.022	0.208	409	0.000	0.000	2410	88.3
25	5	45552.50	opt	0.020	0.335	404	0.000	0.000	3893	80.9
25	10	45552.50	opt	0.021	1.270	401	0.000	0.000	8465	58.1
40	2	61140.80	opt	0.015	0.149	405	0.000	0.000	826	96.0
40	3	56309.88	opt	0.133	0.412	480	0.000	0.000	3272	86.3
40	4	51279.14	opt	0.394	0.810	617	0.000	0.000	5515	81.4
40	5	49741.20	opt	0.065	0.732	408	0.000	0.000	4299	79.1
40	10	49741.20	opt	0.049	3.479	401	0.000	0.000	10430	48.4
50	2	61179.03	opt	0.040	0.219	415	0.000	0.000	1116	94.7
50	3	56729.94	opt	0.192	0.695	466	0.000	0.000	4190	82.0
50	4	52905.77	opt	0.447	1.241	508	0.000	0.000	6219	75.7
50	5	50707.87	opt	0.342	1.571	451	0.000	0.000	6558	71.1
50	10	50707.87	opt	0.073	5.558	401	0.000	0.000	10767	46.7
100	2	63197.10	opt	0.217	1.218	420	0.000	0.000	2684	87.3
100	3	57925.66	opt	1.994	4.575	589	0.000	0.000	8122	72.3

Tabela 3.3 Rezultati GA na AP instancama većih dimenzija

n	p	$OPT_{reš.}$	$GA_{najb.}$	t[s]	t_{tot}[s]	gen	od[%]	σ[%]	eval	keš[%]
100	5	53949.33	opt	18.410	77.079	5227	0.609	0.321	89874	65.6
100	10	51860.03	opt	2.545	174.825	4034	0.000	0.000	99266	50.8
100	15	-	51860.03	1.744	303.285	4011	0.000	0.000	110306	45.1
100	20	-	51860.03	1.188	416.744	4004	0.000	0.000	111164	44.5
200	2	-	67083.28	9.110	64.507	4307	0.000	0.000	26516	87.7
200	3	-	62945.55	106.379	321.421	5740	0.218	0.352	103375	63.9
200	5	-	57419.32	222.719	622.917	6022	0.252	0.796	141261	53.1
200	10	-	55958.75	20.577	988.242	4057	0.000	0.000	118752	41.5
200	15	-	55958.75	19.356	1564.493	4033	0.000	0.000	126902	37.1
200	20	-	55958.75	17.494	2039.444	4021	0.000	0.000	127229	36.8

3.4 Poređenja sa postojećim metodama

U ovoj sekciji data su poređenja predloženog GA pristupa sa heuristikom EH2, boljim od dva heuristička metoda predložena u [Ern04b] i egzaktnim branch-and-bound algoritmom EBnB iz istog rada. Numerički rezultati EH2 i EBnB metoda, koje su testirane na DEC Alpha računaru, preuzeti su iz pomenutog rada.

Prva kolona u Tabelama 3.4 i 3.5 odnose se na dimenzije izabranih CAB i AP instanci na kojima se vrše poređenja (n, p i eventualno α), a naredne kolone sadrže:

- odstupanje rešenja $GA_{najb.}$ od optimalnog (oznaka $od_{najb.}$*[%]*) i prosečno AMD K7/1.33GHz vreme izvršavanja (oznaka t_{AMD}*[s]*),
- EH2 odstupanje (*od[%]*) i DEC vreme izvršavanja (t_{DEC}*[s]),*
- EBnB DEC vreme izvršavanja (t_{DEC}*[s*]).

Tabela 3.4 Poređenja sa EH2 i EBnB na CAB instancama

n p α	GA		EH2		EBnB
	$od_{najb.}$[%]	t_{AMD}[s]	od[%]	t_{DEC}[s]	t_{DEC}[s]
20 2 0.2	0.00	0.087	0.00	< 0.01	0.01
20 2 0.4	0.00	0.087	0.00	0.01	0.01
20 2 0.6	0.00	0.087	0.00	0.01	0.01
20 2 0.8	0.00	0.083	0.00	< 0.01	0.01
20 2 1.0	0.00	0.084	0.00	< 0.01	0.01
20 3 0.2	0.00	0.110	0.00	0.01	0.02
20 3 0.4	0.00	0.104	0.00	0.01	0.02
20 3 0.6	0.00	0.111	0.00	0.01	0.03
20 3 0.8	0.00	0.111	0.00	0.01	0.03
20 3 1.0	0.00	0.110	0.00	0.01	0.01
20 4 0.2	0.00	0.148	0.00	0.02	0.04
20 4 0.4	0.00	0.158	7.76	0.02	0.04
20 4 0.6	0.00	0.149	2.29	0.02	0.14
20 4 0.8	0.00	0.155	3.13	0.02	0.04
20 4 1.0	0.00	0.190	0.00	0.02	0.02
25 2 0.2	0.00	0.097	0.00	0.01	0.01
25 2 0.4	0.00	0.095	0.00	0.01	0.01
25 2 0.6	0.00	0.098	0.00	0.01	0.01
25 2 0.8	0.00	0.096	0.00	0.01	0.02
25 2 1.0	0.00	0.098	0.00	0.01	0.01
25 3 0.2	0.00	0.144	0.17	0.02	0.05
25 3 0.4	0.00	0.166	1.28	0.02	0.04
25 3 0.6	0.00	0.146	0.00	0.03	0.04
25 3 0.8	0.00	0.146	0.00	0.03	0.04
25 3 1.0	0.00	0.149	0.00	0.02	0.03
25 4 0.2	0.00	0.233	5.94	0.05	0.10
25 4 0.4	0.00	0.255	9.46	0.05	0.09
25 4 0.6	0.00	0.292	0.00	0.05	0.10
25 4 0.8	0.00	0.220	0.37	0.06	0.12
25 4 1.0	0.00	0.304	0.00	0.04	0.06

Tabela 3.5. Poređenja sa EH2 i EBnB na AP instancama

n p	GA		Ernst heur.		Ernst BnB
	$od_{najb.}$[%]	t_{AMD}[s]	od[%]	t_{DEC}[s]	t_{DEC}[s]
10 2	0.00	0.079	0.00	< 0.01	< 0.01
10 3	0.00	0.071	0.00	< 0.01	< 0.01
10 4	0.00	0.074	3.54	< 0.01	0.02
10 5	0.00	0.077	0.00	< 0.01	0.02
20 2	0.00	0.086	0.00	< 0.01	0.01
20 3	0.00	0.108	6.09	0.01	0.02
20 4	0.00	0.149	0.99	0.02	0.04
20 5	0.00	0.182	0.00	0.03	0.04
20 10	0.00	0.726	0.00	0.12	0.15
25 2	0.00	0.097	0.00	0.01	0.01
25 3	0.00	0.154	11.10	0.02	0.03
25 4	0.00	0.208	0.00	0.04	0.06
25 5	0.00	0.335	0.00	0.07	0.10
25 10	0.00	1.270	0.00	0.31	0.40
40 2	0.00	0.149	0.00	0.05	0.08
40 3	0.00	0.412	1.75	0.14	0.25
40 4	0.00	0.810	0.00	0.27	0.55
40 5	0.00	0.732	0.00	0.42	1.09
40 10	0.00	3.479	0.00	2.18	139.6
50 2	0.00	0.219	0.00	0.12	0.18
50 3	0.00	0.695	0.00	0.30	0.60
50 4	0.00	1.241	2.42	0.60	1.72
50 5	0.00	1.571	4.33	0.97	5.26
50 10	0.00	5.558	0.00	5.38	14 587
100 2	0.00	1.218	0.00	1.55	2.53
100 3	0.00	4.575	4.00	4.45	14.47
100 5	0.00	77.079	0.76	16.51	536.8
100 10	0.00	174.825	-	-	> 3 dana

Podaci u Tabelama 3.4 i 3.5 pokazuju da je vreme izvršavanja GA nešto veće u poređenju sa EH2, ali da GA dostiže optimalna rešenja na svim instancama, za razliku od EH2 heuristike. EH2 rešenja imaju odstupanja u 8 slučajeva od 30 izabranih CAB instanci i u 10 od 28 AP instanci. U nekim slučajevima odstupanja najboljih EH2 rešenja su velika, recimo za AP n = 10, p = 3 i CAB n = 25, p = 4, α = 0.4 instance problema .

EBnB metod daje rešenje u relativno kratkom vremenu za AP n < 50 i/ili p < 10, ali vreme računanja brzo raste sa povećanjem n, a drastično sa porastom p. Za AP n = 100 i p = 10 EBnB metod nije završio rad čak ni posle 3 dana računanja.

Kao što se može videti iz gore prikazanih rezultata, iako i EH2 i EBnB uglavnom imaju dobre performanse, predloženi GA predstavlja vredan dodatak postojećim metodama za rešavanje UMApHCP. Ali, njegove prednosti se mogu uočiti tek na instancama problema većih dimenzija. Da bi se kvalitet GA pristupa u potpunosti ocenio, bilo bi korisno porediti GA sa drugim metodama na većim instancama, ali takvi rezultati za sada ne postoje u literaturi.

4. DISKRETNO UREĐEN PROBLEM MEDIJANE

Široka primena diskretnih lokacijskih problema u praksi prirodno je dovela do potrebe za razvijanjem novih i fleksibilnih lokacijskih modela koji dobro opisuju širu klasu lokacijskih problema. Diskretno uređen problem medijane (Discrete Ordered Median Problem - DOMP) predstavlja generalizaciju klasičnih diskretnih lokacijskih problema kao što su p-medijan, p-centar i μ-centdian problem. DOMP uopštava probleme koji se bave uspostavljanjem fiksiranog broja snabdevača u mreži na izabranim lokacijama iz unapred datog skupa, pod uslovom da svaki korisnik može biti pridružen samo jednom (prethodno lociranom) snabdevaču i pritom su poznati troškovi transporta za svaki par korisnik-snabdevač. Problemi koje DOMP obuhvata imaju različite funkcije cilja. Za problem p-medijane potrebno je minimizovati sumu transportnih troškova između svih klijenata i odgovarajućih snabdevača. Problem p-centra ima za cilj smanjenje maksimalne cene transporta među svim parovima korisnik-snabdevač. Funkcija cilja centdian lokacijskog problema je konveksna kombinacija funkcija cilja p-medijane i p-centra, kako bi se istovremeno minimizovali ukupni i maksimalni troškovi transporta od svakog snabdevača do njemu pridruženog korisnika. Ove i druge DOMP lokacijske probleme često srećemo u literaturi, a za svaki od njih razvijene su različite metode koje ih nezavisno rešavaju. U ovom poglavlju opisan je GA pristup koji efikasno rešava sve do sada poznate DOMP instance, čak i one sa većim brojem čvorova, što u dosadašnjoj literaturi koja se bavi ovim problemom nije uvek bio slučaj.

4.1 Matematička formulacija

DOMP je prvi definisao Nickel u svom radu [Nic01] uvodeći novi tip funkcije cilja koja generalizuje najpoznatije gore pomenute funkcije. Nickel daje nelinearnu i linearnu celobrojnu formulaciju problema, od kojih poslednja koristi $O(n^3)$ promenljivih i $O(n^3)$ ograničenja, gde je n broj potencijalnih lokacija snabdevača. Boland i saradnici u [Bol03] predlažu dve celobrojne linearne formulacije (ILP formulations) DOMP-a koje uključuju po $O(n^3)$ promenljivih. Jedna od njih zahteva $O(n^3)$ uslova, dok se druga formulacija koja ima $O(n^2)$ ograničenja koristi i u ovom radu.

Označimo sa *I={1,...,n}* skup različitih čvorova mreže, pri čemu svaki čvor predstavlja lokaciju korisnika ili potencijalnog snabdevača. Neka je $C=[C_{ij}]$, $i, j=1,..,n$ data nenegativna matrica transportnih troškova između korisnika *i* i snabdevača *j*. Ne gubeći na opštosti, možemo pretpostaviti da je broj potencijalnih lokacija snabdevača

jednak broju klijenata. Neka je *p≤n* broj lokacija snabdevača koje treba uspostaviti. Rešenje problema je skup X indeksa lociranih snabdevača, i važi: $X \subseteq I$ i $|X| = p$. U rešenju X svaki klijent *i* će biti uslužen od snabdevača *j* sa najnižom cenom transporta, to jest:

$$C_{ij} = C_i(X) := \min_{k \in X} C_{ik} \tag{4.1}$$

DOMP se bitno razlikuje od drugih diskretnih lokacijskih problema (koji takođe podrazumevaju fiksiran broj snabdevača i usluživanje svakog korisnika sa tačno jedne lokacije) po složenosti svoje funkcije cilja. Da bi se izračunala vrednost ove funkcije za dato rešenje X, potrebno je najpre sortitati transportne troškove za snabdevanje klijenata $C_1(X),...,C_n(X)$ u neopadajući niz. Označimo zatim sa σ_X permutaciju skupa *{1,...,n}* za koju važe nejednakosti:

$$C_{\sigma_X(1)}(X) \le C_{\sigma_X(2)}(X) \le ... \le C_{\sigma_X(n)}(X) \tag{4.2}$$

Ovakva permutacija σ_X se naziva *validnom* za DOMP. Neka je $\Lambda = (\lambda_1,..,\lambda_n)$, $\lambda_i \ge 0,$ i=1,...,n vektor koeficijenata transportnih troškova. Pri računanju funkcije cilja skalarno množimo vektor Λ i uređeni (u neopadajućem poretku) vektor troškova $C_{\le} = (C_{\sigma_X(1)}(X), C_{\sigma_X(2)}(X),..., C_{\sigma_X(n)}(X))$, što znači da je za svako *i=1,...,n, i*-ta po redu najniža cena snabdevanja klijenata $C_{\sigma_X(i)}(X)$ pomnožena koeficijentom λ_i.

