Univerzitet u Beogradu
Matematički fakultet

Maja Đukić

HIBRIDNI GENETSKI ALGORITAM ZA REŠAVANJE HAB LOKACIJSKOG PROBLEMA NEOGRANIČENIH KAPACITETA SA VIŠESTRUKIM ALOKACIJAMA

Diplomski – master rad

B e o g r a d
2010.

Mentor:	**Doc. dr Zorica Stanimirović** Matematički fakultet u Beogradu
Komentor:	**dr Jozef Kratica** Viši naučni saradnik Matematički institut SANU
Član komisije:	**Doc. dr Vladimir Filipović** Matematički fakultet u Beogradu

_____

(datum odbrane)

Hibridni genetski algoritam za rešavanje hab lokacijskog problema neograničenih kapaciteta sa višestrukim alokacijama

Rezime

U ovom radu je opisan genetski algoritam za rešavanje hab lokacijskog problema neograničenih kapaciteta sa višestrukim alokacijama. Ovaj NP-težak problem ima značajnu primenu u praksi. Hab lokacijski problemi se dosta koriste u modernim transportnim i telekomunikacijsnim sistemima. Najčešće se primenjuju za dizajniranje drumskih i železničkih sistema, poštanskih sistema, sistema brze isporuke i slično. U cilju poboljšanja efikasnosti predloženog genetskog algoritma, primenjena je hibridizacija genetskog algoritma sa heuristikom lokalnog pretraživanja, pa su tako nastale metode koje su veoma uspešne i pri rešavanju problema velikih dimenzija. U radu su data rešenja i za probleme velikih dimenzija ($n = 130, 200$) za koje optimalna rešenja nisu poznata. Genetski algoritam je testiran na instancama čije optimalno rešenje nije poznato, ali dao je rezultate za koje se može pretpostaviti da su kvalitetni.

Ključne reči: genetski algoritmi, hab lokacijski problemi, kombinatorna optimizacija, metaheuristike

A Hybrid Genetic Algorithm for Solving the Uncapacitated Multiple Allocation Hub Location Problem

Abstract

In this paper, a hybrid genetic algorithm for solving the Uncapacitated Multiple Allocation Hub Location Problem − UMAHLP is proposed. This NP-hard problem has significant application in designing modern transportation and telecommunication networks, such as road and railway systems, postal systems, systems of fast delivery, etc. In order to improve the efficiency, genetic algorithm is combined with the local search heuristic. The proposed hybrid method shows to be very successful in solving problems of large dimensions with up to $n = 120$ nodes. It is also tested on instances with $n = 130$ and $n = 200$ nodes for which no optimal solution is presented in the literature so far. Although the optimal solutions are not known, we believe that the proposed hybrid method provides high quality solutions on these problem instances unsolved before.

Keywords: Genetic Algorithms, Hub Location Problems, Combinatorial Optimization, Metaheuristics

PREDGOVOR

Rad se sastoji od pet poglavlja. U uvodnom poglavlju date su osnovne informacije o hab lokacijskim problemima i genetskim algoritmima. U drugom poglavlju je opisan hab lokacijski problem neograničenih kapaciteta sa višestrukim alokacijama (formulacija, primer, postojeći načini rešavanja). Zatim, u narednom poglavlju je predstavljen predloženi hibridni genetski algoritam za rešavanje datog problema. Četvrto poglavlje sadrži eksperimentalne rezultate genetskog algoritma. Pregled naučnog doprinosa rada i dobijenih rezultata sa zaključkom je dat u petom poglavlju.

Želela bih da se zahvalim:

Mentoru doc. dr Zorici Stanimirović i komentoru dr Jozefu Kratici na rukovođenju pri izradi ovog rada. Posebno zahvalnost dugujem komentoru Jozefu Kratici koji me je zainteresovao za genetske algoritme.

Članu komisije doc. dr Vladimiru Filipović na vrlo pažljivom čitanju rukopisa i korisnim sugestijama koje su doprinele kvalitetu rada.

Kolegi Denisu Ćorić na pomoći pri obradi teksta i crteža i na bezgraničnom razumevanju.

Prijateljima Danici Baranin, Dragani Djukić-Lazović, Tamari Ćorić i Maji Mrkić na velikoj podršci.

I naravno, najviše svojim roditeljima i sestri Ivani na velikom razumevanju i koji su mi rad učinilli neuporedivo lakšim.

Beograd, 2010.

Kandidat

Maja Đukić

1. UVOD

1.1 Lokacijski problemi

Lokacijski problemi predstavljaju posebnu klasu zadataka mininizacije, kod kojih se najčešće zahteva minimizacija rastojanja, ukupnog vremena putovanja ili nekog drugog parametra. Ovi problemi privlače veliko interesovanje i veoma često su predmet istraživanja. Lokacijski problemi, u širem smislu, se odnose na određivanje pozicije jednog ili grupe objekata u prostoru određene dimenzionalnosti. U užem smislu, ovi problemi se odnose na lociranje resursa, skladišnih objekata, terminala, pretovarnih mesta... Objekti za čije se lociranje traži mesto obično su neka vrsta centara, koji pružaju usluge, pa se često nazivaju *snabdevači*, dok se korisnici usluga (objekti koji su već postavljeni) nazivaju *korisnici.*

Pri klasifikovanju lokacijskih problema relevantne su sledeće karakteristike:

- broj objekata koje treba postaviti
- da li je objekat poželjan ili nepoželjan
- metrika koja se koristi za računanje rastojanja između dva objekta
- vrsta prostora u kojem se traži pozicija za objekat (mreža, ravan itd.)
- kapacitet traženih objekata (neograničen ili ograničen)
- oblik funkcije cilja itd.

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), $i = 1, \dots, n$ koordinate n fiksiranih tačaka (lokacije potrošača), w_i, $i = 1, \dots, n$ 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].

Hab lokacijski problemi (*hub location problems*) su postali veoma važna oblast teorije lokacije u poslednjih dvadeset godina. Ovo je uzrokovano u velikoj meri zbog korišćenja mreže habova (*hub networks*) u modernim transportnim i telekomunikacijskim sistemima. Ovi problemi uglavnom spadaju u NP-teške probleme [Gar79], [Brs88], [Cre97], [Jun98], [Dre02]. NP-teški problemi su problemi koji su nerešivi egzaktnim metodama. Ove metode najčešće daju loše rezultate kada se primene na probleme velikih dimenzija. Međutim, iako su rezultati dobijeni primenom heurističkih metoda dosta nepouzdaniji (često možemo dobiti optimalno rešenje, ali ne uvek), NP-teške probleme je bolje rešavati njima.

1.2 Genetski algoritmi

1.2.1 Opšte karakteristike genetskih algoritama

Klasične heuristike razvijane su u cilju rešavanja pojedinačnih problema. Međutim, nastale su nove „moderne" heuristike tj. metaheuristike koje se mogu

primeniti na veliki broj zadataka iz različitih oblasti. Genetski algoritmi su jedni od najpoznatijih metaheuristika.

Prve ideje o genetskim algoritmima izložene su u radu *J.Holland-a* 1975. godine [HII75], i javile su se u okviru tzv. teorije adaptivnih sistema, koja proučava modele efikasnog adaptivnog ponašanja nekih bioloških, specijalno genetskih, sistema. Ovakvi algoritmi su prvobitno kreirani da simuliraju proces genetske evolucije jedne populacije jedinki pod dejstvom okruženja i genetskih operatora. One jedinke iz populacije koje su u većoj meri prilagođene okruženju, međusobno se dalje reprodukuju i tako stvara nova generacija jedinki. Ovaj proces se ponavlja pri čemu se iz generacije u generaciju prosečna prilagođenost članova populacije povećava. Nakon nekog broja generacija čitav postupak se zaustavlja do zadovoljenja jednog ili više kriterijuma zaustavljanja. Najbolji član trenutne populacije predstavlja rešenje genetskog algoritma.

Danas se genetski algoritmi koriste za rešavanje široke klase problema kombinatorne optimizacije. Postoji veliki broj radova o genetskim algoritmima. Neki od njih su: [Djo75], [Gol89], [BeD93a], [BeD93b], [Yur94], [Mic96], [Mit96] i [Müh97]. O genetskim algoritmima se može naći i u domaćoj literaturi: [Čan96], [Kra00], [Fil98], [Tošo4].

Svaka jedinka u populaciji je predstavljena genetskim kodom nad određenom konačnom azbukom. Najčešće se koristi binarno kodiranje, gde se genetski kod sastoji od niza bitova. Binarno kodiranje je najpogodnije za implementaciju zbog svoje jednostavnosti, a nekada je pogodno koristiti azbuke veće kardinalnosti. Kodiranje rešenja je bitan korak genetskog algoritma jer se neadekvatnim izborom koda može doći do loših rezultata bez obzira na ostalu strukturu genetskog algoritma.

Početna populacija je obično slučajno generisana, što obezbeđuje raznovrsnost genetskog materijala. Svakoj jedinki populacije (u praksi ih je najviše do nekoliko stotina) se dodeljuje *funkcija prilagođenosti* ili *funkcija cilja* (*fitness function*) koja određuje kvalitet date jedinke tj. odgovarajućeg rešenja. Zatim se primenjuje *operator selekcije* – uzimaju se bolje jedinke, po nekom postupku odabira i formira nova generacija. Nakon toga neki članovi populacije su podvrgnuti uticajima genetskih operatora: *ukrštanja* i *mutacije*.

Uloga genetskog operatora selekcije je čuvanje i prenošenje dobrih svojstava na sledeću generaciju jedinki. Selekcijom se odabiraju dobre jedinke koje će učestvovati u reprodukciji i ona se primenjuje u skladu sa vrednostima funkcije prilagođenosti. Na taj način se dobri geni ili dobar genetski materijal čuva i prenosi na sledeću generaciju, a loš odumire.

Genetski operator ukrštanja se definiše u najopštijem slučaju kao postupak u kome učestvuju dve ili više jedinki koje nazivamo *roditeljima*. Njihovim ukrštanjem nastaje jedna ili više jedinki koje nazivamo *potomcima*.

Genetski operator mutacije se primenjuje na svaku jedinku (potomak) sa unapred zadatom verovatnoćom mutacije p_{mut}. Verovatnoća p_{mut} treba da bude veoma mala (obično reda 10^{-3}). Ovaj operator se koristi da bi se povremeno unela nova raznovrsnost među jedinke jedne populacije posebno u slučaju kada su one međusobno veoma slične.

Operatori genetskog algoritma se primenjuju uzastopno sve dok ne bude zadovoljen jedan ili više kriterijuma zaustavljanja: maksimalan broj generacija, dostignut optimum itd.