Imajući u vidu gornju notaciju, DOMP se matematički može zapisati na sledeći način:

$$\min_{X \subseteq I, |X| = p} \sum_{i=1}^{n} \lambda_i C_{\sigma_X(i)}(X) \tag{4.3}$$

Za različite izbore vektora koeficijenata troškova Λ dobijamo različite tipove funkcije cilja koji odgovaraju određenim diskretnim lokacijskim problemima iz grupe problema koje DOMP uopštava. Na primer, uzimajući $\Lambda = (1,1,..,1),$ DOMP se svodi na problem p-medijane, za $\Lambda = (0,0,..,1)$ dobijamo p-centar, a $\Lambda = (\mu,\mu,..,\mu,1),\ 0 < \mu < 1,$ daje μ-centdian problem (sa konveksnom kombinacijom median i centar funkcija cilja). Ako je vektor Λ oblika $\Lambda = (0,...,0,1,..,1),$ gde su prvih n-k koeficijenata nule, a poslednjih k jedinice, dobijamo problem k-centra, čiji je cilj minimizacija prosečnih troškova transporta za snabdevanje k "najskupljih" klijenata. I drugi izbori vektora Λ daju probleme iz klase DOMP-a sa značajnom praktičnom primenom. Na primer, vektor $\Lambda = (0,...,0,1,..,1,0,...,0),$ gde su prvih k_1 i poslednjih k_2 koeficijenata jednaki nuli, a središnjih n- k_1- k_2 jednaki jedinici, vodi problemu pod nazivom (k_1+ k_2)-uređena sredina ((k_1+ k_2)-trimmed mean problem), kod kojeg se k_1 najnižih i k_2 najviših transportnih troškova zanemaruju i minimizira srednja vrednost središnjih (po vrednosti) troškova. Uzimajući $\Lambda = (1,...,1,0,..,0,1,...,1),$ sa prvih k_1 vrednosti 1, sledećih n- k_1- k_2 vrednosti 0 i poslednjih k_2 koeficijenata 1, dobijamo problem minimizacije sume k_1 najnižih i k_2 najviših transportnih troškova. Cilj ovog problema iz DOMP klase je uspostavljanje p lokacija snabdevača tako da prosečna cena snabdevanja onih klijenata koji su jako blizu ili jako daleko bude minimalna. Ako postavimo $\Lambda = (2,0,...,0,1)$ DOMP se svodi na problem minimizacije sume najveće i najmanje (pomnožene sa 2) cene transporta, ignorišući sve ostale, itd.

Iz navedenih primera se vidi da se mnogi klasični, ali i neki novi diskretni lokacijski problemi mogu lako izvesti iz DOMP-a modifikujući njegovu funkciju cilja. U [Nic01] dat je primer koji ilustruje veliki uticaj funkcije cilja na optimalnu lokaciju novih snabdevača.

DOMP spada u klasu NP-teških problema, jer predstavlja uopštenje problema p-medijane, čija je NP-kompletnost dokazana u [KH79].

4.1.1 Primer

Razmotrimo DOMP sa n=4, p=2 i matricom transportnih troškova:

$$C = \begin{bmatrix} 0 & 10 & 7 & 8 \\ 10 & 0 & 15 & 18 \\ 7 & 15 & 0 & 12 \\ 8 & 18 & 12 & 0 \end{bmatrix}.$$

Demonstrirajmo način računanja vrednosti funkcije cilja ako uzmemo $\Lambda = (1,1,1,1)$ za vektor koeficijenata troškova (u pitanju je problem p-medijane). Optimalno rešenje se dobija uspostavljanjem snabdevača na lokacijama 1 i 2, tj. X={1, 2}. Očigledno je da se korisnici sa lokacije 2 snabdevaju sa iste, a korisnici na lokacijama 1, 3 i 4 sa lokacije 1. Pridruženi vektor troškova, uređen u neopadajućem poretku je $C_{\leq} = (0,0,7,8).$ Optimalna vrednost funkcije cilja je jednaka 1x0 + 1x0 + 1x7 + 1x8 =15.

4.2 Postojeći načini rešavanja

U [Nic01] pored dve formulacije DOMP-a ne nalazimo bilo kakve predloge o načinu njegovog rešavanja, niti pokušaje sagledavanja efikasnosti pristupa celobrojnog programiranja za ovaj problem. Za planarne i mrežne lokacijske probleme odgovarajuće modele slične DOMP-u proučavali su: Puerto i Fernandez u [PF95] i [PF00], Nickel i Puerto [NP99], Francis i sar. [Fra00], Rodrìguez-Chia i sar. [Rod00], Kalesis i sar. [Kal03].

U radu [Dom03] za rešavanje DOMP-a su predložene dva heuristička pristupa. Prvi metod je zasnovan na VNS metaheuristici iz [HM97] za problem p-medijane. VNS je poznati pristup koji se često koristi za rešavanje diskretnih lokacijskih problema i obično daje rešenja visokog kvaliteta (videti [HM01a], [HM01b] i [HM01c]). Osnovna ideja VNS-a je da se u okviru algoritma lokalne pretrage (local search) sistematski vrši zamena okolina u kojima se traži rešenje. Pretraživanje okolina se može vršiti na dva načina: one koje su bliže tekućem rešenju - manje okoline se postupno istražuju sve dok se nađe bolje rešenje, dok se veće okoline- udaljenije od tekućeg rešenja parcijalno istražuju slučajnim nalaženjem nekog rešenja iz jedne od njih, gde se zatim počinje sa lokalnom pretragom. Algoritam zadržava tekuće rešenje sve dok ga ne poboljša i tada "skoči" u manju okolinu novog rešenja i nastavlja njenu pretragu na prvi način. Pretraživanje okolina se reguliše skupom parametara i najpre se ispituju okoline najbliže tekućem rešenju, zatim one dalje itd. U VNS metod opisan u [Dom03] implementirane su proždrljiva heuristika (greedy heuristics) za nalaženje početnog rešenja i modifikovana heuristika brze zamene (Modified Fast Interchange Heuristics) sa procedurama Modified Move i Modified Update.

Druga heuristika predložena u [Dom03] pod nazivom "evolutivni program" (Evolutionary Program-EP) je proširenje genetskog algoritma iz [MV96]. EP koristi celobrojno kodiranje jedinki vektorom dužine n, pri čemu genetski kod svake jedinke sadrži indekse uspostavljenih lokacija snabdevača sortirane u rastućem poretku. Jedan deo početne populacije generiše se na slučajan način, dok drugi deo čine rešenja dobijena proždrljivom heuristikom. Jedinke u populaciji se rangiraju prema

funkciji prilagođenosti i za stvaranje nove generacije uzima se k najboljih. Operator ukrštanja upoređuje par jedinki-roditelja i markira one gene koji su prisutni u oba genetska koda, a neoznačeni geni se sortiraju u rastući niz i pomeraju ulevo. Ovako transformisani genetski kodovi roditelja razmenjuju sve gene desno od unapred izabrane tačke ukrštanja. Na kraju, dobijene kodove jedinki-potomaka treba ponovo sortirati u skladu sa primenjenim načinom kodiranja. Operator mutacije se realizuje zamenom jednog indeksa lokacije iz genetskog koda jedinke koja se mutira nekim indeksom koji se ne pojavljuje u ostatku koda. Nakon ukrštanja i mutacije, iz skupa svih prisutnih jedinki u populaciji operator selekcije bira k najboljih, tako da je u svakoj iteraciji EP-a veličina populacije konstantna. Iterativni proces se zaustavlja posle izvesnog broja generacija i algoritam uzima najbolju jedinku iz poslednje generacije za približno rešenje problema. Za instance DOMP manjih dimenzija (sem u slučaju problema p-centra), kao i za veće instance p-medijane, obe metaheuristike iz [Dom03] efikasno daju rešenja zadovoljavajućeg kvaliteta. Na svim većim test-instancama DOMP-a predložene heuristike imaju loše performanse.

Boland i saradnici u [Bol03] opisuju egzaktan branch-and-bound metod (BnB) za rešavanje DOMP-a. Autori dokazuju niz osobina optimalnog rešenja koje im omogućavaju da "ojačaju" obe predložene ILP formulacije DOMP-a, koristeći dodatna ograničenja ili pre-procesiranje problema (tačnije, fiksiranje vrednosti nekih promenljivih). Branch-and-bound metod je razvijen u skladu sa specifičnom strukturom DOMP-a i koristi donje granice problema koje se relativno brzo računaju, gornje granice dobijene VNS heuristikom [Dom03] i dva različita načina grananja (max-regret i simple index-order metod). U radu je definisano 8 klasa DOMP-a prema odgovarajućim funkcijama cilja, generisane instance za svaku od klasa problema ($n \leq 30$, $p \leq 16$) i na njima poređene predložene ILP formulacije i različiti načini grananja BnB metoda. Eksperimentalni rezultati pokazuju BnB metod koji koristi formulaciju sa $O(n^3)$ promenljivih i $O(n^3)$ ograničenja i max-regret grananje daje optimalna rešenja u najkraćem procesorskom vremenu na svim testiranim instancama, sem u slučaju instanci iz klase p-centar problema.

Rezultati i poređenja postojećih metoda za rešavanje DOMP-a sa genetskim algoritmom predloženim u ovom radu detaljnije su opisani u sekciji 4.4.

4.3 Hibridni genetski algoritam (HGA)

4.3.1 Reprezentacija i funkcija prilagođenosti

Za rešavanje DOMP-a takođe je izabrano binarno kodiranje jedinki. Svaki bit u genetskom kodu rešenja, čija je vrednost 1 označava da je na datoj lokaciji uspostavljen snabdevač, a vrednost 0 da nije. Pošto za sve lokacije ne postoji ograničenje kapaciteta i pošto iz genetskog koda dobijamo indekse lokacija postavljenih snabdevača, svaki korisnik može da izabere sebi najbližeg snabdevača.

Predložena GA implementacija računa vrednosnu funkciju na dva različita načina, u zavisnosti od broja snabdevača koje treba locirati-p i vrednosti $e_0 = c * \sqrt{n}$. Vrednost konstante c=0.95 je eksperimentalno određena. Za svakog korisnika GA pravi listu indeksa potencijalnih lokacija snabdevača, uređenu u neopadajućem poretku po transportnim troškovima. Ove liste se formiraju pri inicijalizaciji programa pomoću ugrađene funkcije qsort(). Korišćenje lista zahteva dodatni memorijski prostor (oko 50% više memorije), ali je računanje vrednosne fukcije nekoliko puta brže.

Ako je p dovoljno veliko (p > e_0) koristimo indeksiranu listu potencijalnih lokacija snabdevača. Za svakog korisnika algoritam pretražuje odgovarajuću listu indeksa do prve pojave uspostavljenog snabdevača, koji se zatim pridružuje tekućem korisniku. Na ovaj način biramo najpovoljnije lokacije za snabdevanje korisnika jer su liste indeksa uređene u neopadajućem poretku. Pošto je p veliko, potrebno je samo nekoliko koraka za nalaženje prvog indeksa lociranog snabdevača u svakoj listi, pa je vreme izvršavanja vrednosne funkcije malo. Vremenska složenost u ovom slučaju je O(n*n/p), jer nalaženje odgovarajuće liste za tekućeg korisnika zahteva O(n) vremena, a njeno pretraživanje n/p koraka prosečno.

Za male vrednosti p (p ≤ e_0) gornja strategija je suviše spora, jer je potrebno mnogo više koraka za nalaženje prve uspostavljene lokacije snabdevača pri pretraživanju indeksiranih listi. U ovom slučaju koristimo drugu strukturu podataka-niz (o_j) koji sadrži samo indekse lociranih snabdevača:

o_j=i, ako je je snabdevač *i j*-ti po redu u nizu uspostavljenih snabdevača.

Niz (o_j) je relativno male dužine, jer je mali broj snabdevača koje treba locirati, tako da ga možemo brzo pretražiti. Pošto niz nije uređen u nekom poretku, moramo proći kroz sve njegove članove, ali zbog malog broja elemenata izvršavanje je efikasno. Niz se konstruiše samo jednom, na početku u O(n) vremenu, a koristi se n puta. Svako pretraživanje niza zahteva p koraka, tako da je ukupna vremenska složenost druge strategije O(n*p).

Ako je k=max{p, n/p}, tada je:

- u prvom slučaju (p > e_0) broj potrebnih operacija n*n/p ≤ n*k
- u drugom slučaju (p ≤ e_0) potrebno ne više od p koraka za svakog od n klijenata, pa je broj operacija n*p ≤ n*k .

Dakle, opisani algoritam za računanje vrednosne funkcije je složenosti O(n*k).

4.4 Karakteristike predloženog GA

Predloženi GA takođe koristi genetske operatore koji su opisani u 2.3. Primenjeni su isti parametri algoritma iz 2.3. koji su se najbolje pokazali u praksi. Takođe je korišćena ista politika zamene generacija, kao i keširanje. Algoritam je kodiran u programskom jeziku C i testiran na ORLIB instancama problema p-medijane, preuzetih iz [Bea96]. Svi testovi su izvedeni na računaru sa AMD K7/1.33 Ghz procesorom sa 256 MB memorije. Maksimalan broj generacija GA je N_{gen}=5000 za sve instance problema. Algoritam se takođe zaustavlja ako je najbolja jedinka ili najbolja vrednost funkcije cilja ostala nepromenjena tokom N_{rep}=2000 uzastopnih generacija. Zbog nedeterminističke prirode genetskog algoritma testiranje je izvršeno 10 puta na svakoj p-medijan instanci. Koristeći ove kriterijume GA dostiže zadovoljavajuća rešenja u razumnom vremenu izvršavanja, što se vidi iz rezultata datih u tabelama u sekciji 4.5.

4.4.1 Heuristika za poboljšanje rešenja

GA je hibridizovan sa jednom modifikacijom heuristike zamene (modified interchange heuristic) koja je razvijena u cilju poboljšanja kvaliteta rešenja. Heuristika se izvršava u svakoj generaciji algoritma, pre primene genetskih operatora selekcije, ukrštanja i mutacije. Ona se primenjuje samo na najbolju jedinku, ukoliko se ta jedinka promenila u odnosu na prethodnu generaciju. Heuristika je deterministička, tako da ako ne uspeva da popravi najbolju jedinku u tekućoj generaciji, ne može dati nikakvo dalje poboljšanje.

Heuristika najpre pokušava da isključi jednu uspostavljenu lokaciju iz najboljeg rešenja i da uključi drugu lokaciju snabdevača koji već nije uspostavljen. Isključivanje i uključivanje snabdevača se vrši do prvog poboljšanja, a kad ga dobije, ceo postupak se ponavlja na novoj, poboljšanoj jedinki. Proces se zaustavlja kada heuristika ne može više popraviti najbolju jedinku isključivanjem jedne i uključivanjem druge lokacije.

Izvršavanje heuristike se može ubrzati implementacijom keširanja. Svaki put kada pokušavamo sa zamenom lokacija, ispitujemo da li ta nova jedinka postoji u keš memoriji. Ako postoji, njenu vrednost ne računamo već je preuzimamo iz heš-tabele, u suprotnom, vrednost nove jedinke se skraćeno računa na dole opisani način i smešta u heš- tabelu odakle se dalje može po potrebi koristiti.

Neka je *i* indeks uspostavljene lokacije snabdevača kojeg izbacujemo iz tekućeg rešenja (jedinke), a *j* indeks lokacije snabdevača koji ubacujemo, a koji nije prethodno lociran u tekućoj najboljoj jedinki. Sa *imin* označimo indeks snabdevača koji je najbliži klijentu *cl*, a sa *d(i, j)* rastojanje između *i*-te i *j*-te lokacije u mreži. Heuristika za svakog klijenta *cl* ažurira *imin*, pri čemu razlikuje tri slučaja:

1. slučaj. Ako je $d(i,cl) \geq d(imin, cl)$ i $d(imin, cl) \geq d(j, cl)$ tada je $imin=j$,

2.slučaj. Ako uslov *1.slučaja* nije ispunjen, i ako je $i=imin$ i $d(i, cl)<d(j, cl)$ tada računamo vrednosnu funkciju za datog klijenta na isti način kao u 4.3.1. Ovaj slučaj je relativno redak u praksi. Na primer, ako je n=100, p=10, na svakog klijenta u proseku dolazi po 10 lokacija snabdevača. Ako smo izbacili lokaciju *i*, verovatnoća da je $i=imin$ je 1/10. Pošto je verovatnoća da je $i<j$ jednaka1/2, ako bi pretpostavili da je pridruživanje snabdevača klijentima slučajno, verovatnoća da se desi *slučaj 2.* bi bila 1/2 *1/10=1/20. Pošto pridruživanje nije slučajno, verovatnoća nije tačno 1/20, ali je dosta mala, što se vidi i u praksi, jer se rešenje dobija relativno brzo.

3.slučaj. Ako nisu zadovoljeni uslovi oba prethodna slučaja *imin* se ne menja.

Kada za svakog klijenta odredimo indeks lokacije njemu najbližeg snabdevača, potrebno je sortirati formirani niz lokacija pomoću funkcije *qsort()*. Lako se može pokazati da je vremenska složenost izvršavanja heuristike $O(n^2*p*k)$, gde je $k=\max\{p, n/p\}$. S obzirom da se složenost algoritma uvek računa u najlošijem slučaju (*slučaj 2*. koji se retko javlja), algoritam se u praksi brže izvršava nego što teoretska ocena pokazuje.

4.5 Rezultati i poređenja sa postojećim metodama

U literaturi je definisano 8 tipova DOMP problema, označenih sa *T1-T8* ([Bol03]), a koji su klasifikovani prema vektoru koeficijenata transportnih troškova Λ:

- *T1 tip* se dobija za vektor $\Lambda = (1,1,..,1),$ a odgovara problemu p-medijane,
- *T2 tip* se dobija za $\Lambda = (0,0,..,1),$ a odgovara problemu p-centra,
- *T3 tip* se dobija za $\Lambda = (0,...,0,1,..,1),$ gde su prvih n-k koeficijenata nule, a poslednjih k jedinice, a odgovara problemu k-centra,
- *T4 tip* se dobija za $\Lambda = (0,...,0,1,..,1,0,...,0),$ gde su prvih k_1 i poslednjih k_2 koeficijenata jednaki nuli, a središnjih $n-k_1-k_2$ jednaki jedinici, a odgovara problemu $(k_1+ k_2)$-uređene sredine,
- *T5 tip* se dobija za vektor Λ sa binarnim alternirajućim koeficijentima, tj. $\Lambda = (0,1,0,1,...,0,1,0,1),$ ako je n parno, a $\Lambda = (1,0,1,...,0,1,0,1),$ ako je n neparno,
- *T6 tip* se dobija za vektor Λ sa binarnim alternirajućim koeficijentima, pri čemu je poslednji koeficijent 0, tj. $\Lambda = (1,0,1,0,...,1,0,1,0),$ ako je n parno, a $\Lambda = (0,1,0,...,1,0,1,0),$ ako je n neparno,
- *T7 tip* se dobija za vektor Λ koji nastaje ponavljanjem niza 0,1,1 počevši od kraja do početka vektora, tj. $\Lambda = (...,0,1,1,0,1,1),$
- *T8 tip* se dobija za vektor Λ koji nastaje ponavljanjem niza 0,0,1 počevši od kraja do početka vektora, tj. $\Lambda = (...,0,0,1,0,0,1).$

U [Bol03] testirane su DOMP instance malih dimenzija ($n \le 30$, $p \le 16$) za svaki od tipova *T1-T8*, ali pošto te instance nisu javno dostupne, nije bilo moguće na njima testirati opisani hibridni GA. U radu [Dom03] testiranje je obavljeno i na ORLIB instancama p-medijane, koje su višestruko većih dimenzija ($100 \le n \le 900$, $5 \le p \le 200$). Imajući to u vidu, testiranje je izvršeno samo na ovim instancama, preuzetim iz [Bea96].

Rezultati hibridnog GA za DOMP probleme *T1-T8* su dati redom u Tabelama 4.1, 4.3 - 4.5, 4.7 - 4.10 na isti način kao u prethodnim poglavljima, a u Tabelama 4.2 i 4.6 poređenja sa EP i VNS metodama. Za probleme *T1* (p-medijane) i *T2* (p-centra) su poznata optimalna rešenja i ona su data u trećoj koloni sa oznakom *opt.* Za probleme *T3-T8* nisu poznata optimalna rešenja u literaturi, tako da su u tabelama 4.4, 4.5 i 4.7 - 4.10 prikazana samo najbolja rešenja dobijena hibridinim GA. U Tabelama 4.1 i 4.3 kolona od_{najb} se odnosi na odstupanje najboljeg rešenja HGA od optimalnog, a kolone od_{sr} i σ_{sr} na srednje odstupanje i devijaciju svakog od 10 rešenja od optimalnog.

Kao što se može videti iz Tabela 4.1 i 4.2, predloženi hibridni genetski algoritam dostiže optimalna rešenja na više od polovine testiranih instanci (22 od 40), za razliku od obe metode opisane u [Dom03]. Evolutivni program (EP) je dostigao 12, a VNS heuristika 17 optimalnih rešenja od 40 testiranih instanci. Imajući u vidu da su testiranja u [Dom03] izvedena na nešto sporijem računaru (Pentium III/800 MHz sa 1GB RAM), može se zaključiti da su vremena izvršavanja hibridnog GA uglavnom ista ili kraća.

Na primer, na instanci *pmed40* EP dostiže vrednost 5188, VNS 5141 a HGA 5134, dok je optimalna vrednost 5128. Vreme izvršavanja EP-a na ovoj instanci je 1492 PIII-sekundi, VNS-a 4774 PIII-sekundi a HGA 766 AMD-sekundi, pa zaključujemo da je HGA brži od EP-a i 3-4 puta brži od VNS-a na *pmed40.* Slične zaključke možemo izvesti i na većini ostalih instanci većih dimenzija.

Poredeći odstupanja najboljih rešenja ove tri metode od optimalnih za date instance, vidi se da EP ima maksimalno ostupanje (gap) 3.39% na instanci *pmed25* a VNS 1.05% na instanci *pmed22.* HGA ima maksimalno odstupanje najboljeg rešenja 0.654% na instanci *pmed30.* Na *pmed30* i EP i VNS imaju veće odstupanje od HGA (2.36% i 1.01% respektivno).

Tabela 4.1 Rezultati HGA za T1 (p-median)