Na slici 1.1 je dat shematski zapis osnovnih elemenata genetskog algoritma:

```
Unošenje_Ulaznih_Podataka();
Generisanje_Početne_Populacije();
while(!Kriterijum_Zaustavljanja_Genetskog_Algoritma())
{
     for (i=1;i<=N_populacije;i++ )
          obj[i]=Funkcija_Cilja(i);
     Funkcija_Prilagođenosti();
     Selekcija();
     Ukrštanje();
     Mutacija();
}
Štampanje_Izlaznih_Podataka();
```

Slika 1.1 Shematski zapis genetskog algoritma

1.2.2 Prost genetski algoritam

Najjednostavnijji koncept genetskog algoritma je prost genetski algoritam *(simple genetic algorithm).* Sastoji se od proste rulet selekcije, jednopozicionog ukrštanja i proste mutacije. Ova verzija genetskog algoritma može se naći u *Holland-ovoj* knjizi [HII75], a osobine navedenih genetskih operatora su opisane u narednom odeljku.

Kod prostog genetskog algoritma može doći do *preuranjene konvergencije.* Ona se javlja često kao posledica primene proste rulet selekcije. Preuranjena konvergencija se dešava ukoliko jedna ili nekoliko relativno dobrih jedinki, ali ne i optimalnih, preovlada u populaciji i proces konvergira u lokalnom ekstremu. Visoko prilagođene jedinke će vremenom istisnuti lošije jedinke iz populacije, iako one možda sadrže dobre gene koji mogu dati kvalitetnije potomke. Na taj način dolazi do gubitka genetskog materijala koji ni mutacija ne može rešiti u praksi.

Postoji još jedan nedostatak prostog genetskog algoritma - *spora konvergencija.* Ona se javlja obično u završnoj fazi genetskog algoritma. Nekad se desi 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 genetskog algoritma

Prost genetski algoritam zbog svojih nedostataka ne može se uspešno primeniti na složenije probleme kombinatorne optimizacije. Razvijeniji koncept genetskog algoritma sadrži razne vrste kodiranja, razne funkcije prilagođenosti, usavršene operatore selekcije, ukrštanja i mutacije.

1.2.3.1 Kodiranje i funkcija prilagođenosti

U praksi se pokazalo da je kod genetskih algoritama uglavnom najpogodnije binarno kodiranje koje svakom rešenju dodeljuje kod unapred zadate dužine nad azbukom {0,1}, a nekad možemo kodirati nad azbukama veće kardinalnosti (na primer, 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].

Što se tiče funkcija prilagođenosti, postoje različiti načini računanja: *direktno preuzimanje, linearno skaliranje, skaliranje u jedinični interval, sigma odsecanje* ([Kra00]). U zavisnosti od vrste problema jedan od ovih načina koristimo, a takođe je moguće i njihovo kombinovanje.

Postoji mogućnost da se tokom rada genetskog algoritma generišu jedinke koje ne odgovaraju nijednom rešenju i te jedinke nazivamo *nekorektnim.* Nekorektne genetske kodove možemo ignorisati i dodeliti im nulu za vrednost funkcije prilagođenosti. U narednoj generaciji oni će biti automatski izbačeni iz populacije. Tako da u populaciji ostaju samo korektni genetski kodovi. Ovaj postupak se u praksi može primeniti samo na slučajeve gde ima najviše $50 - 70\%$ nekorektnih jedinki (kodova). Moguće je i uključivanje nekorektnih kodova u genetski algoritam. Pri tome računamo vrednost svake jedinke (korektne ili nekorektne) u populaciji. Za svaku nekorektnu jedinku računamo kaznenu funkciju prilagođenosti. Na osnovu datih vrednosti, računamo prilagođenost svake jedinke i dalje nastavljamo klasični genetski algoritam.

Nekorektne jedinke možemo „popraviti" do korektnih primenom neke metode [Kra07]. Pri tome za vrednost nekorektne jedinke preuzimamo vrednost korektne jedinke. Više o načinima rešavanja problema nekorektnih jedinki može se naći u [Lvi93a], [SmA93] i [Orv93].

1.2.3.2 Selekcija

Da bi izbegli problem preuranjene i spore konvergencije možemo koristiti *selekciju zasnovanu na rangiranju (rank based selection)* genetskih kodova prema njihovim prilagođenostima. Funkcija prilagođenosti jedinke jednaka je nekom rangu iz unapred zadatog niza rangova. Prema tome, ona zavisi samo od poretka jedinki u populaciji.

Jedan od boljih operatora selekcije je *turnirska selekcija (tournament selection).* Broj jedinki koje učestvuju na turniru je N_{tour} i on predstavlja parametar turnirske selekcije. Ovaj broj se najčešće zadaje unapred. Ova selekcija je takva da se prvo na slučajan način biraju podskupovi od po N_{tour} jedinki. U svakom od skupova jedinki se simulira „turnir" i u narednu generaciju prolazi najbolja jedinka, odnosno jedinka sa najboljom funkcijom prilagođenosti. Najčešći problem kod ove selekcije je izabrati

N_{tour} tako da se smanjuju nepovoljni stohastički efekti, kako bi što kvalitetniji genetski materijal prošao u sledeću generaciju. U nekim slučajevima je poželjnije da veličina turnira ne bude ceo broj i tada se uspešno pokazala *fino-gradirana turnirska selekcija.* Opis ove i ostalih oblika selekcija mogu se naći u [Fil98]. Primena fino-gradirane turnirske selekcije i njeno poređenje sa drugim oblicima selekcije dati su u [Fil00], [Fil01], [Fil03], [Fil06].

1.2.3.3 Ukrštanje

Ukrštanjem vršimo razmenu genetskog materijala jedinki (roditelja) i tako dobijamo nove jedinke (potomke). Može se očekivati da će ovako dobijeni potomci zadržati „dobre" osobine svojih roditelja. Ponekad ukrštanjem kombinujemo i loše gene kvalitetnijih roditelja i tako nastaju lošiji potomci u odnosu na svoje roditelje.

U prostom genetskom algoritmu primenjuje se *jednopoziciono ukrštanje*, gde se na slučajan uniforman način bira broj $k \in \{0,1,\ldots,N-1\}$ koji predstavlja tačku ukrštanja, tj. poziciju u kodovima roditelja u odnosu na koju će se razmeniti njihov sadržaj (N je dužina genetskog koda). Zatim, svi simboli u ova dva koda, od pozicije $k+1$ do zadnje pozicije $N-1$, uzajamno razmenjuju mesta (u tabeli 1.1 je dato $k=3$ i $N=8$). Operator ukrštanja može biti definisan i u odnosu na više tačaka ukrštanja. Tako, na primer, kod *dvopozicionog ukrštanja*, slučajno se biraju dve pozicije k_1 i $k_2 \in \{0,1,\ldots,N-1\}$ i uzajamno se razmenjuju delovi kodova roditelja od pozicije k_1+1 do pozicije k_2 (u tabeli 1.1 je dato $k_1=2$, $k_2=5$ i $N=8$).

Tabela 1.1 Operator jednopozicionog i dvopozicionog ukrštanja

broj tačaka ukrštanja	roditelji	potomci
1	**1001|1010** **1100|1111**	**1001|1111** **1100|1010**
2	**010|101|11** **110|111|00**	**010|111|11** **110|101|00**

Postoji i *višepoziciono* ili *uniformno* ukrštanje. Svaki roditeljski par generiše *„masku*", tj. binarni niz iste dužine kao genetski kod roditelja. Roditelji razmenjuju gene na svim pozicijama na kojima maska ima vrednost 0, a na mestima gde je vrednost 1 roditelji zadržavaju svoje gene (videti sliku 1.2).

maska: 10110011

pre ukrštanja	posle ukrštanja
XXXXXXXX	XYXXYYXX
YYYYYYYY	YXYYXXYY

Slika 1.2 Operator uniformnog ukrštanja

Više o ovom operatoru može se naći u [Spe91].

1.2.3.4 Mutacija

Mutacija je jedna od najznačajnijih operatora kod genetskih algoritama. Postoji više varijanata ovog operatora. Neki od njih su: *prosta mutacija, mutacija pomoću binomne raspodele, mutacija pomoću normalne raspodele.*

U slučaju kada su jedinke binarno kodirane i populacija nema nekorektnih jedinki najčešće se koristi operator proste mutacije. Prosta mutacija podrazumeva izmenu određenog gena u genetskom kodu jedinke. Nivo mutacije p_{mut} je unapred zadat i predstavlja verovatnoću sa kojom se svaki bit mutira. Nekad se prosta mutacija realizuje i preko „maske“.

01**0**1010101
↓
01**1**1010100

Slika 1.3 Operator proste mutacije

U tabeli 1.1 imamo četiri roditeljska koda. Na poziciji broj 2 svakog koda je vrednost 0 i da nema mutacije u sledećim generacijama ne bi mogla da se povrati vrednost 1 na toj poziciji nikakvom selekcijom niti ukrštanjem.

Realizacija operatora proste mutacije može se ubrzati pomoću binomne ili normalne raspodele. Pri primeni *mutacije pomoću binomne raspodele* određuje se broj X_{mut} - broj mutiranih gena jedinke. Broj X_{mut} je slučajna veličina koja ima binomnu raspodelu $B(N, p_{mut})$, gde je N dužina genetskog koda, a p_{mut} je nivo mutacije. Neka je F njena funkcija raspodele. Zatim, biramo slučajan broj $s \in [0,1]$ i pronalazimo X_{mut} tako da važi $F(X_{mut}) \leq F(s) < F(X_{mut}+1)$. Na kraju se na slučajan način bira pozicija u genetskom kodu na kome se vrši mutacija.

Ukoliko je proizvod dužine genetskog koda N i nivoa mutacije p_{mut} dovoljno veliki pogodno je pomenutu binomnu raspodelu koju ima slučajna veličina X_{mut}

aproksimirati normalnom raspodelom $N(N \cdot p_{mut}, N \cdot p_{mut} \cdot (1 - p_{mut}))$. Više o osobinama navedenih mutacija može se naći u [Kra00], [Toš04].

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. Najprimenjivaniji kriterijumi zaustavljanja genetskog algoritma su:

- Dostignuti maksimalni broj generacija
- Sličnost jedinki u populaciji
- Ponavljanje najbolje jedinke određeni (maksimalni) broj puta
- Dostizanje optimalnog rešenja ako je ono unapred poznato
- Dokazana optimalnost najbolje jedinke (ukoliko je to moguće)
- Ograničeno vreme izvršavanja genetskog algoritma
- Prekid od strane korisnika itd.

Svaki od navedenih kriterijuma ima dobre i loše strane, tako da se u praksi najbolje pokazalo njihovo kombinovanje. Korišćenjem više kriterijuma zaustavljanja smanjuje se mogućnost loše procene prekida genetskog algoritma.