instanca	n	p	Opt.	GA_{najb}	od_{najb}[%]	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	5819	5819	0.000	0.181	2.919	2062	0.000	0.000	37923	63.3
pmed2	100	10	4093	4093	0.000	0.203	3.379	2039	0.000	0.000	49215	51.8
pmed3	100	10	4250	4250	0.000	0.241	3.384	2058	0.000	0.000	46962	54.4
pmed4	100	20	3034	3034	0.000	1.377	4.715	2643	0.079	0.167	65599	50.4
pmed5	100	33	1355	1355	0.000	1.886	5.722	2812	0.000	0.000	71418	49.3
pmed6	200	5	7824	7824	0.000	0.591	8.615	2017	0.000	0.000	47448	53.0
pmed7	200	10	5631	5631	0.000	5.168	15.590	2863	0.043	0.069	77066	46.2
pmed8	200	20	4445	4445	0.000	3.696	11.605	2497	0.020	0.064	70512	43.6
pmed9	200	40	2734	2740	0.219	3.932	14.432	2365	0.646	0.150	68565	42.1
pmed10	200	67	1255	1256	0.080	7.949	21.763	2504	0.773	0.346	71868	42.6
pmed11	300	5	7696	7696	0.000	0.723	15.221	2001	0.000	0.000	49622	50.5
pmed12	300	10	6634	6634	0.000	3.428	22.852	2192	0.000	0.000	61015	44.4
pmed13	300	30	4374	4381	0.160	8.772	22.393	2597	0.206	0.075	76926	40.8
pmed14	300	60	2968	2969	0.034	16.046	34.192	2655	0.128	0.074	78913	40.6
pmed15	300	100	1729	1735	0.347	36.211	56.106	3108	0.526	0.145	92973	40.3
pmed16	400	5	8162	8162	0.000	5.541	28.152	2288	0.000	0.000	58954	48.6
pmed17	400	10	6999	6999	0.000	12.821	47.575	2522	0.026	0.048	71531	43.4
pmed18	400	40	4809	4809	0.000	23.049	44.661	2547	0.029	0.026	76689	39.9
pmed19	400	80	2845	2848	0.105	44.280	79.621	2560	0.252	0.098	77988	39.2
pmed20	400	133	1789	1794	0.279	76.173	138.629	2873	0.446	0.134	89264	37.8
pmed21	500	5	9138	9138	0.000	2.323	34.118	2001	0.000	0.000	52161	47.9
pmed22	500	10	8579	8579	0.000	13.491	64.166	2171	0.193	0.408	63150	41.9
pmed23	500	50	4619	4624	0.108	47.592	75.542	2493	0.151	0.056	75890	39.3
pmed24	500	100	2961	2966	0.169	89.951	192.806	2257	0.284	0.084	71143	37.1
pmed25	500	167	1828	1835	0.383	189.711	254.447	2669	0.525	0.074	84005	37.2
pmed26	600	5	9917	9917	0.000	9.865	52.265	2169	0.000	0.000	57795	46.8
pmed27	600	10	8307	8307	0.000	18.245	87.084	2218	0.000	0.000	64566	41.9
pmed28	600	60	4498	4500	0.044	102.494	161.932	2815	0.195	0.064	87136	38.2
pmed29	600	120	3033	3036	0.099	179.188	273.325	2963	0.218	0.078	93437	37.0
pmed30	600	200	1989	2002	0.654	492.092	651.279	3455	0.979	0.257	110826	36.0
pmed31	700	5	10086	10086	0.000	6.743	60.908	2001	0.000	0.000	54454	45.7
pmed32	700	10	9297	9297	0.000	38.093	122.546	2386	0.017	0.022	70071	41.4
pmed33	700	70	4700	4719	0.404	153.248	206.433	2295	0.419	0.010	71310	38.0
pmed34	700	140	3013	3023	0.332	310.593	485.249	2567	0.547	0.126	81823	36.3
pmed35	800	5	10400	10400	0.000	7.644	74.310	2001	0.000	0.000	53903	46.2
pmed36	800	10	9934	9951	0.171	33.488	133.443	2001	0.171	0.000	57148	43.0
pmed37	800	80	5057	5063	0.119	183.140	242.446	2001	0.119	0.000	62145	38.0
pmed38	900	5	11060	11060	0.000	29.468	117.790	2405	0.000	0.000	66513	44.8
pmed39	900	10	9423	9423	0.000	38.886	164.910	2001	0.000	0.000	59057	41.1
pmed40	900	90	5128	5134	0.117	583.392	766.518	3183	0.212	0.055	100220	37.1

Tabela 4.2 Poređenje HGA sa EP i VNS za T1 (p-median)

				HGA			EP			VNS		
instanca	n	p	Opt.	$GA_{najb.}$	od_{najb}[%]	t_{AMD}[s]	$EP_{najb.}$	od[%]	t_{PIII}[s]	$VNS_{najb.}$	od[%]	t_{PIII}[s]
pmed1	100	5	5819	5819	0.000	2.919	5819	0.00	25.42	5819	0.00	1.19
pmed2	100	10	4093	4093	0.000	3.379	4093	0.00	37.55	4093	0.00	2.97
pmed3	100	10	4250	4250	0.000	3.384	4250	0.00	37.88	4250	0.00	3.00
pmed4	100	20	3034	3034	0.000	4.715	3046	0.40	61.48	3046	0.40	5.98
pmed5	100	33	1355	1355	0.000	5.722	1361	0.44	93.22	1358	0.22	6.81
pmed6	200	5	7824	7824	0.000	8.615	7824	0.00	36.25	7824	0.00	7.95
pmed7	200	10	5631	5631	0.000	15.590	5645	0.25	55.39	5639	0.14	12.72
pmed8	200	20	4445	4445	0.000	11.605	4465	0.45	91.81	4457	0.27	21.05
pmed9	200	40	2734	2740	0.219	14.432	2762	1.02	170.25	2753	0.69	41.98
pmed10	200	67	1255	1256	0.080	21.763	1277	1.75	290.53	1259	0.32	72.22
pmed11	300	5	7696	7696	0.000	15.221	7696	0.00	47.98	7696	0.00	12.52
pmed12	300	10	6634	6634	0.000	22.852	6634	0.00	75.63	6634	0.00	26.02
pmed13	300	30	4374	4381	0.160	22.393	4432	1.33	193.22	4374	0.00	87.92
pmed14	300	60	2968	2969	0.034	34.192	2997	0.98	359.58	2969	0.03	241.95
pmed15	300	100	1729	1735	0.347	56.106	1749	1.16	580.98	1739	0.58	363.39
pmed16	400	5	8162	8162	0.000	28.152	8183	0.26	56.89	8162	0.00	24.36
pmed17	400	10	6999	6999	0.000	47.575	6999	0.00	95.08	6999	0.00	47.30
pmed18	400	40	4809	4809	0.000	44.661	4880	1.48	320.38	4811	0.04	275.69
pmed19	400	80	2845	2848	0.105	79.621	2891	1.62	604.36	2864	0.67	469.30
pmed20	400	133	1789	1794	0.279	138.629	1832	2.40	963.44	1790	0.06	915.17
pmed21	500	5	9138	9138	0.000	34.118	9138	0.00	70.14	9138	0.00	27.39
pmed22	500	10	8579	8579	0.000	64.166	8669	1.05	116.59	8669	1.05	64.25
pmed23	500	50	4619	4624	0.108	75.542	4651	0.69	486.08	4619	0.00	443.23
pmed24	500	100	2961	2966	0.169	192.806	3009	1.62	924.66	2967	0.20	1382.84
pmed25	500	167	1828	1835	0.383	254.447	1890	3.39	1484.13	1841	0.71	2297.25
pmed26	600	5	9917	9917	0.000	52.265	9919	0.02	84.34	9917	0.00	48.45
pmed27	600	10	8307	8307	0.000	87.084	8330	0.28	136.53	8310	0.04	127.63
pmed28	600	60	4498	4500	0.044	161.932	4573	1.67	673.30	4508	0.22	965.48
pmed29	600	120	3033	3036	0.099	273.325	3099	2.18	1268.89	3036	0.10	2758.56
pmed30	600	200	1989	2002	0.654	651.279	2036	2.36	2403.33	2009	1.01	3002.34
pmed31	700	5	10086	10086	0.000	60.908	10086	0.00	92.67	10086	0.00	56.02
pmed32	700	10	9297	9297	0.000	122.546	9319	0.24	156.50	9301	0.04	165.27
pmed33	700	70	4700	4719	0.404	206.433	4781	1.72	894.19	4705	0.11	2311.03
pmed34	700	140	3013	3023	0.332	485.249	3100	2.89	1762.69	3024	0.37	5384.19
pmed35	800	5	10400	10400	0.000	74.310	10400	0.00	109.86	10400	0.00	88.50
pmed36	800	10	9934	9951	0.171	133.443	9947	0.13	182.06	9934	0.00	200.97
pmed37	800	80	5057	5063	0.119	242.446	5126	1.36	1190.25	5066	0.18	2830.30
pmed38	900	5	11060	11060	0.000	117.790	11060	0.00	120.14	11060	0.00	150.53
pmed39	900	10	9423	9423	0.000	164.910	9423	0.00	207.75	9423	0.00	200.73
pmed40	900	90	5128	5134	0.117	766.518	5188	1.17	1492.59	5141	0.25	4774.38

Iz Tabele 4.3 se vidi da HGA dostiže optimalna rešenja na 9 od ukupno 40 instanci. Procentualna odstupanja $GA_{najb.}$ od optimalnog rešenja su velika jer su vrednosti rešenja dosta manja u odnosu na vrednosti problema *T1*. Problem *T2* je teži od *T1*, tako da je kvalitet rešenja nešto lošiji i vreme izvršavanja HGA nešto duže. U literaturi ne nalazimo rešenja problema *T2* na ovim instancama DOMP-a, pa izostaju poređenja sa ostalim metodama za rešavanje DOMP-a.

Tabela 4.3 Rezultati T2 (p-centar)

instanca	n	p	Opt.	GA_{najb}	od_{najb}[%]	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	127	127	0.00	0.483	3.662	2214	0.000	0.000	47589	57.1
pmed2	100	10	98	100	2.04	0.875	4.093	2390	3.140	0.994	54543	54.4
pmed3	100	10	93	94	1.08	0.863	4.198	2319	3.420	2.738	53068	54.3
pmed4	100	20	74	76	2.70	2.967	6.325	2854	8.753	4.608	71096	50.1
pmed5	100	33	48	48	0.00	4.904	8.908	3139	7.083	6.747	82763	46.9
pmed6	200	5	84	84	0.00	3.839	12.347	2763	0.833	1.129	68747	50.2
pmed7	200	10	64	66	3.13	3.954	15.787	2322	4.797	1.507	63635	45.3
pmed8	200	20	55	57	3.64	13.346	23.439	2944	12.412	7.443	86673	41.4
pmed9	200	40	37	42	13.51	27.164	42.140	2863	29.462	11.170	91605	36.2
pmed10	200	67	20	28	40.00	19.479	37.676	2603	65.357	15.834	80760	38.3
pmed11	300	5	59	60	1.69	0.942	17.766	2017	1.690	0.000	50034	50.5
pmed12	300	10	51	52	1.96	14.211	39.024	2534	2.537	0.929	73923	41.8
pmed13	300	30	35	41	17.14	36.445	63.444	2711	24.701	5.568	92269	31.9
pmed14	300	60	26	36	38.46	42.870	80.499	2412	53.460	5.270	80663	33.3
pmed15	300	100	18	27	50.00	30.084	80.178	2183	75.556	11.771	71598	34.7
pmed16	400	5	47	47	0.00	3.512	30.682	2048	0.000	0.000	54166	47.2
pmed17	400	10	39	41	5.13	17.683	62.733	2369	6.106	2.356	73418	38.2
pmed18	400	40	28	34	21.43	72.887	129.978	2544	33.195	7.594	90735	28.7
pmed19	400	80	19	29	52.63	55.673	143.668	2125	57.458	4.362	72505	32.0
pmed20	400	133	14	26	85.71	67.760	213.274	2348	104.941	7.692	79864	32.4
pmed21	500	5	40	40	0.00	6.227	45.503	2182	0.000	0.000	59207	45.9
pmed22	500	10	38	39	2.63	46.866	110.660	2663	6.989	6.515	85732	35.6
pmed23	500	50	23	30	30.43	152.940	268.599	2468	34.430	4.098	90608	26.7
pmed24	500	100	16	23	43.75	305.790	500.467	3024	54.620	9.668	109678	27.6
pmed25	500	167	12	22	83.33	271.913	523.732	2173	86.512	2.196	76954	29.4
pmed26	600	5	38	38	0.00	17.261	67.059	2426	1.842	1.776	69016	43.4
pmed27	600	10	32	34	6.25	31.258	134.310	2172	8.603	1.240	72702	33.1
pmed28	600	60	19	25	31.58	187.796	387.670	2777	36.380	2.530	102488	26.4
pmed29	600	120	13	22	69.23	365.883	691.577	2384	75.594	3.833	87820	26.8
pmed30	600	200	11	20	81.82	584.734	1217.801	2233	91.320	7.976	81909	26.8
pmed31	700	5	30	30	0.00	33.835	102.694	2502	0.000	0.000	71934	42.7
pmed32	700	10	29	30	3.45	85.350	212.964	2339	8.450	2.357	78379	33.1
pmed33	700	70	16	22	37.50	299.493	648.209	2111	42.955	3.586	80194	24.2
pmed34	700	140	12	20	66.67	1043.917	1658.282	2648	78.170	10.014	100506	24.2
pmed35	800	5	30	30	0.00	29.828	116.184	2399	2.667	1.405	70049	41.7
pmed36	800	10	27	29	7.41	62.860	239.267	2257	11.203	1.957	76559	32.2
pmed37	800	80	16	23	43.75	375.635	927.441	2099	45.054	2.100	78538	25.3
pmed38	900	5	29	29	0.00	37.659	158.255	2217	0.000	0.000	61043	45.1
pmed39	900	10	23	25	8.70	110.324	336.564	2285	9.900	3.795	78413	31.6
pmed40	900	90	14	20	42.86	808.591	1621.151	2278	47.860	2.357	87057	23.7

U koloni $GA_{najb.}$ tabela 4.4, 4.5 i 4.7 - 4.10 data su samo najbolja rešenja HGA za probleme *T3-T8* respektivno na p-medijan instancama, pošto odgovarajuća optimalna rešenja nisu poznata. Kolone od_{sr}*[%]* i σ_{sr}*[%]* odnose se na srednje odstupanje i devijaciju (u procentima) svakog od 10 rešenja od najboljeg rešenja dobijenog hibridnim genetskim algoritmom. Prosečno vreme potrebno algoritmu da dobije najbolju vrednost je dato u *t[s]* koloni, t_{tot}*[s]* predstavlja ukupno vreme rada HGA za svih 5000 generacija, dok je *gen* prosečan broj generacija za koji se dostiže najbolje rešenje.

Tabela 4.4 Rezultati T3

instanca	n	p	GA_{najb}	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	3148	0.178	3.010	2047	0.000	0.000	39426	61.5
pmed2	100	10	2444	1.100	4.273	2603	0.037	0.059	61834	52.5
pmed3	100	10	2452	0.461	3.758	2194	0.000	0.000	51391	53.2
pmed4	100	20	1959	1.091	4.345	2550	0.342	0.303	64001	49.9
pmed5	100	33	1072	0.705	4.196	2169	0.000	0.000	54554	49.8
pmed6	200	5	4163	0.251	7.983	2001	0.000	0.000	47167	52.9
pmed7	200	10	3157	7.724	17.485	3134	0.181	0.125	86614	44.8
pmed8	200	20	2630	4.333	12.303	2677	0.183	0.222	76248	43.1
pmed9	200	40	1854	11.119	19.345	3557	0.502	0.253	101476	43.0
pmed10	200	67	933	10.656	20.384	3190	0.890	0.691	91107	43.1
pmed11	300	5	4115	0.830	16.032	2001	0.000	0.000	51272	48.8
pmed12	300	10	3613	8.878	28.423	2681	0.014	0.044	75306	43.9
pmed13	300	30	2705	7.252	21.890	2396	0.222	0.118	72013	40.0
pmed14	300	60	1939	20.931	34.184	2981	0.413	0.346	88729	40.5
pmed15	300	100	1290	31.790	49.331	3099	0.519	0.354	93155	39.9
pmed16	400	5	4220	1.318	25.502	2001	0.000	0.000	52182	47.9
pmed17	400	10	3746	6.110	42.537	2132	0.000	0.000	60549	43.3
pmed18	400	40	2872	21.466	45.296	2651	0.265	0.251	80762	39.1
pmed19	400	80	1828	30.477	56.336	2204	0.131	0.074	66482	39.8
pmed20	400	133	1311	106.696	174.229	3379	0.686	0.350	104483	38.2
pmed21	500	5	4612	15.459	47.821	2822	0.007	0.021	74666	47.2
pmed22	500	10	4466	27.080	79.248	2681	0.000	0.000	78889	41.2
pmed23	500	50	2790	50.980	78.994	2950	0.237	0.165	90719	38.6
pmed24	500	100	1903	115.255	169.743	3143	0.268	0.152	98639	37.3
pmed25	500	167	1334	211.125	296.925	3168	0.990	0.456	99484	37.3
pmed26	600	5	4975	5.880	48.129	2001	0.000	0.000	53589	46.5
pmed27	600	10	4389	48.565	119.598	3068	0.052	0.082	92153	40.0
pmed28	600	60	2689	109.703	173.095	2929	0.390	0.249	91762	37.4
pmed29	600	120	1962	290.937	426.372	2859	0.255	0.115	90027	37.1
pmed30	600	200	1408	362.299	524.296	2769	0.142	0.058	88727	36.0
pmed31	700	5	4923	11.163	65.655	2001	0.000	0.000	54383	45.7
pmed32	700	10	4772	47.641	134.242	2594	0.052	0.085	77867	40.1
pmed33	700	70	2782	169.064	280.644	2706	0.270	0.204	85884	36.7
pmed34	700	140	1970	390.486	587.720	2859	0.426	0.209	91212	36.2
pmed35	800	5	5089	7.523	74.712	2001	0.000	0.000	54763	45.3
pmed36	800	10	5009	52.375	158.062	2345	0.000	0.000	71321	39.3
pmed37	800	80	3004	349.732	430.369	3310	0.093	0.101	104767	36.8
pmed38	900	5	5364	39.260	127.369	2467	0.004	0.012	66676	46.1
pmed39	900	10	4772	33.497	160.625	2001	0.000	0.000	60138	40.0
pmed40	900	90	2993	459.065	597.992	2675	0.050	0.081	84168	37.1

Tabela 4.5 Rezultati HGA za T4 (problem (k_1+ k_2)-uređene sredine)

instanca	n	p	GA_{najb}	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	4523	0.064	2.870	2001	0.000	0.000	37527	62.5
pmed2	100	10	2987	0.198	3.385	2033	0.000	0.000	49356	51.5
pmed3	100	10	3067	0.138	3.351	2001	0.000	0.000	46422	53.7
pmed4	100	20	2137	1.780	5.128	2897	0.080	0.110	72337	50.2
pmed5	100	33	818	0.884	4.837	2360	0.000	0.000	65865	44.3
pmed6	200	5	6064	0.634	8.754	2023	0.000	0.000	48154	52.5
pmed7	200	10	4206	1.547	11.766	2167	0.000	0.000	56941	47.5
pmed8	200	20	3182	3.529	12.202	2468	0.006	0.020	68714	44.4
pmed9	200	40	1807	7.454	18.481	2790	0.111	0.233	80860	42.0
pmed10	200	67	818	9.853	27.905	2518	0.672	0.284	76554	39.3
pmed11	300	5	5979	0.921	15.444	2001	0.000	0.000	49284	50.8
pmed12	300	10	5021	1.631	21.057	2001	0.000	0.000	55075	45.0
pmed13	300	30	3133	7.674	22.712	2329	0.192	0.170	68360	41.4
pmed14	300	60	1949	30.393	50.271	3468	0.349	0.250	103267	40.5
pmed15	300	100	1134	40.323	83.866	2573	0.256	0.128	83555	35.0
pmed16	400	5	6341	1.476	24.237	2001	0.000	0.000	50168	49.9
pmed17	400	10	5381	3.385	37.933	2001	0.000	0.000	57218	42.9
pmed18	400	40	3437	33.363	54.822	3412	0.035	0.027	102298	40.1
pmed19	400	80	1926	38.269	63.999	2930	0.348	0.303	89385	39.1
pmed20	400	133	1146	81.173	201.458	2393	0.489	0.215	79466	33.6
pmed21	500	5	7245	1.908	33.997	2001	0.000	0.000	52764	47.3
pmed22	500	10	6685	12.651	63.664	2199	0.000	0.000	64354	41.6
pmed23	500	50	3307	54.653	82.941	3047	0.088	0.115	93742	38.6
pmed24	500	100	2005	68.610	131.277	2169	0.000	0.000	68133	37.3
pmed25	500	167	1149	281.596	537.011	2773	0.191	0.200	94382	31.9
pmed26	600	5	7787	4.308	46.058	2001	0.000	0.000	52923	47.2
pmed27	600	10	6444	16.044	85.349	2184	0.000	0.000	64004	41.5
pmed28	600	60	3202	85.197	146.599	2146	0.000	0.000	66419	38.2
pmed29	600	120	2006	131.286	205.499	2462	0.040	0.032	77613	37.0
pmed30	600	200	1295	381.607	759.281	2910	0.216	0.108	100737	30.8
pmed31	700	5	8046	17.285	71.425	2244	0.000	0.000	60651	46.0
pmed32	700	10	7278	25.123	111.564	2334	0.027	0.032	70536	39.6
pmed33	700	70	3405	142.849	216.540	2576	0.112	0.098	80436	37.6
pmed34	700	140	2033	423.547	681.274	2449	0.064	0.052	78238	36.3
pmed35	800	5	8191	15.995	83.567	2128	0.000	0.000	57646	45.9
pmed36	800	10	7796	96.724	198.654	2981	0.049	0.103	87822	41.1
pmed37	800	80	3600	205.158	358.066	2315	0.000	0.000	72422	37.5
pmed38	900	5	8720	19.594	102.448	2001	0.000	0.000	53482	46.6
pmed39	900	10	7360	42.451	170.158	2084	0.000	0.000	61372	41.2
pmed40	900	90	3710	441.484	693.154	2252	0.003	0.009	70509	37.5

U [Dom03] dati su rezultati EP i VNS pristupa za problem (k_1+ k_2)-uređene sredine, tako da je moguće porediti najbolja rešenja ovih heuristika sa najboljim rešenjima HGA na p-medijan instancama. U prvoj koloni Tabele 4.6 nalaze se dimenzije rešavanih instanci (n i p), a naredne kolone sadrže najbolja rešenja i ukupno vreme izvršavanja HGA, EP i VNS metoda respektivno. Najmanje poznate vrednosti za svaku instancu su podvučene u tabeli, dok su ona rešenja HGA koja su bolja u odnosu na obe metode podvučena i boldovana. Kao što se može videti, HGA dostiže najbolja poznata rešenja, sem na instancama *pmed15, pmed23* i *pmed34,* gde je VNS postigao nešto bolje rezultate. Hibridni GA daje bolja rešenja od EP heuristike na svim testiranim p-medijan instancama, a od VNS metoda na 11 instanci problema *T4*. Testiranja u [Dom03] su izvedena na nešto sporijem računaru (Pentium III/800 MHz sa 1GB RAM), ali pošto je vreme izvršavanja HGA i do nekoliko puta

kraće u odnosu na EP metodu, može se zaključiti da je HGA brži od EP. Na nekoliko instanci, uglavnom manjih dimenzija (pmed1, pmed3, pmed6,...), VNS ima nešto kraće vreme izvršavanja od HGA. Ipak, treba napomenuti da je u t_{AMD}[s] koloni dato ukupno vreme rada hibridnog GA (do ispunjenja nekog od kriterijuma zaustavljanja), a iz kolone *t[s]* Tabele 4.5 se može videti da je vreme za koje HGA prvi put dobija najbolje rešenje znatno kraće (ostatak vremena algoritam i dalje pokušava da ga poboljša).

Tabela 4.6 Poređenje HGA sa EP i VNS za T4

			HGA		EP		VNS	
instanca	**n**	**p**	**GA_{najb}**	**t_{AMD}[s]**	**$EP_{najb.}$**	**t_{PIII}[s]**	**$VNS_{najb.}$**	**t_{PIII}[s]**
pmed1	100	5	*4523*	2.870	*4523*	25.20	*4523*	1.27
pmed2	100	10	*2987*	3.385	2993	36.98	*2987*	3.80
pmed3	100	10	*3067*	3.351	*3067*	36.91	3074	2.80
pmed4	100	20	***2137***	5.128	2153	60.80	2142	6.98
pmed5	100	33	*818*	4.837	839	92.08	*818*	8.22
pmed6	200	5	*6064*	8.754	*6064*	35.52	6078	7.88
pmed7	200	10	*4206*	11.766	4225	54.17	*4206*	13.41
pmed8	200	20	*3182*	12.202	3248	91.95	*3182*	28.30
pmed9	200	40	***1807***	18.481	1831	167.61	1816	66.39
pmed10	200	67	***818***	27.905	849	274.09	829	75.91
pmed11	300	5	*5979*	15.444	*5979*	47.75	*5979*	13.30
pmed12	300	10	*5021*	21.057	*5021*	73.83	*5021*	25.86
pmed13	300	30	*3133*	22.712	3175	183.25	*3133*	97.80
pmed14	300	60	***1949***	50.271	2027	346.42	1957	303.64
pmed15	300	100	1134	83.866	1181	549.67	***1133***	415.80
pmed16	400	5	*6341*	24.237	*6341*	56.06	*6341*	24.13
pmed17	400	10	***5381***	37.933	5440	89.30	5413	43.83
pmed18	400	40	***3437***	54.822	3463	309.50	3443	261.86
pmed19	400	80	***1926***	63.999	1973	618.88	1933	779.77
pmed20	400	133	***1146***	201.458	1191	1000.41	1152	1108.48
pmed21	500	5	*7245*	33.997	*7245*	71.69	*7245*	24.22
pmed22	500	10	***6685***	63.664	6749	117.88	6722	58.58
pmed23	500	50	3307	82.941	3379	461.50	***3306***	639.95
pmed24	500	100	*2005*	131.277	2068	888.27	*2005*	1455.81
pmed25	500	167	***1149***	537.011	1198	1524.86	1151	2552.02
pmed26	600	5	*7787*	46.058	7789	87.30	*7787*	48.11
pmed27	600	10	*6444*	85.349	6481	141.97	*6444*	141.70
pmed28	600	60	***3202***	146.599	3304	687.42	3210	1113.89
pmed29	600	120	*2006*	205.499	2087	1249.78	*2006*	3178.69
pmed30	600	200	***1295***	759.281	1359	1976.77	1308	4942.75
pmed31	700	5	*8046*	71.425	8047	90.81	*8046*	66.16
pmed32	700	10	***7278***	111.564	7318	148.77	7280	162.97
pmed33	700	70	***3405***	216.540	3463	857.47	3413	2377.72
pmed34	700	140	2033	681.274	2083	1624.61	***2023***	5657.56
pmed35	800	5	*8191*	83.567	*8191*	102.58	*8191*	72.58
pmed36	800	10	***7796***	198.654	7840	170.38	7820	201.64
pmed37	800	80	***3600***	358.066	3684	1086.13	3604	3170.70
pmed38	900	5	*8720*	102.448	8768	111.61	*8720*	140.84
pmed39	900	10	*7360*	170.158	7398	189.19	*7360*	313.03
pmed40	900	90	***3710***	693.154	3768	1372.98	3718	5422.73