1.2.3.6 Ostali aspekti genetskog algoritma

Jedan od važnijih aspekata genetskog algoritma je *politika zamene generacija*. Najčešće se primenjuju sledeće strategije:

a. *Generacijska (generational)*
b. *Stacionarna (steady- state)*
c. *Elitistička strategija (elitist strategy)*

Generacijski genetski algoritam u svakoj generaciji vrši promenu čitave populacije novim jedinkama (potomcima).

Stacionarni genetski algoritam generiše samo deo populacije u svakoj generaciji, a preostale jedinke se prenose iz prethodne generacije.

Elitistička strategija zamene generacija omogućava da jedna ili više najboljih (elitnih) jedinki prođe direktno u narednu generaciju, bez primene genetskih operatora selekcije, ukrštanja i mutacije i bez računanja funkcije prilagođenosti. Ova strategija je bolja od prethodne dve, mada je moguće i njihovo kombinovanje.

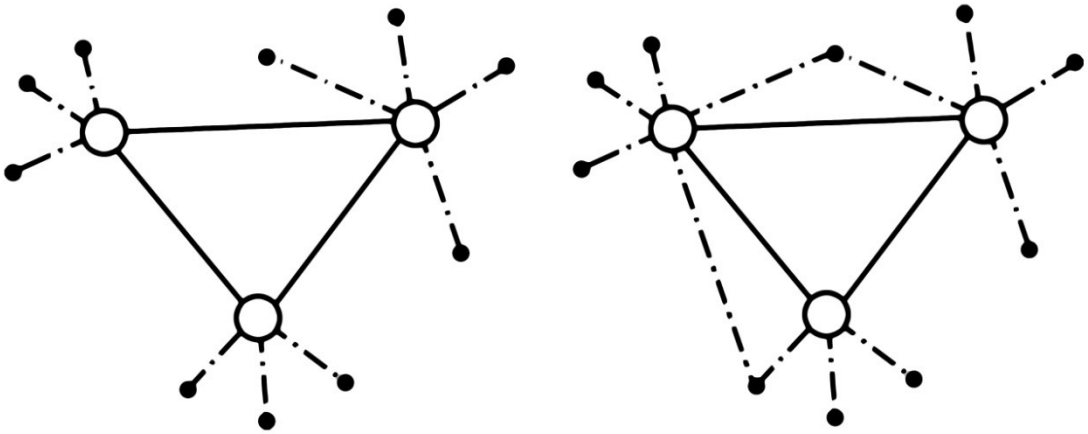
Genetski algoritmi se mogu kombinovati sa drugim heuristikama. Heuristike (jedna ili više) se mogu koristiti ne samo za poboljšavanje početne populacije, već i svake naredne generacije. Možemo ih primenjivati na celu populaciju, jedan njen deo ili na pojedinačnu (često najbolju) jedinku. Paralelizacijom i keširanjem se, takođe, bitno poboljšavaju performanse genetskog algoritma (videti u [Kra00]) .

2. HAB LOKACIJSKI PROBLEM NEOGRANIČENIH KAPACITETA SA VIŠESTRUKIM ALOKACIJAMA

Hab lokacijski problemi se koriste za dizajniranje transportnih mreža (železnički i drumski sistem, poštanski sistemi, prevoženje putnika i robe u avio saobraćaju, sistem brze isporuke itd.). Protok između čvorova u mreži možemo definisati kao broj putnika ili količinu robe koju treba transportovati od početnog čvora (snabdevača) do krajnjeg čvora (korisnika). Transport između ova dva čvora se ne uspostavlja direktno, već je usmeren preko skupa čvorova koji su označeni kao *habovi*. Habovi se koriste kao transportni terminali, centri preusmeravanja, selekcije ili sakupljanja robe ili putnika. Koristeći habove kao tačke preusmeravanja protoka i povećavajući transport između habova, kapacitet mreže se može iskoristiti dosta efikasnije. Cena transporta između habova po jedinici količine je niža, pa su i ukupni troškovi transporta u mreži znatno manji. Hab–mrežama možemo transportovati (prenositi) i informacije u telekomunikacijskim sistemima preko odgovarajućih linkova (telefonskim ili optičkim kablovima, satelitskim kanalima). Da bi smanjili cenu protoka podataka u mrežama koriste se takođe 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 (videti sliku 2.1 a)).
- *šema višestruke alokacije (multiple allocaton scheme)*, koja dopušta svakom ne - hab čvoru da komunicira sa jednim ili više habova (videti sliku 2.1 b)).



Slika 2.1 Primer hab-mreže a) sa jednostrukim b) sa višestrukim alokacijama

Na slici 2.1 ne-hab čvorovi su označeni tačkama, a habovi krugovima. Pregled hab lokacijskih problema i njihova klasifikacija dati su u [Cam96], [Cam02].

2.1 Matematička formulacija

Neka je $I = \{1, \ldots, n\}$ 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}. U formulaciji problema promenljive su korišćene na sledeći način:

- Binarna promenljiva y_k uzima vrednost 1 ako je hab lociran na $k-$tom čvoru, u suprotnom ima vrednost 0.
- x_{ijkm} je količina protoka koja polazi iz čvora i koji se sakuplja u habu k, distribuira se preko haba m i prosleđuje u čvor j.

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. Troškovi uspostavljanja snabdevača preko haba k su označeni sa f_k. Koristeći gornju

notaciju, UMAHLP (*Uncapacitated Multiple Allocation Hub Location Problem*) matematički se može zapisati kao (videti u [CGM07]):

$$\min \sum_{i,j,k,m} W_{ij}(\chi C_{ik} + \alpha C_{km} + \delta C_{mj}) x_{ijkm} + \sum_{k} f_k y_k \tag{2.1}$$

Uz uslove:

$$\sum_{k,m} x_{ijkm} = 1 \quad \text{, za svako } i,j \tag{2.2}$$

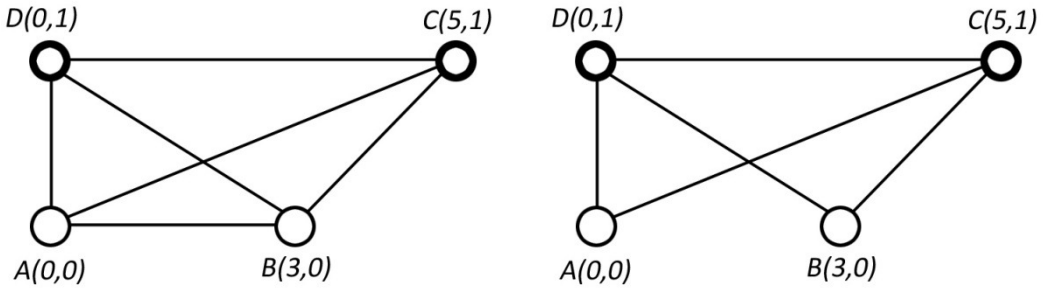
$$\sum_{m} x_{ijkm} + \sum_{m,m\neq k} x_{ijmk} \leq y_k \quad \text{, za svako } i,j,k \tag{2.3}$$

$$y_k \in \{0,1\} \quad \text{, za svako } k \tag{2.4}$$

$$x_{ijkm} \geq 0 \quad \text{, za svako } i,j,k,m \tag{2.5}$$

Funkcija cilja (2.1) minimizuje sumu transportnih troškova snabdevač-hab, hab-hab i hab-korisnik, pomnoženih sa koeficijentima χ, α i δ respektivno i na tu sumu se dodaju fiksni troškovi f_k za svaki uspostavljeni hab (tamo gde je $y_k = 1$). Ograničenje (2.2) označava da se celokupni protok kreće kroz sve parove čvorova, dok ograničenje (2.3) osigurava da se protok kreće samo kroz otvorene habove. Ograničenja (2.4) i (2.5) označavaju binarnu, odnosno ne-negativnu reprezentaciju promenljivih y_k i x_{ijkm} respektivno.

2.2 Primer za UMAHLP



Slika 2.2 Hab-mreža (sa višestrukim alokacijama)

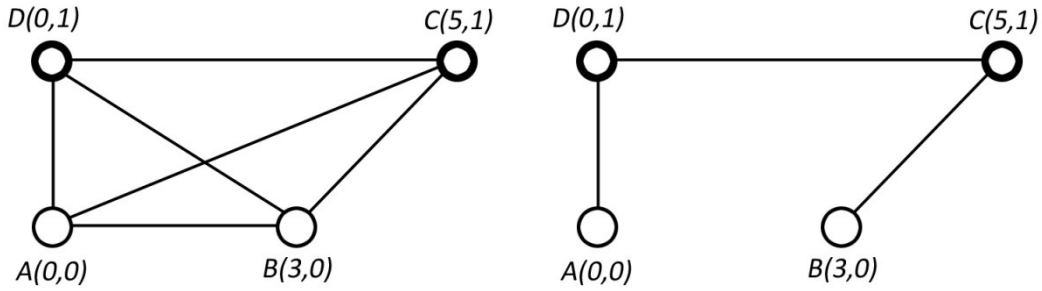
Na levoj strani slike 2.2 je data mreža sa četiri čvora A, B, C i D (čvorovi su zadati Dekartovim koordinatama u ravni). Odgovarajuća matrica rastojanja između parova čvorova u mreži je:

$$\begin{bmatrix} d(A,A) & d(A,B) & d(A,C) & d(A,D) \\ d(B,A) & d(B,B) & d(B,C) & d(B,D) \\ d(C,A) & d(C,B) & d(C,C) & d(C,D) \\ d(D,A) & d(D,B) & d(D,C) & d(D,D) \end{bmatrix} = \begin{bmatrix} 0 & 3 & \sqrt{26} & 1 \\ 3 & 0 & \sqrt{5} & \sqrt{10} \\ \sqrt{26} & \sqrt{5} & 0 & 5 \\ 1 & \sqrt{10} & 5 & 0 \end{bmatrix}$$

Neka je količina protoka od čvora-snabdevača do čvora-korisnika ista i jednaka 1. Vrednosti parametara su: $\chi = 1$, $\delta = 1$ i $\alpha = 0.5$. Fiksni troškovi uspostavljanja habova u čvorovima $A, B, \boldsymbol{C}, \boldsymbol{D}$ su redom $25, 45, 20, 55$. Na desnoj strani slike 2.2 možemo videti da su habovi su uspostavljeni u čvorovima $\boldsymbol{C}$ i $\boldsymbol{D}$, a ne-hab čvorovi A i B su pridruženi habovima $\boldsymbol{C}$ i $\boldsymbol{D}$. Svaki od habova $\boldsymbol{C}$ i $\boldsymbol{D}$ je pridružen samom sebi. Kod šeme višestruke alokacije ne-hab čvorovi A i B mogu komunicirati i preko haba $\boldsymbol{C}$ i preko haba $\boldsymbol{D}$ u zavisnosti od toga koja je cena transporta povoljnija. Troškovi transporta između svakog para čvorova u mreži su: „$A - \boldsymbol{C} - A$“ $\sqrt{26} \cdot 1 + \sqrt{26} \cdot 1 = 10.1980$, „$\underline{A - \boldsymbol{D} - A}$“ $1 \cdot 1 + 1 \cdot 1 = 2$, „$A - \boldsymbol{C} - B$“ $\sqrt{26} + \sqrt{5} = 7.3351$, „$\underline{A - \boldsymbol{D} - B}$“ $1 + \sqrt{10} = 4.1623$, „$A - \boldsymbol{C}$“ $\sqrt{26} = 5.0990$, „$\underline{A - \boldsymbol{D} - \boldsymbol{C}}$“ $1 + 0.5 * 5 = 3.5000$, „$\underline{A - \boldsymbol{D}}$“ 1, „$A - \boldsymbol{C} - \boldsymbol{D}$“ $\sqrt{26} + 0.5 * 5 = 7.5990$, „$\underline{B - \boldsymbol{C} - B}$“ $\sqrt{5} + \sqrt{5} = 4.4721$, „$B - \boldsymbol{D} - B$“ $\sqrt{10} + \sqrt{10} = 6.3245$, „$\underline{B - \boldsymbol{C}}$“ $\sqrt{5} = 2.2361$, „$B - \boldsymbol{D} - \boldsymbol{C}$“ $\sqrt{10} + 0.5 * 5 = 5.6623$, „$\underline{B - \boldsymbol{D}}$“ $\sqrt{10} = 3.1623$, „$B - \boldsymbol{C} - \boldsymbol{D}$“ $\sqrt{5} + 0.5 * 5 = 4.7361$, „$\underline{\boldsymbol{C} - \boldsymbol{C}}$“ 0, „$\underline{\boldsymbol{C} - \boldsymbol{D}}$“ $5 * 0.5 = 2.5000$, „$\underline{\boldsymbol{D} - \boldsymbol{D}}$“ 0. Za transport robe od čvora-snabdevača A do čvora-korisnika A biramo hab $\boldsymbol{D}$, jer su troškovi „$A - \boldsymbol{D} - A$“ 2 manji od transporta „$A - \boldsymbol{C} - A$“ 10.1980. Slično biramo i kod ostalih parova čvorova u mreži i povoljnije cene transporta su podvučene. Vrednost funkcije cilja (minimalni troškovi transporta sa fiksnim troškovima uspostavljanja habova) je:

$$(2 + 4.1623 + 3.5000 + 1 + 4.1623 + 4.4721 + 2.2361 + 3.1623 + 3.500 + 2.2361 + 0 + 2.5000 + 1 + 3.1623 + 2.5000 + 0) + (20 + 55) = 114.5935$$

U slučaju šeme jednostruke alokacije (videti na desnoj strani slike 2.3) uspostavljeni su habovi u čvorovima $\boldsymbol{C}$ i $\boldsymbol{D}$, ali svaki ne-hab čvor komunicira preko tačno jednog, njemu pridruženog haba (čvor A preko haba $\boldsymbol{D}$, a čvor B preko haba $\boldsymbol{C}$).



Slika 2.3 Hab-mreža (sa jednostrukim alokacijama)

Ako su nam vrednosti parametara kao u slučaju šeme višestruke alokacije, tada će troškovi transporta između svakog para čvorova u mreži sa jednostrukim alokacijama biti: „$A - \boldsymbol{D} - A$“ $1 + 1 = 2$, „$A - \boldsymbol{D} - \boldsymbol{C} - B$“ $1 + 0.5 * 5 + \sqrt{5} = 5.7361$, „$A - \boldsymbol{D} - \boldsymbol{C}$“ $1 + 0.5 * 5 = 3.5000$, „$A - \boldsymbol{D}$“ 1, „$B - \boldsymbol{C} - B$“ $\sqrt{5} + \sqrt{5} = 4.4721$, „$B - \boldsymbol{C}$“ $\sqrt{5} = 2.2361$, „$B - \boldsymbol{C} - \boldsymbol{D}$“ $\sqrt{5} + 0.5 * 5 = 4.7361$, „$\boldsymbol{C} - \boldsymbol{C}$“ 0, „$\boldsymbol{C} - \boldsymbol{D}$“ $0.5 * 5 = 2.5000$, „$\boldsymbol{D} - \boldsymbol{D}$“ 0. Vrednost funkcije cilja je:

$$(2 + 5.7361 + 3.5000 + 1 + 5.7361 + 4.4721 + 2.2361 + 4.7361 + 3.500 + 2.2361 + 0 + 2.5000 + 1 + 4.7361 + 2.5000 + 0) + (20 + 55) = 120.8887$$

Znači, minimalni troškovi transporta sa fiksnim troškovima kod šeme višestruke alokacije se ostvaruju za 114.5935, a kod šeme jednostruke alokacije za 120.8887. Ušteda kod višestruke alokacije iznosi oko 5.21%.

2.3 Postojeći načini rešavanja

Postoji nekoliko pristupa u literaturi koji razmatraju UMAHLP. Problem je prvi put formulisan u [Cam94]. Tehnika dualnog penjanja unutar metode grananja i ograničavanja (*Branch-and-Bound scheme*) je prvi put primenjena za rešavanje UMAHLP u [Kli96] na ORLIB (*Operation Research Library-Standardna kolekcija instanci za operaciona istraživanja*) (videti u [Bea96]) hab instancama sa brojem čvorova $n \leq 25$. Sličan pristup je korišćen u [May02]. Rezultati su prikazani na primeru koji je imao do 40 čvorova.

Nova kvadratna formulacija problema, bazirana na ideji lanca snabdevanja, je predstavljena u [AbH98b]. Ova nova formulacija se pokazala kao pogodna za korišćenje metode grananja i ograničavanja. Autori su pokazali primer sa slučajno generisanim čvorovima gde je $n \leq 80$.

U [Bol04] su korišćene mešovite linearne i celobrojne programske formulacije (MILP-*Mixed Integer Linear Programming*) za tri višestruke alokacije hab lokacijskih problema, uključujući i UMAHLP. Za svaki problem su razvijane pretprocesorske procedure i određivane preciznije granice. Ovaj pristup je testiran na standardnim ORLIB AP (*Australian Post*) instancama problema sa najviše 50 čvorova.

Glavna ideja u [MCL06] je da učvrsti MILP formulaciju i da smanji broj uslova koristeći rezultate na polju poliedarske strukture problema pakovanja grupa. Druga ideja, koja je prikazana u [CGM07], je razmatrala problem dualnosti MILP formulacije. Autori su prvo konstruisali heuristički metod, baziran na tehnici dualnog penjanja, koji daje skoro 70% optimalnih rešenja na ORLIB instancama do 120 čvorova. Ta heuristika je kasnije ubačena u *Branch-and-Bound* algoritam koji je davao optimalna rešenja u svim slučajevima.

UMAHLP je NP-kompletan, 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 $O(n^3)$ vremenu izvršavanja ([Ern98a]).

3. PREDLOŽENI HIBRIDNI GENETSKI ALGORITAM

3.1 Reprezentacija i funkcija cilja

U implementaciji genetskog algoritma jedinke su binarno kodiranje. Svako rešenje je predstavljeno binarnim stringom dužine N. U genetskom kodu cifra 1 označava da je uspostavljen hab na toj lokaciji,a cifra 0 da nije uspostavljen. Pošto snabdevač i korisnik mogu biti pridruženi samo uspostavljenim hab-lokacijama, iz genetskog koda se dobija niz (y_k). Takođe, ne postoje kapaciteti, tako da vrednosti x_{ijkm} mogu biti izračunate za vreme ocene funkcije cilja.

Primer 3.1: Neka je jedinka predstavljena binarnim stringom dužine $N = 8$ (**01011001)**. Habovi su uspostavljeni na prvoj, trećoj, četvrtoj i sedmoj poziciji.

Primer 3.2: U primeru 2.2 imamo genetski kod dužine $N = 4$ $(AB\boldsymbol{CD})$. Habovi su uspostavljeni na pozicijama u oznaci $\boldsymbol{C}$ i $\boldsymbol{D}$.

Za fiksiran skup habova (y_k) koristi se modifikacija poznatog *Floyd-Warshall* algoritma najkraćeg puta, opisanog u [Ern98a]. Posle nalaženja najkraćih puteva za svaki par čvorova u mreži, lako se računa funkcija cilja. To se postiže jednostavnim sumiranjem najkraćih rastojanja snabdevač-hab, hab-hab i hab-korisnik pomnoženih odgovarajućim parametrima χ, α i δ.

3.2 Princip lokalnog pretraživanja

Princip lokalnog pretraživanja, primenjen za rešavanje najopštijeg oblika problema kombinatorne optimizacije $\min_{x \in X} f(x)$, podrazumeva definisanje neke *strukture okolina* u prostoru dopustivih rešenja X u okviru koje se svakom $x \in X$ pridružuje neki podskup $N(x) \subseteq X, x \notin N(x)$, koji se naziva *okolinom* dopustivog

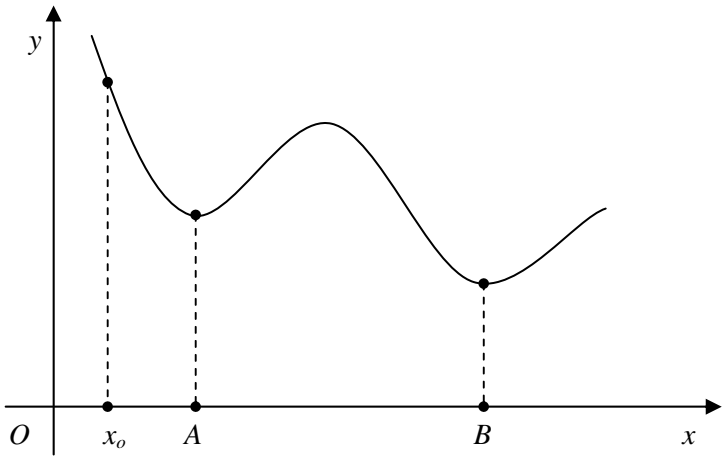
rešenja x, a njeni članovi su *susedi* od x. U lokalnom pretraživanju se polazi od proizvoljne tačke iz X kao od početnog rešenja, pa se u svakoj iteraciji pretražuje okolina trenutnog rešenja i u njoj nalazi, prema nekom definisanom *pravilu izbora,* sused koji predstavlja sledeće rešenje. Ako se takav sused ne može naći, pretraživanje staje i za aproksimaciju optimalnog rešenja problema $\min_{x\in x} f(x)$ uzima se ono među generisanim rešenjima za koje je vrednost funkcije cilja $f(x)$ najmanja.