Tabela 4.7 Rezultati T5

instanca	n	p	GA_{najb}	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	2941	0.062	2.874	2001	0.000	0.000	37197	62.9
pmed2	100	10	2075	0.417	3.621	2151	0.000	0.000	52267	51.5
pmed3	100	10	2160	0.787	4.069	2356	0.000	0.000	55805	52.7
pmed4	100	20	1532	0.571	4.000	2155	0.294	0.103	52692	51.2
pmed5	100	33	689	1.877	5.656	2785	0.479	0.438	70422	49.5
pmed6	200	5	3938	1.153	9.334	2171	0.000	0.000	50702	53.4
pmed7	200	10	2835	5.432	15.853	2870	0.159	0.296	77102	46.4
pmed8	200	20	2244	3.639	11.808	2457	0.018	0.038	69013	43.9
pmed9	200	40	1374	8.816	17.948	3145	0.655	0.301	89562	43.0
pmed10	200	67	633	5.278	18.018	2160	0.047	0.150	61363	43.3
pmed11	300	5	3869	1.276	15.945	2001	0.000	0.000	49619	50.5
pmed12	300	10	3344	7.830	27.603	2578	0.003	0.009	71536	44.5
pmed13	300	30	2198	14.165	28.888	3042	0.155	0.103	90408	40.6
pmed14	300	60	1492	22.184	37.480	2831	0.194	0.136	84613	40.3
pmed15	300	100	871	27.674	55.406	2821	0.207	0.169	85401	39.5
pmed16	400	5	4094	4.388	27.321	2187	0.000	0.000	56038	48.8
pmed17	400	10	3512	17.943	54.264	2865	0.000	0.000	81304	43.3
pmed18	400	40	2416	31.674	53.142	3132	0.182	0.173	95310	39.1
pmed19	400	80	1435	72.000	98.424	3296	0.251	0.290	101059	38.8
pmed20	400	133	905	107.478	162.580	3497	0.309	0.232	108728	37.9
pmed21	500	5	4578	2.098	34.359	2001	0.000	0.000	52332	47.8
pmed22	500	10	4310	19.817	71.925	2325	0.000	0.000	66437	43.0
pmed23	500	50	2325	58.304	87.609	2735	0.090	0.066	84362	38.4
pmed24	500	100	1486	94.203	133.997	2433	0.323	0.173	76195	37.5
pmed25	500	167	925	247.802	334.984	2857	0.346	0.363	90637	36.6
pmed26	600	5	4970	12.442	55.595	2354	0.030	0.049	62980	46.6
pmed27	600	10	4162	19.563	92.152	2268	0.000	0.000	67379	40.7
pmed28	600	60	2257	156.916	197.117	3933	0.111	0.111	123671	37.2
pmed29	600	120	1524	269.999	360.575	3049	0.571	0.278	96634	36.6
pmed30	600	200	1010	466.896	638.944	2412	0.317	0.223	77154	36.2
pmed31	700	5	5054	19.231	75.715	2307	0.002	0.006	62605	45.8
pmed32	700	10	4683	54.407	141.569	2829	0.009	0.011	84177	40.5
pmed33	700	70	2361	284.940	361.220	3872	0.224	0.207	122063	37.0
pmed34	700	140	1519	621.714	724.288	3683	0.421	0.351	118098	35.9
pmed35	800	5	5209	7.729	75.018	2001	0.000	0.000	53771	46.3
pmed36	800	10	4981	121.640	223.647	3529	0.060	0.127	105120	40.5
pmed37	800	80	2542	311.274	382.012	3002	0.220	0.141	95705	36.3
pmed38	900	5	5542	17.619	102.503	2001	0.000	0.000	55019	45.1
pmed39	900	10	4745	21.906	149.329	2001	0.000	0.000	59650	40.5
pmed40	900	90	2578	737.696	807.783	3788	0.128	0.093	120770	36.3

Tabela 4.8 Rezultati T6

instanca	n	p	GA_{najb}	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	2873	0.263	3.032	2108	0.000	0.000	38317	63.7
pmed2	100	10	2018	0.224	3.424	2035	0.000	0.000	48612	52.3
pmed3	100	10	2062	0.110	3.330	2001	0.000	0.000	45747	54.3
pmed4	100	20	1491	0.881	4.204	2350	0.557	0.319	58336	50.6
pmed5	100	33	662	1.494	5.384	2607	0.076	0.147	66755	48.9
pmed6	200	5	3884	0.651	8.819	2032	0.000	0.000	47886	52.9
pmed7	200	10	2795	4.051	14.556	2597	0.061	0.034	69791	46.3
pmed8	200	20	2190	4.913	13.159	2833	0.018	0.044	79811	43.7
pmed9	200	40	1350	6.397	14.923	2571	0.281	0.495	73753	42.8
pmed10	200	67	619	5.756	19.369	2290	0.113	0.078	65399	43.0
pmed11	300	5	3827	0.656	15.441	2001	0.000	0.000	49876	50.2
pmed12	300	10	3288	7.356	27.313	2557	0.000	0.000	71822	43.9
pmed13	300	30	2173	13.503	28.273	2767	0.060	0.065	82062	40.7
pmed14	300	60	1477	19.064	36.780	3026	0.339	0.181	90123	40.5
pmed15	300	100	862	23.169	46.743	2548	0.151	0.182	77774	39.1
pmed16	400	5	4065	11.060	34.916	2769	0.000	0.000	71018	48.8
pmed17	400	10	3487	15.492	51.314	2573	0.011	0.015	73180	43.2
pmed18	400	40	2388	37.954	58.429	3301	0.188	0.164	99769	39.6
pmed19	400	80	1418	62.315	97.285	2938	0.205	0.131	90034	38.7
pmed20	400	133	895	96.542	183.124	2270	0.212	0.082	71958	36.7
pmed21	500	5	4560	2.865	35.347	2001	0.000	0.000	52314	47.8
pmed22	500	10	4269	7.689	58.798	2001	0.000	0.000	57125	43.0
pmed23	500	50	2301	84.803	117.215	3421	0.139	0.091	106687	37.7
pmed24	500	100	1479	122.243	176.714	2873	0.257	0.127	90079	37.6
pmed25	500	167	917	281.377	392.986	2684	0.327	0.267	86549	35.5
pmed26	600	5	4945	12.903	56.524	2406	0.000	0.000	64299	46.7
pmed27	600	10	4143	28.237	100.066	2470	0.005	0.015	72884	41.1
pmed28	600	60	2243	108.305	150.247	2684	0.147	0.113	84266	37.3
pmed29	600	120	1515	323.628	391.856	3198	0.508	0.516	102922	35.7
pmed30	600	200	1007	442.130	737.999	2581	0.258	0.205	83612	35.3
pmed31	700	5	5031	10.674	67.848	2191	0.000	0.000	59637	45.6
pmed32	700	10	4610	23.701	110.264	2083	0.000	0.000	61929	40.7
pmed33	700	70	2346	246.416	302.423	3713	0.277	0.271	117587	36.9
pmed34	700	140	1512	489.824	690.052	3060	0.311	0.261	98400	35.8
pmed35	800	5	5189	7.518	78.382	2001	0.000	0.000	54248	45.9
pmed36	800	10	4953	99.735	203.422	2867	0.020	0.064	84949	40.9
pmed37	800	80	2523	456.560	510.649	3949	0.139	0.163	126161	36.1
pmed38	900	5	5518	28.818	114.217	2001	0.000	0.000	53249	46.9
pmed39	900	10	4678	36.858	165.301	2001	0.000	0.000	58830	41.3
pmed40	900	90	2566	648.561	733.214	3681	0.249	0.179	117117	36.4

Tabela 4.9 Rezultati T7

instanca	n	p	GA_{najb}	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	3924	0.068	2.906	2001	0.000	0.000	37123	63.0
pmed2	100	10	2769	1.220	4.487	2630	0.000	0.000	64550	51.0
pmed3	100	10	2874	0.280	3.521	2070	0.000	0.000	46439	55.2
pmed4	100	20	2061	0.213	3.600	2001	0.000	0.000	48529	51.6
pmed5	100	33	923	2.036	5.931	2957	0.141	0.089	74101	49.9
pmed6	200	5	5250	0.347	8.561	2001	0.000	0.000	46768	53.3
pmed7	200	10	3778	3.177	13.749	2394	0.000	0.000	64154	46.5
pmed8	200	20	2993	5.386	13.312	2770	0.010	0.023	77803	43.9
pmed9	200	40	1834	8.974	18.012	3034	0.545	0.305	86915	42.7
pmed10	200	67	844	7.916	18.828	2648	0.036	0.112	75102	43.3
pmed11	300	5	5155	0.744	15.633	2001	0.000	0.000	49594	50.5
pmed12	300	10	4450	5.400	25.522	2397	0.000	0.000	67097	44.1
pmed13	300	30	2939	10.640	26.314	2691	0.085	0.102	80006	40.6
pmed14	300	60	1992	23.566	38.669	2919	0.141	0.091	86524	40.8
pmed15	300	100	1162	26.112	47.977	2844	0.164	0.131	85838	39.6
pmed16	400	5	5456	1.707	25.235	2001	0.000	0.000	51804	48.3
pmed17	400	10	4681	13.497	48.940	2463	0.006	0.020	69892	43.3
pmed18	400	40	3225	34.954	55.774	2994	0.180	0.131	90027	39.9
pmed19	400	80	1913	54.795	76.892	2906	0.105	0.055	88014	39.6
pmed20	400	133	1198	83.603	109.404	3207	0.242	0.150	98786	38.4
pmed21	500	5	6106	5.449	38.262	2186	0.000	0.000	57197	47.7
pmed22	500	10	5745	26.849	78.547	2564	0.000	0.000	74128	42.3
pmed23	500	50	3094	46.140	74.769	2269	0.055	0.051	69503	38.9
pmed24	500	100	1984	147.025	193.475	3370	0.156	0.122	105123	37.7
pmed25	500	167	1224	239.131	356.267	3489	0.466	0.400	110446	36.7
pmed26	600	5	6628	15.982	59.624	2501	0.000	0.000	67000	46.5
pmed27	600	10	5556	29.468	100.579	2561	0.014	0.019	75637	41.0
pmed28	600	60	3007	104.083	144.859	2840	0.216	0.161	88495	37.8
pmed29	600	120	2033	227.536	312.067	3036	0.226	0.205	95959	36.9
pmed30	600	200	1337	451.022	639.431	3201	0.337	0.280	101976	36.4
pmed31	700	5	6735	5.112	60.697	2001	0.000	0.000	54479	45.6
pmed32	700	10	6226	18.411	105.574	2001	0.000	0.000	59548	40.6
pmed33	700	70	3147	220.565	276.079	2851	0.232	0.179	89871	37.1
pmed34	700	140	2025	571.798	749.744	3557	0.247	0.210	113339	36.4
pmed35	800	5	6944	9.685	78.138	2001	0.000	0.000	53714	46.4
pmed36	800	10	6639	39.803	144.872	2001	0.000	0.000	58751	41.4
pmed37	800	80	3385	350.850	420.676	3788	0.074	0.071	119395	37.0
pmed38	900	5	7390	38.772	128.325	2432	0.005	0.017	67168	44.8
pmed39	900	10	6310	40.917	170.124	2001	0.000	0.000	58808	41.3
pmed40	900	90	3436	606.491	671.071	3873	0.201	0.176	123535	36.3