Jasno je da efikasnost jedne konkretne metode koja se bazira na prethodnom principu,tj. stepen njene šanse da u razumnom vremenu dobije rešenje dovoljno blisko optimalnom, zavisi, pre svega, od načina definisanja pojma „okolina“, kao i od toga kako se formuliše pravilo za izbor sledećeg rešenja u svakom koraku metode.

Kod diskretnog dopustivog skupa X okolina $N(x)$ nekog dopustivog rešenja x može se u opštem slučaju definisati kao skup svih onih elemenata y iz X koji mogu biti dobijeni direktno iz x primenom neke, unapred zadate, modifikacije nazvane *pomak* iz x u y, označen kao $m(x, y)$. Da bi neka struktura okolina obezbedila efikasnost lokalnog pretraživanja, intuitivno je jasno da, pri određivanju ove modifikacije, treba težiti da budu zadovoljeni sledeći uslovi:

- Okolina svake tačke iz X treba da bude *simetrična, tj.* $y \in N(x)$ akko $x \in N(y)$;
- Struktura okolina treba da zadovoljava uslov *dostižnosti,* tj. da se, polazeći od bilo koje tačke prostora X, može nizom uzastopnih pomaka doći do bilo koje druge ovog prostora;
- Pomak treba da omogući što jednostavnije i brže generisanje susednih rešenja, tj. preporučljivo je da ova modifikacija bude polinomijalne složenosti;
- Okolina ne treba da bude ni suviše velika ni suviše mala, da bi se, s jedne strane, mogla lakše odgovarajuće pretražiti, a s druge strane da bi pružila dovoljno mogućnosti da se nađe sledeća tačka pretraživanja.

Pravila izbora sledeće tačke x_{n+1} u procesu lokalnog pretraživanja mogu se definisati na različite načine. Tako se kod klasičnih *metoda „spuštanja“* dozvoljavaju samo *„silazeći“* pomaci, tj. pomaci u susede koji poboljšavaju funkciju cilja, jer se x_{n+1} bira tako da je $f(x_{n+1}) < f(x_n)$. Međutim, glavni nedostatak pri primeni metoda spuštanja na probleme nalaženja globalnih minimuma (u koje spada i problem $\min_{x\in X} f(x)$), je što se često u procesu pretraživanja ne mogu izbeći zamke lokalnih minimuma. Naime, pretraživanje se može „zaglaviti“ u okolini nekog lokalnog minimuma, iz koje se ne može izvući korišćenjem samo silazećih pomaka. (Na primer, ako kod funkcije $f(x)$, prikazane na slici 3.1, metoda spuštanja pođe od tačke x_0, pretraživanje će se završiti u lokalnom minimumu A i neće moći da dosegne globalni munimum B.)



Slika 3.1 Zaglavljivanje u lokalnom minimumu

3.3 Genetski operatori

3.3.1 Selekcija

Predloženi genetskog algoritma koristi fino-gradiranu turnirsku selekciju, opisanu u [Fil98]. Umesto celobrojnog parametra N_{tour}- veličina turnirske grupe, fino-gradirana turnirska selekcija zavisi od realnog parametra F_{tour}- željena srednja veličina turnira. Operator fino-gradirane turnirske selekcije realizuje dva tipa turnira. Prvi tip je održan k_1 puta i njegova veličina je $[F_{tour}] + 1$. Drugi tip se realizuje k_2 puta sa $[F_{tour}]$ jedinki-učesnika turnira. Pošto je vrednost $F_{tour} = 5.4$ korišćena u ovoj implementaciji genetskog algoritma, odgovarajuće vrednosti k_1 i k_2 (za $N_{tour} = 50$ ne-elitnih jedinki) su 20 i 30 respektivno.

Vreme izvršavanja operatora fino-gradirane turnirske selekcije je $O(N_{tour} * F_{tour})$. U praksi, parametar F_{tour}se koristi kao konstanta (ne zavisi od N_{tour}), što daje $O(N_{tour})$ vremensku složenost. Detaljnije o fino-gradiranoj turnirskoj selekciji može se naći u [Fil98],[Fil00],[Fil03],[Fil06].

3.3.2 Ukrštanje

U implementaciji genetskog algoritma se koristi jednopoziciono ukrštanje. Pri jednopozicionom ukrštanju na početku se bira $N/2$ parova jedinki gde je N veličina populacije. Zatim se za svaki par jedinki, sa unapred zadatom verovatnoćom (nivoom)

ukrštanja, na slučajan način bira tačka ukrštanja i vrši razmena delova genetskih kodova datih parova jedinki posle date pozicije.

Primer 3.3: Neka je dat nivo ukrštanja $p_{cross} = 0.85$ (to znači da oko 85% parova jedinki razmeni genetski materijal) i populacija od pet jedinki 01011, 10001, 00100, 10100, 11001. Neka su izabrani parovi jedinki (prva, peta) i (druga, četvrta), i neka se u oba para vrši ukrštanje. Ako je pozicija za ukrštanje prvog para $k = 1$, a drugog para $k = 3$, pa se tada ukrštanjem dobijaju sledeće nove jedinke:

Prvi par jedinki (prva, peta)		Drugi par jedinki (druga, četvrta)	
Pre ukrštanja	Posle ukrštanja	Pre ukrštanja	Posle ukrštanja
01 | 011	**01001**	**1000 | 1**	**10000**
11 | 001	**11011**	**1010 | 0**	**10101**

Posle ukrštanja populacija će izgledati: **01001, 10000, 00100, 10101, 11011.**

3.3.3 Mutacija

Jedinke-potomci generisane operatorom ukrštanja podležu mutaciji sa zaleđenim bitovima. Operator mutacije se realizuje promenom slučajno izabranog gena u genetskom kodu jedinki (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.

Može se desiti da tokom izvršavanja genetskog algoritma da (skoro) sve jedinke u populaciji imaju isti gen na određenoj poziciji. Takvi geni (može ih biti više) se nazivaju „zaleđeni“ geni. 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 da promene vrednost nijednog zaleđenog bita, a osnovni nivo mutacije je često veoma mali da bi povratili izgubljeni regioni 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. Dakle,u ovoj implementaciji zaleđeni bitovi se mutiraju sa 2.5 puta većom verovatnoćom ($1.0/n$) u odnosu na one koji nisu zaleđeni ($0.4/n$).

Primer 3.4: Neka nam je data populacija od šest jedinki: 01011111, 11010100, 00011010, 01000110, 11001011, 10010101. Izvršićemo ukršanje nad sledećim parovima jedinki: prva i šesta, druga i peta, treća i četvrta i neka je pozicija za ukrštanje prvog para 2, drugog 3, trećeg 4.

Prvi par jedinki (prva i šesta)

Pre ukrštanja	Posle ukrštanja
0 1 0 | 1 1 1 1 1	0 1 **0** 1 0 1 0 1
1 0 0 | 1 0 1 0 1	1 0 **0** 1 1 1 1 1

Drugi par jedinki (druga i peta)

Pre ukrštanja	Posle ukrštanja
1 1 0 1 | 0 1 0 0	1 1 **0** 1 1 0 1 1
1 1 0 0 | 1 0 1 1	1 1 **0** 0 0 1 0 0

Treći par jedinki (treća i četvrta)

Pre ukrštanja	Posle ukrštanja
0 0 0 1 1 | 0 1 0	0 0 **0** 1 1 1 1 0
0 1 0 0 0 | 1 1 0	0 1 **0** 0 0 0 1 0

Dakle, posle ukrštanja roditelja na drugoj poziciji kod svih šest potomaka imamo „zaleđeni“ gen $\mathbf{0}$.

3.3.4 Politika zamene generacija

Početna populacija broji 150 jedinki i ona je slučajno generisana obezbeđujući pri tom maksimalnu raznovrsnost genetskog materijala. U ovom pristupu se primenjuje tzv. *stacionarni genetski algoritam sa elitističkom strategijom* ([Kra00], [Kra01]). Dakle, trećinu populacije ćemo zameniti u svakoj generaciji, a dve trećine tj. 100 jedinki će direktno proći u sledeću generaciju. Funkcije cilja tih 100 najboljih (elitnih) jedinki računamo samo u prvoj generaciji i tako obezbeđujemo uštedu vremena izračunavanja. Njihove fitnes vrednosti se postavljaju na nulu, tako da će operator selekcije izbeći da ih prenese u narednu generaciju. Ovo je veoma efikasna metoda za čuvanje raznolikosti genetskog materijala i sprečava preuranjenu konvergenciju. Jedinke sa istom funkcijom cilja, ali različitim genetskim kodovima mogu nekad dominirati u populaciji. U slučaju da su im i genetski kodovi slični, genetski algoritam se može izvršiti u lokalnom optimumu. Iz tih razloga je korisno ograničiti njihovo pojavljivanje nekom konstantom N_{rv} (to je postavljeno na 40 u ovoj implementaciji genetskog algoritma).

3.3.5 Keširanje genetskog algoritma

Keširanje ima veliki značaj jer omogućava uštedu vremena pri izvršavanju genetskog algoritma ([Kra99], [Kra01]). Kod genetskog algoritma veći deo vremena se potroši na računanje vrednosti funkcije cilja jedinke. Ta vrednost zajedno sa genetskim kodom jedinke se smešta u memoriju. Pre toga se proverava da li se ta jedinka pojavljivala u keš memoriji. Ako jeste, njena funkcija cilja se ne računa ponovo, već se uzima vrednost iz keš memorije. Inače, vrednost funkcije cilja se računa i zajedno sa genetskim kodom se smešta u keš memoriju.

Keš memorija se sastoji iz blokova i svaki blok služi za memorisanje jedne jedinke i eventualno direktno očitavanje njene vrednosti. Broj blokova u keš memoriji je ograničen. U slučaju da se napuni keš memorija izbacićemo najstariji nekorišćeni blok.

Tehnika keširanja je zasnovana na strategiji najstarijeg nekorišćenog bloka (*Least Recently Used Strategy-LRU*). Ova strategija je implementirana pomoću heš-red (*hash-queue*) strukture (kao u [Kra00a]), gde su smešteni pokazivači na sve blokove keš memorije i ona omogućava sledeće operacije:

1. Pretraživanje keš memorije korišćenjem heš tabele
2. Izbacivanje najstarijeg nekorišćenog bloka iz keš memorije
3. Ubacivanje aktuelne jedinke u keš memoriju ako se ona već ne nalazi u keš memoriji

U ovoj implementaciji genetskog algoritma broj keširanih vrednosti funkcije cilja je ograničen na $N_{cache} = 5000$.

3.3.6 Ostale karakteristike hibridnog genetskog algoritma

U praksi vrlo često se javlja potreba da se izvršenje genetskog algoritma na neki način ubrza. U tom cilju se uvode nove heuristike čija primena i po nekoliko puta može ubrzati rad genetskog algoritma. Heuristike je moguće primeniti na neku pojedinačnu jedinku (najčešće je to najbolja jedinka), na nekoliko jedinki ili na celu populaciju. Sama heuristika može se na više načina upotrebiti unutar genetskog algoritma:

1. Heuristika za računanje optimizacione funkcije (čitanjem odgovarajućeg genetskog koda jedinke)
2. Heuristike za korekciju rešenja
3. Heuristike za generisanje početne populacije

U ovom poglavlju je opisana metaheuristika koja predstavlja hibridizaciju genetskog algoritma. Hibridni genetski algoritam koristi heuristiku lokalnog pretraživanja za poboljšanje 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.

U svakoj generaciji, pre primene genetskih operatora, kod UMAHLP-a heuristikom lokalnog pretraživanja pokušaćemo da poboljšamo kvalitet rešenja. To ćemo uraditi tako što ćemo na najboljoj jedinki na nekom čvoru promeniti status tj. ako je na određenoj poziciji ne-hab čvor on postaje hab i obrnuto. Posle svake promene izračunaćemo vrednost funkcije cilja. Ukoliko dobijemo poboljšanje nastavljamo dalje sve dok se jedinka može poboljšati. Inače, nastavljamo sa radom genetskog algoritma.

Na slici 3.2 je prikazana osnovna struktura hibridnog genetskog algoritma:

```
Unošenje_ Ulaznih_ Podataka();
Generisanje_Početne_Populacije();
while(!Kriterijum_Zaustavljanja_Genetskog_Algoritma())
{
     for (i=1;i<=N_populacije;i++ )
     {
          Obj[i]= Funkcija_Cilja(i);
          Heuristika_Lokalnog_Pretraživanja(i);
     }
     Funkcija_Prilagođenosti();
     Selekcija();
     Ukrštanje();
     Mutacija();
}
Štampanje_Izlaznih_Podataka();
```

Slika 3.2 Osnovna struktura hibridnog genetskog algoritma

Cilj genetskog algoritma je da dobre jedinke češće prolaze u sledeću generaciju da bi njihovim ukrštanjem dobili još bolje jedinke. Međutim, može se desiti da pojedine dobre jedinke počnu da dominiraju populacijom izbacujući jedinke koje imaju lošiju vrednost, ali koje mogu da sadrže dobre gene. Na taj način će se kopirati dobre jedinke. Fizičko uklanjanje višestrukih pojava jedinke bilo bi relativno sporo. Zato se jedinke markiraju u tekućoj, a implicintno uklanjaju u sledećoj generaciji. To se vrši dodeljivanjem nulte vrednosti za prilagođenost tih jedinki, pa ih operator selekcije uopšte ne uzima u obzir pri izboru jedinki za narednu generaciju.

3.3.7 Parametri genetskog algoritma

Izbor parametara genetskog algoritma, kao što su nivo ukrštanja, nivo mutacije, broj jedinki u populaciji, je vrlo važan za primenu genetskog algoritma. Mnogi pokušaji poboljšanja performansi genetskih algoritama zasnovani na određivanju optimalnih parametara genetskih algoritama su dali relativno skromne rezultate.

Pri izvršavanju genetskog algoritma je primećeno da fiksni izbor parametara, a naročito nivo mutacije, nije uvek najpogodniji ([Bëc93]). Raznovrsnost genetskog materijala nije uvek jednaka u svim fazama izvršavanja genetskog algoritma, pa su česti slučajevi da se optimalne vrednosti nivoa ukrštanja, odnosno mutacije, menjaju tokom izvršavanja genetskog algoritma. Načini promene parametara genetskog algoritma u toku izvršavanja su grupisani u dve kategorije:

- Fiksna promena parametara kod koje se unapred zadaje smer promene (povećavanje ili smanjivanje vrednosti tokom generacija) i formula po kojoj se vrši promena.
- Adaptivna promena parametara, koja parametar može promeniti u zavisnosti od toga kakve je rezultate operator do tada dao, odnosno koliko je bio uspešan.

4. REZULTATI

U ovom poglavlju predstavljeni su rezultati predloženog genetskog algoritma i sva testiranja su izvedena na računaru Intel sa 2.5GHz procesorom i sa 1 GB RAM memorije pod Linux (Knoppix 5.3.1) operativnim sistemom. Algoritam je kodiran u programskom jeziku C. Za testiranje algoritma korišćene su AP instance. AP instance su dobijene iz studije o australijskom poštanskom sistemu isporuke. Instanca najveće dimenzije iz ovog skupa uključuje 200 čvorova (regione poštanskih brojeva), dok se manje instance mogu dobiti iz najveće agregacijom skupa čvorova. Rastojanje između gradova zadovoljava nejednakost trougla, ali protok između uređenih parova početnih i krajnih čvorova nije simetričan. Fiksne cene su uključene u AP setovima podataka kao u [Ern99]. Koeficijenti χ, δ i α koji odgovaraju kolekciji, distribuciji i transportu između habova uzimaju iste vrednosti kao u [CGM07].

Kolone u tabelama 4.1-4.4 sadrže sledeće podatke (po navedenom redu):

- Dimenzija trenutne AP instance gde L (*loose*) označava „lakše“, a T (*tight*) „teže“ fiksne cene;
- Optimalno rešenje (opt_{vr}) ako je poznato unapred, u suprotnom pišemo „ $-$ “;
- Najbolje rešenje genetskog algoritma (GA_{najb}) u slučaju kad genetski algoritam dostiže optimum za trenutnu instancu;
- Prosečno vreme t (u sekundama) potrebno da bi se dobio najbolji rezultat genetskog algoritma;
- Prosečno ukupno vreme t_{uk} (u sekundama) za završetak genetskog algoritma;
- Prosečan ukupan broj generacija (gen);
- Prosečna relativna greška *Gr* (u procentima) rešenja genetskog algoritma u odnosu na opt_{vr} ili GA_{najb}, računa se po formuli

$$Gr = \left| \frac{GA_{najb} - opt_{vr}}{GA_{najb}} \right| \cdot 100$$

- Standardna devijacija σ relativne greške (u procentima);

- Prosečan broj izračunavanja funkcije cilja ($eval$);
- Prosečna vrednost uštede primenom tehnike keširanja $Keš$ (u procentima)

U svakoj AP instanci genetski algoritam je bio pokrenut 20 puta. Maksimalan broj generacija u ovoj implementaciji genetskog algoritma je $N_{gen} = 1000$. Ponavljanje vrednosti najbolje funkcije cilja je ograničeno konstantom $N_{rep} = 500$.

Tabela 4.1 Rezultati genetskog algoritma na AP instancama sa χ=3, α=0.75 i δ=2

$inst.$	opt_{vr}	GA_{najb}	t (s)	t_{uk} (s)	gen	Gr (%)	σ (%)	$eval$	$Keš$ (%)
10L	221 032.734	221 032.734	<0.001	0.012	202.7	0.000	0.000	524.8	94.900
10T	257 558.086	257 558.086	<0.001	0.012	201.0	0.000	0.000	557.5	94.500
20L	230 385.454	230 385.454	<0.001	0.041	203.5	0.000	0.000	1633.8	84.200
20T	266 877.485	266 877.485	<0.001	0.047	207.1	0.000	0.000	1675.1	84.000
25L	232 406.746	232 406.746	<0.001	0.075	204.8	0.000	0.000	2090.8	79.900
25T	292 032.080	292 032.080	0.001	0.069	207.0	0.000	0.000	2124.2	79.800
40L	237 114.749	237 114.749	0.016	0.237	218.6	0.000	0.000	3118.2	71.800
40T	293 164.836	293 164.836	<0.001	0.202	201.0	0.000	0.000	2893.2	71.600
50L	233 905.303	233 905.303	0.020	0.424	212.4	0.000	0.000	3648.6	66.100
50T	296 024.896	296 024.896	0.021	0.404	214.7	0.000	0.000	3601.9	66.900
60L	225 042.310	225 042.310	0.018	0.620	205.7	0.000	0.000	3772.0	63.800
60T	243 416.450	243 416.450	0.036	0.868	215.8	0.000	0.000	4090.1	62.600
70L	229 874.500	229 874.500	0.078	1.223	223.3	0.041	0.185	4578.5	59.500
70T	249 602.845	249 602.845	0.050	1.097	214.8	0.000	0.000	4335.6	60.200
80L	225 166.922	225 166.922	0.294	2.449	257.9	0.000	0.000	5360.7	58.700
80T	268 209.406	268 209.406	0.129	1.885	230.3	0.000	0.000	4886.3	58.100
90L	226 857.465	226 857.465	0.153	2.405	227.3	0.000	0.000	5208.1	54.600
90T	277 417.972	277 417.972	0.144	2.465	225.4	0.000	0.000	5078.5	55.500
100L	235 097.228	235 097.228	0.447	4.218	250.2	0.000	0.000	5669.8	55.200
100T	305 097.949	305 097.949	0.049	2.513	204.6	0.000	0.000	4716.6	54.600
110L	218 661.965	218 661.965	0.168	3.679	215.9	0.067	0.299	4955.1	54.700
110T	223 891.822	223 891.822	0.326	4.623	235.8	0.000	0.000	5554.6	53.400
120L	222 238.922	222 238.922	0.201	4.312	216.4	0.000	0.000	5054.7	53.900
120T	229 581.755	229 581.755	0.725	6.418	254.1	0.000	0.000	6339.4	50.600
130L	–	223 814.109	0.891	9.876	260.3	0.000	0.000	6334.9	51.900
130T	–	230 865.451	0.822	10.017	258.1	0.000	0.000	6550.1	49.700
200L	–	230 204.343	6.596	47.401	285.9	1.487	1.837	7722.7	46.500
200T	–	268 787.633	8.173	61.300	297.8	0.126	0.160	8376.1	44.300