Tabela 4.10 Rezultati T8

instanca	n	p	GA_{najb}	t[s]	t_{tot}[s]	gen	od_{sr}[%]	σ[%]	eval	keš [%]
pmed1	100	5	1986	0.041	2.925	2001	0.000	0.000	37654	62.4
pmed2	100	10	1400	0.624	3.986	2258	0.114	0.114	56019	50.5
pmed3	100	10	1456	0.489	3.833	2162	0.000	0.000	50305	53.5
pmed4	100	20	1040	1.141	4.539	2484	0.183	0.183	61136	51.0
pmed5	100	33	468	1.586	5.294	2680	0.769	0.769	67231	49.9
pmed6	200	5	2642	0.652	8.935	2056	0.000	0.000	47943	53.4
pmed7	200	10	1902	5.610	16.302	2877	0.063	0.063	77671	46.1
pmed8	200	20	1516	6.765	15.133	3113	0.066	0.066	87831	43.6
pmed9	200	40	924	7.566	16.892	2823	0.823	0.823	81123	42.7
pmed10	200	67	424	10.359	21.568	2808	0.684	0.684	79333	43.5
pmed11	300	5	2588	1.886	17.061	2119	0.000	0.000	52482	50.6
pmed12	300	10	2249	1.276	21.870	2001	0.000	0.000	55879	44.2
pmed13	300	30	1475	12.439	26.703	2851	0.156	0.156	84851	40.6
pmed14	300	60	999	17.652	33.879	2550	0.390	0.390	76195	40.3
pmed15	300	100	581	26.884	49.187	2658	0.998	0.998	80905	39.3
pmed16	400	5	2735	4.856	28.835	2193	0.000	0.000	56281	48.8
pmed17	400	10	2347	16.289	51.876	2671	0.000	0.000	75921	43.2
pmed18	400	40	1618	34.660	56.289	3211	0.210	0.210	98189	39.0
pmed19	400	80	959	62.386	96.298	3373	0.626	0.626	103610	38.7
pmed20	400	133	604	122.004	168.466	3376	1.109	1.109	105505	37.5
pmed21	500	5	3062	6.503	39.455	2252	0.000	0.000	58684	48.0
pmed22	500	10	2888	28.145	82.059	2638	0.000	0.000	76800	41.9
pmed23	500	50	1554	63.016	94.280	3159	0.290	0.290	99007	37.3
pmed24	500	100	1000	154.738	209.639	2706	0.250	0.250	85513	36.9
pmed25	500	167	619	298.659	440.307	3062	0.824	0.824	97826	36.0
pmed26	600	5	3323	15.761	60.933	2589	0.000	0.000	69814	46.2
pmed27	600	10	2784	49.141	123.233	2979	0.007	0.007	88631	40.5
pmed28	600	60	1512	146.934	193.419	3137	0.159	0.159	98461	37.3
pmed29	600	120	1022	344.267	431.098	3521	0.881	0.881	112910	35.9
pmed30	600	200	684	981.975	1216.51	3388	0.629	0.629	111748	34.1
pmed31	700	5	3374	17.313	75.129	2365	0.003	0.003	64265	45.7
pmed32	700	10	3143	22.957	112.060	2261	0.000	0.000	67987	40.0
pmed33	700	70	1583	191.245	266.069	2909	0.164	0.164	91240	37.3
pmed34	700	140	1019	597.877	730.627	3392	0.697	0.697	110098	35.1
pmed35	800	5	3479	7.915	76.572	2001	0.000	0.000	53786	46.3
pmed36	800	10	3330	80.213	186.669	2668	0.108	0.108	78618	41.2
pmed37	800	80	1704	404.663	472.201	3895	0.329	0.329	124579	36.2
pmed38	900	5	3704	47.404	137.476	2533	0.011	0.011	69452	45.3
pmed39	900	10	3187	35.608	165.892	2001	0.000	0.000	59223	40.9
pmed40	900	90	1726	708.072	808.157	3526	0.214	0.214	112725	36.1

Iz svega navedenog može se zaključiti da je hibridizacija GA sa modifikovanom heuristikom zamene uspešan metod za rešavanje DOMP-a koji daje rešenja zadovoljavajućeg kvaliteta u poređenju sa EP i VNS heuristikom za one klase DOMP problema na kojima je bilo moguće porediti rezultate. Prikazana su i najbolja rešenja HGA za probleme *T5-T8* na instancama p-medijane koja do sada nisu postojala u literaturi i za koje nije poznato optimalno rešenje. Zbog svojih dobrih performansi na instancama većih dimenzija, hibridni genetski algoritam ima značajno mesto među postojećim metodama za rešavanje ovog lokacijskog problema.

5. ZAKLJUČAK

U ovom radu je opisana GA implementacija za rešavanje tri diskretna lokacijska problema: p-hab medijane, p-hab centra i diskretno uređen problem medijane. Za svaki od rešavanih problema dati su: matematička formulacija i kratak opis problema, pregled postojećih metoda, opis predloženog GA, dobijeni rezultati i njihovo poređenje sa rezultatima ostalih metoda.

5.1 Pregled primenjenih metoda i dobijenih rezultata

Predložena GA implementacija koristi glavne aspekte genetskih algoritama za rešavanje NP-kompletnih problema, uz modifikaciju koncepta GA u skladu sa prirodom rešavanih problema. Implementacija je podeljena u tri dela: glavni deo koji je zajednički za sve probleme, deo koji se odnosi na diskretne lokacijske probleme i programski segment za svaki konkretan problem. Koristeći ovakvu strukturu implementacije, moguće je relativno lako rekonfigurisati i dopuniti sistem, tako da se može primeniti i na druge slične lokacijske probleme.

Pri rešavanju sva tri lokacijska problema korišćeno je binarno kodiranje. Osim efikasnog računanja vrednosne funkcije, primenjeno je popravljanje nekorektnih jedinki u populaciji do korektnih. Na taj način omogućena je primena modifikovanih genetskih operatora koji čuvaju korektnost, čime se isključuje pojava nekorektnih jedinki u svakoj generaciji. Najbolje rezultate pokazali su: fino gradirana turnirska selekcija, modifikovano uniformno ukrštanje i mutacija sa zaleđenim bitovima. Keširanje poboljšava performanse GA, ali ne utiče na njegov rad. Primenjene su i razne metode za sprečavanje preuranjene konvergencije, gubljenja raznovrsnosti genetskog materijala i spore konvergencije genetskog algoritma. U poglavlju 4. opisana je hibridizacija GA sa jednom modifikacijom heuristike zamene u cilju poboljšanja kvaliteta rešenja DOMP-a.

Dalje proširenje i unapređivanje datih rezultata može se izvršiti u nekoliko pravaca:

- paralelizacija opisanog genetskog algoritma i izvršavanje na višeprocesorskom računaru,
- hibridizacija sa nekim drugim heuristikama i/ili egzaktnim metodama,
- primena ovog pristupa na neke druge lokacijske probleme.

5.2 Naučni doprinos rada

Najznačajniji novi rezultati u ovom radu su:

- implementacija keširanja za ubrzavanje heuristike zamene,
- korišćenje vrednosti potencijalnih rešenja dobijenih heuristikom u glavnom delu GA,
- primena binarnog kodiranja bez pojave nekorektnih jedinki popravljanjem u vrednosnoj funkciji i čuvanjem korektnosti pomoću modifikovanih genetskih operatora,
- ograničavanje broja jedinki u populaciji sa istom vrednošću a različitim genetskim kodom.

Predloženi pristup predstavlja pogodno sredstvo za rešavanje široke klase NP-teških diskretnih lokacijskih problema. Zbog svojih, gore navedenih karakteristika, naučno istraživanje opisano u ovom radu daje doprinos oblastima kombinatorne optimizacije, lokacijskih problema i genetskih algoritama. Dobijeni rezultati su u pripremi za objavljivanje u inostranim i domaćim časopisima.

LITERATURA

[Ah93] **Ahuja R., Magnanti T., Orlin J.,** "Network Flows: Theory, Algorithms and Applications", Prentice-Hall, New York (1993).

[AbH98] **Abdinnour-Helm S.**, "A hybrid heuristic for the uncapacitated hub location problems", *European Journal of Operational Research*, Vol. 106, pp.489-499 (1998).

[AbH99] **Abdinnour-Helm S.**, "Network design in supply chain management", *Internatonal Journal of Agile Mangement Systems,* Vol. 1, No. 2., pp.99-106 (1999).

[Alk04] **Alkhalifah Y., Wainwright R.L.,** "A Genetic Algoriyhm Applied to Graph Problems Involving Subsets of Vertices", *CEC* (2004).

[Alp03] **Alp O., Erkut E.**, "An Efficient Genetic Algorithm for the p-Median Problem", *Annals of Operations Research*, Kluwer Academic Publishers (2003).

[Ant89] **Antonisse J.**, "A new interpretation of schema that overturns the binary encoding constraint", in: *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, Calif., pp. 86-91 (1989)

[Ayk95] **Aykin T.**, "Network Policies for Hub-and-Spoke Systems with Application to the Air Transportation System", *Transportation Science*, Vol.29, pp.201-221 (1995)

[Bea96] **Beasley J.E.**, "Obtaining Test Problems via Internet", *Journal of Global Optimization*, Vol. 8, pp. 429-433 (1996)
http://msmga.ms.ic.ac.uk/jeb/orlib/info.html

[BeD93a] **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).
ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps

[BeD93b] **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).
ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview2.ps

[Bëc92a] **Bëck T.**, "Self-adaptation in Genetic Algorithms", in: *Proceedings of the First European Conference on Artificial Life*, MIT Press (1992).
ftp://lumpi.informatik.uni-dortmund.de/pub/GA/papers/ecal92.ps.gz

[Bëc93] **Bëck T.**, "Optimal Mutation Rates in Genetic Search", in: *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, Calif., pp. 2-8 (1993).
ftp://lumpi.informatik.uni-dortmund.de/pub/GA/papers/icga93.ps.gz

[Bol03] **Boland N., Domínguez-Marin P., Nickel S., Puerto J.**, "Exact procedures for solving the Discrete Ordered Median Problem", *ITWM Bericht,* Vol. 47, Fraunhofer Institut für Tecno-und Wirtschaftsmathematik (ITWM), Kaiserslautern, Germany (2003).

[Bol04] **Boland N., Krishnamoorthy M., Ernst A.T., Ebery J.**, "Preprocessing and Cutting for Multiple Allocation Hub Location Problems", *European Journal of Operational Research*, Vol. 155, pp.638-653 (2004).

[Boz02] **Bozkaya B., Zhang J., Erkut E.,** "An Efficient Genetic Algorithm for the p-median problem", in Hamacher H. and Drezner Z, eds.: *Facility Location : Applications and Theory*, Springer-Verlag, Berlin-Heidelberg, pp.179-204 (2002)

[Bra89] **Brandeau M., Chiu S**., "An Overview of Representative Problems in Location Research", *Management Science*, Vol. 35, pp.645-674 (1989).

[Brml91] **Bramlette M.F.**, "Initialisation, mutation and selection methods in genetic algorithms for function optimization", in: *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, Calif., pp. 100-107 (1991).

[Cam94] **Campbell, J. F.,** " Integer programming formulations of discrete hub location problems", *European Journal of Operational Research* Vol. 72, pp. 387-405 (1994)

[Cam96] **Campbell J.F.**, "Hub Location and the p-hub Median Problem", *Operations Research,* Vol. 44, No. 6, pp. 923-935 (1996)

[Cam02] **Campbell J.F.,Ernst A. and Krishnamoorthy M.**, "Hub Location Problems" in Hamacher H. and Drezner Z., eds.: *Facility Location : Applications and Theory*, Springer-Verlag, Berlin-Heidelberg, pp. 373-407 (2002)

[Cor01] **Correa E.S., Steiner M.T., Freitas A., Carnieri C.,** "A Genetic Algorithm for the p-median Problem", *GECCO*, (2001).

[Cre97] **Crescenci P., Kann V.,** "A compendium of NP optimization problems" (1997). *http://www.nada.kth.se/theory/problemlist.html*

[Čan96] **Čangalović M.**, "Opšte heuristike za rešavanje problema kombinatorne optimizacije", u: *Kombinatorna optimizacija: Matematička teorija i algoritmi*, str. 320-350 (1996).

[DD93] **Dibble, C., Densham P.J.**, "Generating Interesting Alternatives in GIS and SDSS Using Genetic Algorithms", *GISILIS symposium,* University of Nebras, Lincoln (1993).

[Dre03] **Drezner Z., Erkut E.,** "An Efficient Genetic Algorithm for the p-median Problem", *Annals of Operations Research,* Vol. 122, pp.21-42 (2003).

[Dea85] **Dearing P.M.**, "Location problems", *Operations Research Letters*, Vol. 4, pp. 95-98 (1985).

[DJo75] **De Jong K.E.**, "An analysis of the behavior of a class of genetic adaptive systems", *PhD thesis*, University of Michigan (1975).

[Dom03] **Domínguez-Marin P., Hansen P., Mlademović N., Nickel S.**, "Heuristic Procedures for Solving the Discrete Ordered Median Problem", *ITWM Bericht,* Vol. 46, Fraunhofer Institut für Tecno-und Wirtschaftsmathematik (ITWM), Kaiserslautern, Germany (2003).

[Dor96] **Dorigo M., Maniezzo V., Colorni A.**, "Ant System: Optimizing by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, Vol. 26, No. 1, pp. 29-41 (February 1996).

[Dre02] **Drezner Z., Klamroth K., Schöbel A., Weslowsky G.O.,**The Weber problem, in H. Hamacher and Z. Drezner, eds.: *Facility Location : Applications and Theory*, Springer-Verlag, Berlin-Heidelberg, 1-25 (2002).

[Dv99] **Dvorett J.**, "Compatibility-Based Genetic Algorithm: A New Approach to thr p-Median Problem" (1999).

[Ern98] **Ernst A.T., Krisnamoorthy M.**, "Exact and Heuristic Algorithms for the Uncapacitated Multiple Allocation p-hub Median Problem ", *European Journal of Operational Research,* Vol.104., pp.100-112 (1998)

[Ern98a] **Ernst, A.T., Krishnamoorthy M**.," An Exact Solution Approach Based on Shortest-paths for p-hub Median Problem", *INFORMS Journal of Computing*, Vol. **10**, pp. 149-162 (1998).