Tabela 4.2 Rezultati genetskog algoritma na AP instancama sa χ=1, α=0.1 i δ=1

$inst.$	opt_{vr}	GA_{najb}	t (s)	t_{uk} (s)	gen	Gr (%)	σ (%)	$eval$	$Keš$ (%)
10L	122 038.940	122 038.940	<0.001	0.010	201	0.000	0.000	546	94.600
10T	127 425.939	127 425.939	<0.001	0.010	203	0.000	0.000	537	94.800
20L	125 309.816	125 309.816	<0.001	0.023	205	0.000	0.000	1610	84.500
20T	129 079.794	129 079.794	<0.001	0.022	201	0.000	0.000	1584	84.500
25L	126 821.800	126 821.800	0.001	0.040	208	0.000	0.000	2037	80.700
25T	143 422.390	143 422.390	0.002	0.041	211	0.000	0.000	1995	81.300
40L	124 994.499	124 994.499	0.017	0.113	226	0.000	0.000	3065	73.100
40T	140 962.910	140 962.910	<0.001	0.105	201	0.000	0.000	2774	72.800
50L	120 871.926	120 871.926	0.012	0.183	207	0.000	0.000	3378	67.800
50T	152 294.536	152 294.536	<0.001	0.179	201	0.000	0.000	3282	67.800
60L	112 991.944	112 991.944	0.026	0.281	212	0.000	0.000	3710	65.500
60T	124 961.384	124 961.384	0.033	0.329	215	0.000	0.000	4026	63.000
70L	114 595.951	114 595.951	0.134	0.488	256	0.000	0.000	4732	63.300
70T	134 324.296	134 324.296	0.052	0.482	217	0.000	0.000	4371	60.200
80L	116 505.953	116 505.953	0.164	0.669	248	0.040	0.178	4974	60.100
80T	138 970.736	138 970.736	0.077	0.661	218	0.000	0.000	4636	58.000
90L	115 225.601	115 225.601	0.061	0.765	210	0.000	0.000	4652	56.200
90T	130 558.600	130 558.600	0.121	0.871	223	0.000	0.000	5020	55.500
100L	123 822.587	123 822.587	0.160	1.060	226	0.005	0.023	5195	54.600
100T	143 119.855	143 119.855	0.048	1.039	205	0.000	0.000	4757	54.200
110L	110 192.705	110 192.705	0.234	1.346	230	0.000	0.000	5297	54.400
110T	114 895.505	114 895.505	0.167	1.374	219	0.000	0.000	5129	53.700
120L	111 758.347	111 758.347	0.781	2.082	302	0.534	0.950	6881	54.600
120T	118 376.769	118 376.769	0.503	1.894	256	0.000	0.000	6242	51.700
130L	–	115286.957	0.780	2.610	270	0.000	0.000	6510	52.000
130T	–	119538.946	0.249	2.208	217	0.000	0.000	5530	49.700
200L	–	120377.895	2.273	7.240	279	0.418	0.728	7451	46.900
200T	–	133716.442	1.364	6.747	241	0.004	0.013	6770	44.500

Tabela 4.3 Rezultati genetskog algoritma na AP instancama sa χ=1, α=0.5 i δ=1

$inst.$	opt_{vr}	GA_{najb}	t (s)	t_{uk} (s)	gen	Gr (%)	σ (%)	$eval$	$Keš$ (%)
10L	125 591.591	125 591.591	<0.001	0.010	201	0.000	0.000	539	94.700
10T	127 425.939	127 425.939	<0.001	0.010	204	0.000	0.000	534	94.800
20L	126 058.465	126 058.465	<0.001	0.020	201	0.000	0.000	1519	85.100
20T	129 079.794	129 079.794	<0.001	0.021	201	0.000	0.000	1558	84.700
25L	126 900.890	126 900.890	0.001	0.036	212	0.000	0.000	1897	82.400
25T	143 422.390	143 422.390	<0.001	0.039	210	0.000	0.000	1955	81.600
40L	125 199.814	125 199.814	<0.001	0.100	201	0.000	0.000	2650	74.000
40T	140 962.910	140 962.910	<0.001	0.103	201	0.000	0.000	2750	73.000
50L	124 917.187	124 917.187	0.001	0.170	201	0.000	0.000	3204	68.600
50T	152 294.536	152 294.536	<0.001	0.174	201	0.000	0.000	3233	68.300
60L	116 799.121	116 799.121	0.016	0.260	206	0.000	0.000	3478	66.700
60T	124 961.384	124 961.384	0.032	0.324	214	0.000	0.000	4012	63.000
70L	120 503.243	120 503.243	0.078	0.429	231	0.000	0.000	4256	63.500
70T	135 016.621	135 016.621	0.056	0.476	219	0.000	0.000	4368	60.500
80L	119 405.594	119 405.594	0.029	0.555	204	0.000	0.000	4194	59.500
80T	138 970.736	138 970.736	0.076	0.649	218	0.000	0.000	4611	58.200
90L	118 611.695	118 611.695	0.128	0.780	227	0.000	0.000	4801	58.100
90T	130 558.600	130 558.600	0.119	0.853	223	0.000	0.000	4991	55.700
100L	125 484.484	125 484.484	0.237	1.061	245	0.010	0.031	5332	56.700
100T	143 119.855	143 119.855	0.060	1.060	207	0.000	0.000	4786	54.300
110L	116 255.117	116 255.117	0.225	1.274	230	0.257	1.150	5146	55.700
110T	121 484.974	121 484.974	0.267	1.413	233	0.000	0.000	5403	54.100
120L	118 048.051	118 048.051	0.603	1.824	279	0.120	0.537	6271	55.400
120T	122 850.043	122 850.043	0.314	1.676	232	0.000	0.000	5650	51.900
130L	–	120773.444	0.153	1.970	210	0.029	0.127	5140	51.700
130T	–	126138.979	0.147	2.070	208	0.000	0.000	5288	50.000
200L	–	122401.965	1.110	6.127	237	0.452	0.513	6277	47.400
200T	–	133772.797	0.143	5.704	201	0.000	0.000	5671	44.400

Tabela 4.4 Rezultati genetskog algoritma na AP instancama sa χ=1, α=0.9 i δ=1

$inst.$	opt_{vr}	GA_{najb}	t (s)	t_{uk} (s)	gen	Gr (%)	σ (%)	$eval$	$Keš$ (%)
10L	125 591.591	125 591.591	<0.001	0.019	201	0.000	0.000	527	94.800
10T	127 425.939	127 425.939	<0.001	0.019	203	0.000	0.000	532	94.800
20L	126 058.465	126 058.465	<0.001	0.030	201	0.000	0.000	1504	85.300
20T	129 079.794	129 079.794	<0.001	0.031	201	0.000	0.000	1526	85.000
25L	126 900.890	126 900.890	0.001	0.045	210	0.000	0.000	1873	82.400
25T	143 422.390	143 422.390	<0.001	0.049	210	0.000	0.000	1949	81.700
40L	125 199.814	125 199.814	<0.001	0.107	201	0.000	0.000	2629	74.200
40T	140 962.910	140 962.910	<0.001	0.110	201	0.000	0.000	2743	73.100
50L	124 917.187	124 917.187	0.001	0.180	201	0.000	0.000	3195	68.700
50T	152 294.536	152 294.536	<0.001	0.182	201	0.000	0.000	3233	68.300
60L	116 799.121	116 799.121	0.025	0.270	211	0.000	0.000	3490	67.400
60T	124 961.384	124 961.384	0.030	0.330	213	0.000	0.000	3986	63.100
70L	121 858.663	121 858.663	0.075	0.430	229	0.000	0.000	4195	63.700
70T	135 016.621	135 016.621	0.057	0.481	219	0.000	0.000	4370	60.600
80L	119 405.594	119 405.594	0.028	0.561	204	0.000	0.000	4173	59.600
80T	138 970.736	138 970.736	0.070	0.656	217	0.000	0.000	4574	58.300
90L	118 611.695	118 611.695	0.132	0.778	230	0.000	0.000	4765	58.900
90T	130 558.600	130 558.600	0.111	0.854	221	0.000	0.000	4963	55.700
100L	125 484.484	125 484.484	0.296	1.097	259	0.005	0.023	5530	57.400
100T	143 119.855	143 119.855	0.053	1.032	206	0.000	0.000	4751	54.400
110L	119 007.810	119 007.810	0.270	1.316	238	0.000	0.000	5286	56.000
110T	122 257.504	122 257.504	0.079	1.251	207	0.000	0.000	4879	53.400
120L	119 561.474	119 561.474	0.366	1.622	246	0.115	0.513	5616	54.700
120T	122 850.043	122 850.043	0.315	1.669	233	0.000	0.000	5652	52.000
130L	–	120773.444	0.173	2.021	212	0.000	0.000	5158	52.000
130T	–	126138.979	0.151	2.102	208	0.000	0.000	5268	50.100
200L	–	122401.965	1.155	6.001	240	0.352	0.492	6348	47.400
200T	–	133772.797	0.141	5.579	201	0.000	0.000	5662	44.500

Kao što se može videti iz tabela 4.1-4.4, za sve AP instance (osim za 130L, 130T,200L, 200T), se postiže optimalno rešenje. U sve četiri tabele najbolje rešenje genetskog algoritma se poklapa sa optimalnim u prve 24 instance.

Tabela 4.1: Prosečno vreme potrebno da bi se dobio najbolji rezultat genetskog algoritma za prve 24 instance je $t(s) \leq 0.725s$, a za poslednje 4 instance (130L, 130T, 200L i 200T) se kreće u intervalu $0.822 \leq t(s) \leq 8.173$. Zatim, prosečno ukupno vreme za završetak genetskog algoritma je najmanje za instance 10L, 10T ($0.012s$), a najveće za 200T ($61.300s$). Najmanji prosečan broj generacija je na instancama 10T, 40T ($gen = 201.0$), a najveći na 200T ($gen = 297.8$). Prosečna relativna greška rešenja genetskog algoritma je za sve instance $Gr = 0$, osim za

instance 70L ($Gr = 0.041\%$), 110L($Gr = 0.067\%$), 200L($Gr = 1.487\%$), 200T($Gr = 0.126\%$). Standardna devijacija relativne greške σ je jednaka 0, osim za instance 70L($\sigma = 0.185\%$), 110L($\sigma = 0.299\%$), 200L($\sigma = 1.837\%$), 200T($\sigma = 0.160\%$). Broj izračunavanja funkcije cilja $eval$ se kreće od 524.8 (10L) do 8376.1 (200T). Prosečna vrednost uštede primenom tehnike keširanja je najmanja za instancu 200T (44.300%), a najveća za 10L (94.900%).

Tabela 4.2: Prosečno vreme potrebno da bi se dobio najbolji rezultat genetskog algoritma za prve 24 instance je $t(s) \leq 0.781$, a za poslednje 4 instance (130L, 130T, 200L i 200T) se kreće u intervalu $0.249 \leq t(s) \leq 2.273$. Zatim, prosečno ukupno vreme za završetak genetskog algoritma je najmanje za instance 10L, 10T ($0.010s$), a najveće za 200L ($7.240s$). Najmanji prosečan broj generacija je na instancama 10L, 20T, 40T i 50T ($gen = 201$), a najveći na 120L ($gen = 302$). Prosečna relativna greška rešenja genetskog algoritma je za sve instance $Gr = 0$, osim na instancama 80L($Gr = 0.040\%$), 100L($Gr = 0.005\%$), 120L($Gr = 0.534\%$), 200L($Gr = 0.418\%$), 200T($Gr = 0.004\%$). Standardna devijacija relativne greške σ je jednaka 0, osim za instance 80L($\sigma = 0.178\%$), 100L($\sigma = 0.023\%$), 120L($\sigma = 0.950\%$), 200L ($\sigma = 0.728\%$), 200T ($\sigma = 0.013\%$). Broj izračunavanja funkcije cilja $eval$ se kreće od 537 (10T) do 7451 (200L). Prosečna vrednost uštede primenom tehnike keširanja je najmanja za instancu 200T (44.500%), a najveća za 10T (94.800%).

Tabeli 4.3: Prosečno vreme potrebno da bi se dobio najbolji rezultat genetskog algoritma za prve 24 instance je $t(s) \leq 0.603$, a za poslednje 4 instance (130L, 130T, 200L i 200T) se kreće u intervalu $0.147 \leq t(s) \leq 1.110$. Zatim, prosečno ukupno vreme za završetak genetskog algoritma je najmanje za instance 10L, 10T ($0.010s$), a najveće za 200L ($6.127s$). Najmanji prosečan ukupan broj generacija je na instancama 10L, 20L, 20T, 40L, 40T, 50L, 50T, 200T ($gen = 201$), a najveći na 120L ($gen = 279$). Prosečna relativna greška rešenja genetskog algoritma je za sve instance $Gr = 0$, osim na 100L($Gr = 0.010\%$), 110L($Gr = 0.257\%$), 120L($Gr = 0.120\%$), 130L($Gr = 0.029\%$), 200L($Gr = 0.452\%$). Standardna devijacija relativne greške σ je jednaka 0, osim na 100L ($\sigma = 0.031\%$), 110L($\sigma = 1.150\%$), 120L($\sigma = 0.537\%$), 130L($\sigma = 0.127\%$), 200L($\sigma = 0.513\%$). Broj izračunavanja funkcije cilja $eval$ se kreće od 534 (10T) do 6277 (200L). Prosečna vrednost uštede primenom tehnike keširanja je najmanja za instancu 200T (44.400%), a najveća za 10T (94.800%).

Tabeli 4.4: Prosečno vreme potrebno da bi se dobio najbolji rezultat genetskog algoritma za prve 24 instance je $t(s) \leq 0.366$, a za poslednje 4 instance (130L, 130T, 200L i 200T) se kreće u intervalu $0.141 \leq t(s) \leq 1.155$. Zatim, prosečno ukupno vreme za završetak genetskog algoritma je najmanje za instance 10L, 10T ($0.019s$), a najveće za 200L ($6.001s$). Najmanji prosečan ukupan broj generacija je na instancama 10L, 20L, 20T, 40L, 40T, 50L, 50T i 200T ($gen = 201$), a najveći na 100L ($gen = 259$). Prosečna relativna greška rešenja genetskog algoritma je za sve instance $Gr = 0$, osim

na 100L($Gr = 0.005\%$), 120L($Gr = 0.115\%$), 200L($Gr = 0.352\%$). Standardna devijacija relativne greške σ je jednaka 0, osim na 100L($\sigma = 0.023\%$), 120L($\sigma = 0.513\%$), 200L($\sigma = 0.492\%$). Broj izračunavanja funkcije cilja $eval$ se kreće od 527 (10L) do 6348 (200L). Prosečna vrednost uštede primenom tehnike keširanja je najmanja za instancu 200T (44.500%), a najveća za 10L, 10T (94.800%).

Kao što se može videti u tabelama 4.1-4.4, predloženi genetski algoritam brzo dostiže sva poznata optimalna rešenja ($n \leq 120$) za $t \leq 0.781s$. Za ostale instance većih dimenzija, za koje optimum nije poznat, genetski algoritam dobija rešenje za $t \leq 8.173s$. Međutim, kolone t_{uk} u tabelama 4.1-4.4 pokazuju da naš algoritam prolazi kroz dodatno vreme $t_{uk} - t$ (dok ne bude zadovoljen završni kriterijum), iako je dostigao optimalno rešenje. Pristup predloženog genetskog algoritma ne može da potvrdi optimalnost dobijenih rešenja, ali predstavlja značajan doprinos postojećoj metodi za rešavanje UMAHLP zato što omogućava rešavanje instanci problema velikih dimenzija koje su ranije bile nerešive.

5. ZAKLJUČAK

U ovom radu opisan je hibridni genetski algoritam koji je posebno dizajniran za rešavanje UMAHLP. Predloženi pristup je primenjen na NP-težak problem kombinatorne optimizacije koji ima značajnu primenu u praksi.

Razmatrani problem je težak za rešavanje i to je jedan od razloga što postojeće egzaktne metode mogu rešiti samo instance problema manjih dimenzija. Postojeće heuristike, zasnovane na principu lokalnog pretraživanja, uglavnom ne daju rešenja zadovoljavajućeg kvaliteta na instancama problema većih dimenzija. Predložena hibridizacija evolutivnog pristupa i lokalnog pretraživanja hibridnog genetskog algoritma je vrlo robustna i efikasna u rešavanju datog problema, čak i na instancama velikih dimenzija.

Koncept genetskog algoritma opisan u ovom radu koristi binarno kodiranje jedinki-rešenja. Primenjeni genetski operatori koji su pokazali najbolje rezultate pri rešavanju UMAHLP su: fino-gradirana turnirska selekcija, operator jednopozicionog ukrštanja i operator mutacije sa zaleđenim genima. Kod implementacije genetskog algoritma keširanje u velikoj meri poboljšava performanse, ali ne utiče na ostale aspekte algoritma. Takođe, primenjene su razne metode za sprečavanje preuranjene konvergencije usled gubljenja raznovrsnosti genetskog materijala i spore konvergencije genetskog algoritma. Predložena implementacija primenjuje stacionarni genetski algoritam sa elitističkom strategijom. Heuristika lokalnog pretraživanja se primenjuje na najbolju jedinku u svakoj generaciji genetskog algoritma, ukoliko se ona promenila u odnosu na prethodnu generaciju. Lokalno pretraživanje se koristi pre primene genetskih operatora selekcije, ukrštanja i mutacije. Prikazani rezultati hibridnog genetskog algoritma jasno pokazuju da su dobijena rešenja visokog kvaliteta. Na svim rešavanim instancama problema dobijena su optimalna rešenja. Predloženi koncept hibridnog genetskog algoritma takođe daje rešenja na instancama problema velikih dimenzija koje do sada nisu razmatrane u literaturi.

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

1. Paralelizacija opisanog genetskog algoritma i izvršavanje na paralelnim računarima sa većim brojem procesora
2. Hibridizacija sa nekim drugim heuristikama i/ili egzaktnim metodama
3. Modifikacija opisanog genetskog algoritma za rešavanje sličnih problema kombinatorne optimizacije

LITERATURA

[AbH98a] **Abdinnour-Helm S.**, "A Hybrid Heuristic for the Uncapacitated Hub Location Problems", *European Journal of Operational Research*, Vol. 106, pp.489-499 (1998).

[AbH98b] **Abdinnour-Helm S., Vekataramanan M.A.**, "Solution Approaches to Hub Location Problems", *Annals of Operations Research*, *European Journal of Operational Research*, Vol. 78, pp.31-50 (1998).

[AlK08] **Alumur S., Kara B.Y.,** " Network Hub Location Problems: The state of the art", *European Journal of Operational Research*, Vol. 190, pp. 1-21 (2008)

[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

[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

[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)

[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).

[Brs88] **Brassard G., Bratley P.**, „Algorithms: Theory and Practice“, Prentice-Hall Int., Englewoods Cliffs NJ (1988)

[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)

[CGM07] **Cánovas L., García S., Marin A.,** „Solving the Uncapacitated Multiple Allocation Hub Location Problem by Means of a Dual-ascent Technique“, *European Journal of Operational Research*, Vol. 179, pp. 990-1007 (2007)

[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).

[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).

[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).

[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).

[Ern98b] **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)

[Ern99] **Ernst, A.T., Krishnamoorthy M**.,"Solution algorithms for the capacitated single allocation hub location problem“, Annals of Operational Research, Vol.86, pp.141-159 (1999)

[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)

[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).

[Fil06] **Filipović V.,** “Operatori selekcije i migracije i WEB servisi kod paralelnih evolutivnih algoritama”, Doktorska disertacija, Matematički fakultet, Beograd (2006)

[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).

[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).

[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).

[Jun98] Jungnickel D., “Graphs, Networks and Algorithms”, Springer Verlag, Berlin (1998)

[Kli96] **Klincewitz J.G.,** “A Dual Algorithm for the Uncapacitated Hub Location Problem”, *Location Science*, Vol.4, No.3, pp.173-184 (1996).

[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).

[Kra05] **Kratica J.,Stanimirović Z., Tošić D., Filipović V.,** „Genetic Algorithm for Solving Uncapacitated Multiple Allocation Hub Location Problem“, *Computing & Informatics*, vol.24, pp. 1001-1012 (2005).

[Kra07] **Kratica J., Kovačević-Vujčić V., Čangalović M.,** „Computing the Metric Dimension of Graphs by Genetic Algorithms“, submitted to *Computational Optimization and Applications* (2007)

[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).

[Mar08] **Marić M.,** “Rešavanje nekih NP – teških hijerarhijsko-lokacijskih problema primenom genetskih algoritama”, Doktorska disertacija, Matematički fakultet, Beograd (2008).

[MCL06] **Marín A., Cánovas L., Landete M.,** "New formulations for the Uncapacitated Multiple Allocation Hub Location Problem.", *European Journal of Operational Research*, Vol.172 (1),pp. 274-292 (2006).

[May02] **Mayer G., Wagner B.,** "An exact solution method for the multiple allocation hub location problem", *Computers and Operational Research,* Vol. 29, pp. 715-739 (2002).

[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).

[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).

[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).

[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).

[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).

[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).

[Sta04] **Stanimirović Z.,** "Rešavanje nekih diskretnih lokacijskih problema primenom genetskih algoritama", Magistarska teza, Matematički fakultet, Beograd (2004).

[Sta07] **Stanimirović Z.,** "Genetski algoritmi za rešavanje nekih NP-teških hab lokacijskih problema", Doktorska disertacija, Matematički fakultet, Beograd (2007).

[Top05] **Topcuoglu H., Corut F., Ermis M., Yilmaz G.,** "Solving the Uncapacitated Hub Location Problem Using Genetic Algorithms", *Computers & Operations* Research, Vol.32, pp. 967-984 (2005)

[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