[Ern04a] **Ernst, A.T., Hamacher H. , Jiang H., Krishnamoorthy M., Woeginger G.,** "Heuristic Algorithms for the Uncapacitated Hub Center Single Allocation Problem", *European Journal of Operational Research* (2004.to appear)

[Ern04b] **Ernst, A.T., Hamacher H. , Jiang H., Krishnamoorthy M., Woeginger G.**, "Uncapacitated Single and Multiple Allocation p-Hub Center Problems", *Operations Research*, (2004. to appear)

[Fan90] **Fang L., Li T.**, "Design of competition based neural networks for combinatorial optimization", *International Journal on Neural System,* Vol. 3, pp. 221-235 (1990).

[Fil98] **Filipović V.** "Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama", *Magistarski rad*, Univerzitet u Beogradu, Matematički fakultet (1998).

[Fil00] **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).

[Fil01] **Filipović V., Tošić D., Kratica J.,** "Experimental Results in Applying of Fine Grained Tournament Selection", Proceedings of the 10th Congress of Yugoslav Mathematicians, pp. 331-336, Belgrade, 21.-24.01. (2001).

[Fil03] **Filipović, V.** "Fine-Grained Tournament Selection Operator in Genetic Algorithms", *Computing and Informatics* **22**(2), 143-161.(2003).

[Fra92] **Francis R.L., McGinnis L.F., White J.A.**, "Facility Layout and Location: An Analytical Approach", Second Edition, Prentice-Hall International, Englewood Cliffs, NJ (1992).

[Fra00] **Francis R.L., Lowe T.J., Tamir A.**, "Aggregation error bounds for a class of location models", *Operations Research*, Vol. 48, pp. 294-307 (2000).

[Gar79] **Garey M.R.**, **Johnson D.S.**, "Computers and Intractibility: A Guide to the Theory of NP Completness", W.H. Freeman and Co. (1979).

[Geo74] **Geoffrion A., McBride R.**, "Lagrangean Relaxation for Integer Programming", *Mathematical Programming Study*, Vol. 2, pp. 82-114 (1974).

[Glo86] **Glover F.**, "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research*, Vol. 5, pp. 533-549 (1986).

[Glo90] **Glover F.**, "Tabu search: A Tutorial", *Interfaces*, Vol 20, pp. 74-94 (1990).

[Glo03] **Glover F., Kochenberger G.A.**, "Handbook of Metaheuristics", Kluwer Academic Publishers, Boston-Dordrecht-London (2003).

[Gol89] **Goldberg D.E.**, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publ. Comp., Reading, Mass., 412 pp (1989).

[Ham04] **Hamacher H.W., Labbe M., Nickel S., Sonneborn T.**, "Adapting Polyhedral Properties from Facility to Hub Location Problems", *Discrete Applied Mathematics* (2004) to appear

[Hll75] **Holland J.H.**, "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor (1975).

[HM97] **Hansen P., Mladenović N.,** "Variable neighborhood search for the p-median", *Location Science*, Vol. 5, No. 4, pp. 207-226 (1997).

[HM01a] **Hansen P., Mladenović N.,** "Developments of Variable Neighborhood Search", in Ribeiro C.C. and Hansen P. eds.: *Essays and Surveys in Metaheuristics,* Kluwer Academic Publishers, pp.415-439 (2001).

[HM01b] **Hansen P., Mladenović N.,** " Variable Neighborhood Search", in Glover F. and Kochenberg G. eds.: *Handbook of Metaheuristics* (2001).

[HM01c] **Hansen P., Mladenović N.,** " Variable neighborhood search: Principles and applications ", *European Journal of Operations Research*, Vol.130, No.3, pp. 449-467 (2001).

[Hut02] **Hutter M.,** "Fitness Uniform Selection to Preserve Genetic Diversity", in *Proceedings of the 2002 Congress on Evolutionary Computation*, CEC-2002, Hawaii, pp. 783-788 (2002).

[Geo74] **Geoffrion A., McBride R.**, "Lagrangean Relaxation for Integer Programming", *Mathematical Programming Study*, Vol. 2, pp. 82-114 (1974).

[Jan91] **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).

[Kal03] **Kalesics J., Nickel S., Puerto J.**, "Multifacility Ordered Median Problems on networks: a further analysis", *Networks*, Vol. 41, No. 1, pp.1-12 (2003).

[Kir83] **Kirkpatrick S., Gellat C., Vecchi M.**, "Optimization by simulated annealing", *Science*, Vol. 220, pp. 671-680 (1983).

[KH79] **Kariv O., Hakimi S.I.**, "An algorithmic approach to network location problems.II :The p-median", *SIAM Journal on Applied Mathematics*, Vol.37, pp. 539-560 (1979).

[Kra98] **Kratica J.,Filipović V., Tošić D.**, "Solving the uncapacitated Warehouse Location problem by SGA Eith Add-Heuristic", *XV ECPD International Conference on Material Handling and Warehousing*, University of Belgrade, Faculty of Mechanical Engineering, Materials Handing Institute, Belgrade (1998).

[Kra99] **Kratica J.**, "Improving Performances of the Genetic Algorithm by Caching", *Computers and Artificial Inteligence,* Vol. 18, No. 3, pp.271-283 (1999).

[Kra00] **Kratica J.,** "Paralelizacija genetskih algoritama za rešavanje nekih NP-kompletnih problema", Doktorska disertacija, Matematički fakultet, Beograd (2000).

[Kra01] **Kratica J., Tošić D., Filipović V., Ljubić I.**, "Solving the Simple Plant Location Problem by Genetic Algorithm", *RAIRO Operations Research*, Vol. 73, No. 1, pp.127-142 (2001)

[Kra02] **Kratica J., Tošić D., Filipović V., Ljubić. I**., "A genetic algorithm for the uncapacitated network design problem", *Soft Computing in Industry - Recent Applications*, Engineering series, pp. 329-338. Springer (2002).

[Kra03] **Kratica J., Ljubić I., Tošić D.,** "A genetic algorithm for the index selection problem", *Springer Lecture Notes in Computer Science*, Vol. 2611, pp. 281-291 (2003).

[Lor00] **Lorena L.A.N., Furtado J.C**., "Construstive genetic Alhorithm for Clustering Problem", *Evolutionary Computation*, Massachusetts Institute of Technology (2000).

[Lov88] **Love R.F., Morris J.G., Wesolowshy G.O.**, *Facilities Location: Models & Methods*, North-Holland, New York (1988).

[Lvi93a] **Levine D.**, "A Parallel Genetic Algorithm for the Set Partitioning Problem, *PhD thesis*, Argonne National Laboratory, ANL-94/23, Illinois Institute of Technology, (1993). ftp://info.mcs.anl.gov/pub/tech_reports/reports/ANL9423.ps.Z

[Lju00] **Ljubić I., Kratica J.,** "A Genetic Algorithm for the Biconnectivity Augmentation Problem", *Proceedings of the Conference on Evolutionary Computation - CEC 2000*, pp. 89-96, San Diego, CA, USA, July 16-19 (2000).

[Lju00a] **Ljubić I., Raidl G.R., Kratica J**., "A Hybrid GA for the Edge-Biconnectivity Augmentation Problem", *Springer Lecture Notes in Computer Science* ,Vol. 1917, pp. 641-650 (2000).

[Lju04] **Ljubić I.**, "Exact and Memetic Algorithms for Two Network Design Problems", *PhD thesis*, Institute of Computer Graphics, Vienna University of Technology (2004).

[Mic96] **Michalewicz Z.**, "Genetic Algorithms + Data Structures = Evolution Programs", Third Edition, Springer Verlag, Berlin Heideleberg (1996).

[Mih03] **Mihelic J., Robic B.**, "Genetic algorithm for the k-center location problem", *XIV EWGLA*, Corfu, Greece (2003).

[Mir90] **Mirchandani P.B. and Francis R.L.**, "Discrete Location Theory", John Wiley & Sons (1990).

[Mit96] **Mitchell M.,** "An Introduction to Genetic Algorithms", MIT Press (1996).

[Mla95] **Mladenović N.**, "A variable neighborhood algorithm - a new metaheuristics for combinatorial optimization", Abstracts of papers presented at Optimization days, Montreal (1995).

[Mla97] , **Mladenović N., Hansen P.** "Variable neighborhood search", *Computers Operations Research,* Vol. 24, pp. 1097-1100, (1997).

[Mla04] **Mladenović N.,***Kontinualni lokacijski problemi,* Matematički institut SANU, Beograd (2004).

[Müh97] **Mühlenbein H.**, "Genetic algorithms", *Local Search in Combinatorial* Optimization, eds. Aarts E.H.L., Lenstra J.K., John Wiley & Sons Ltd., pp. 137-172 (1997).

[MV96] **Moreno Vega J. M.**, "Metaheurísticias en Localización", Análisias Teórico y Experimental", PhD Thesis (in spanish), University of La Laguna (1996)

[NP99] **Nickel S., Puerto J.**, "A unified approach to network location problems", *Networks*, Vol. 34, pp. 283-290 (1999).

[Nic01] **Nickel S.,** "Discrete Ordered Weber Problem", in Fleichmann B., Lach R. and Derigs U. eds.: *Operations Research Proceedings 2000.*, pp. 71-76, Springer (2001).

[OK91] **O'Kelly, M. E., Miller H.J.,** "Solution strategies for the single facility minimax hub location problem", *Papers in Regional Science: The Journal of the RSAI* Vol. 70, pp. 367-380 (1991).

[OK96] **O'Kelly M., Bryan D., Skorin-Kapov D., Skorin Kapov J.,** "Hub Network Design with Single and Multiple Allocation: A Computational Study", *Location Science*, Vol. 3, pp. 125-138 (1996)

[Orv93] **Orvosh D., Davis L.**, "Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints", in: *Proceedings of the Fifth International Conference on Genetic Algorithms - ICGA '93,* Morgan Kaufmann, San Mateo, Calif., p. 650 (1993).

[PF95] **Puerto J., Fernández F.R.,** "The symmetrical single facility location problem", *Techical Report*, Faculty of Mathematics, University of Sevilla (1995).

[PF00] **Puerto J., Fernández F.R.,** "Geometrical properties of the symmetrical single facility location problem", *Journal of Nonlinear and Convex Analysis*, Vol. 1, No. 3, pp. 321-342 (2000).

[Ree93] **Reeves, C.R.,**"Genetic Algorithms", Modern Heuristic Techniques for Combinatorial Problems, John Willy and Sons, New York (1993).

[Rod00] **Rodríguez-Chía A., Nickel S., Puerto J., Fernández F.R.,** "A flexibile approach to location problems", *Mathematical Methods of Operations Research*, Vol. 51, pp. 69-89 (2000).

[Sko94] **Skorin-Kapov D., Skorin Kapov J.,** "On Tabu Search for the Location of Interacting Hub Facilities", *European Journal of Operational Research*, Vol. 73, pp. 502-509 (1994).

[Sko96] **Skorin-Kapov D., Skorin Kapov J., O'Kelly M.**, "Tight Linear Programming Relaxations of Uncapacitated p-hub Median Problems", *European Journal of Operational Research*, Vol. 94, pp. 582-593 (1996).

[SmA93] **Smith A.E., Tate D.M.**, "Genetic optimization using a penalty function", in: *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, Calif., pp. 499-505 (1993).

[Soh98] **Sohn J., Park S.**, "Efficient Solution Procedure and Reduced Size Formulations for p-hub Location Problems", *European Journal of Operational Research*, Vol. 108, pp.118-126 (1998).

[Spe91] **Spears W., De Jong K.**, "On the virtues of parametrized uniform crossover", in: *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, Calif., pp. 230-236 (1991).
ftp://ftp.aic.nrl.navy.mil/pub/papers/1991/AIC-91-022.ps.Z

[Sri94] **Srinivas M., Patnaik L.M.**, "Genetic Algorithms: A Survey", *IEEE Computer*, pp. 17-26 (June 1994).

[Tošo4] **Tošić D., Mladenović N., Kratica J., Filpović V.,** "Genetski algoritmi*",* Matematički institut SANU, Beograd (2004).

[Vuj96] **Vujošević M., Stanojević M., Mladenović N.**, "Metode optimizacije: mrežni, lokacijski i višekriterijumski modeli", *Društvo operacionih istraživača Jugoslavije,* Beograd (1996).

[Web09] **Weber A**.,"Über den Standort der Industrien", Tübingen (1909),(English translation by Friedrrich C.J.,*Theory of the Location of Industries*,University of Chicago Press (1909).

[Yur94] **Yuret D.**, "From Genetic Algorithms to Efficient Optimization", *MSc Thesis*, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (1994).
http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Yuret.MSc_Thesis.From_Genetic_Algorithms_to_Efficient_Optimization.ps.gz

SADRŽAJ