Univerzitet u Beogradu
Matematički fakultet

Zorica Stanimirović

GENETSKI ALGORITMI ZA REŠAVANJE NEKIH NP-TEŠKIH HAB LOKACIJSKIH PROBLEMA

Doktorska disertacija

B e o g r a d

2007.

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

Komentor: **dr Jozef Kratica**
naučni saradnik
Matematički institut SANU

Članovi komisije: **Prof. dr Dušan Tošić**
Matematički fakultet u Beogradu

dr Milan Dražić
Matematički fakultet u Beogradu

Datum odbrane: _____

Genetski algoritmi za rešavanje nekih NP-teških hab lokacijskih problema

Rezime

U ovom radu opisani su različiti genetski algoritmi (GA) za rešavanje četiri NP-teška hab lokacijska problema: problem p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama (USApHMP), problem p-hab medijane/centra ograničenih kapaciteta sa jednostrukim alokacijama (CSApHMP/CSApHCP) i hab lokacijski problem ograničenih kapaciteta sa jednostrukim alokacijama (CSAHLP). Ovi hab lokacijski problemi nalaze veliku primenu u dizajniranju transportnih i telekomunikacijskih sistema, poštanskih i drugih sistema isporuke, lokalnih i globalnih računarskih mreža, itd.

Za problem p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama (USApHMP), razvijene su GA metode koje koriste dva različita načina kodiranja i adekvatne modifikovane genetske operatore. U cilju poboljšanja efikasnosti predloženih genetskih algoritama, primenjena je hibridizacija oba GA koncepta sa heuristikom lokalnog pretraživanja, pa su tako nastale hibridne HGA1 i HGA2 metode koje su veoma uspešne i pri rešavanju problema velikih dimenzija.

Za rešavanje hab lokacijskih problema ograničenih kapaciteta CSApHMP, CSApHCP i CSAHLP takođe su predložene razne verzije genetskih algoritama. Primenjene su dve raziči te reprezentacije rešenja i odgovarajući genetski operatori razvijeni u skladu sa prirodom problema. Implementirani genetski operatori čuvaju korektnost jedinki u tokom generacija GA i u smislu očuvanja broja uspostavljenih habova i u smislu ograničenja kapaciteta habova.

Sve opisane genetske (evolutivne) metode testirane su na odgovarajućim standardnim ORLIB instancama iz literature. Za sva četiri hab lokacijska problema koja su razmatrana u ovom radu, predloženi (hibridni) genetski algoritmi dostižu sve do sada poznate optimalne vrednosti na datim instancama u zadovoljavajućem vremenu izvršavanja. U radu su data rešenja i za probleme velikih dimenzija (n=100,200 p≤20) za koje optimalna rešenja nisu poznata, a neki od ovih problema do sada nisu rešavani u literaturi. Dobijeni rezultati predloženih GA metoda jasno ukazuju na značaj i potencijal genetskih pristupa rešavanju hab i drugih lokacijskih problema.

Ključne reči: *Evolutivne metode, Hab lokacijski problemi, Genetski algoritmi, Kombinatorna optimizacija*

Genetic Algorithms for solving some NP-hard hub location problems

Abstract

In this paper some new genetic algorithms (GA) for solving four NP-hard hub location problems are described: Uncapacitated Single Allocation p-hub Center Problem (USApHCP), Capacitated Single Allocation p-hub Median/Center Problem (CSApHMP/CSApHCP) and Capacitated Single Allocation Hub Location Problem (CSAHLP). These hub ploblems have various applications in designing transportation and telecommunications systems, postal and other delivery systems, local and golobal computer area networks, etc.

For the Uncapacitated Single Allocation p-hub Center Problem (USApHCP), two hybrid heuristic methods, named HGA1 and HGA2 are proposed. These methods are a combination of a genetic algorithm and a generalization of the well-known fast interchange heuristic (IH). In order to investigate the effect of encoding on GA performance, two different encoding schemes are implemented: binary encoding in HGA1, and integer representation in HGA2. Modified genetic operators that keep the feasibility of individuals are designed and implemented in both HGA1 and HGA2. The performed computational experiments showed the effectiveness of both hybrid methods, even for solving large-scaled problem instances

For the capacitated variants of hub location problems CSApHMP, CSApHCP i CSAHLP, new genetic approaches are also described. In proposed genetic algorithms, new encoding schemes are implemented with appropriate objective functions. By using specific representation and modified genetic operators, proposed GA approaches keep the feasibility of individuals, i.e. the fixed number of established hubs and/or satisfying the capacity constraints on hubs.

The numerical experiments were carried out on the standard hub data set from the literature. For all four hub problems that were studied, the corresponding GA method proved to be robust and efficient in solving the problem instances with up to 200 nodes and 20 hubs. Computational experiments demonstrate that
all proposed GA methods reach all previously known optimal solutions on tested hub instances. The algorithm is also benchmarked on large scale hub instances with n=100,200 nodes and $p \leq 20$ hubs that are not solved (to optimality) so far.
The presented computational results clearly indicate the usefulness of the proposed GA approaches.

Key words: *Evolutionary Methods, Hub Location Problems, Genetic Algorithms, Combinatorial Optimization*

PREDGOVOR

Rad se sastoji od šest poglavlja: uvodno poglavlje daje osnovne informacije o hab lokacijskim problemima i kratak pregled postojećih metoda iz literature za njihovo rešavanje. U uvodnom poglavlju date su još osnovne karakteristike genetskih algoritama (GA) i pregled dosadašnjih primena GA na probleme kombinatorne optimizacije, a posebno na hab lokacijske probleme. Primena GA za rešavanje problema p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama opisana je u drugom poglavlju. U tri naredna poglavlja predložene su GA implementacije za rešavanje tri hab lokacijska problema ogranicenih kapaciteta: hab lokacijskog problema sa jednostrukim alokacijama u trećem, problema p-hab medijane sa jednostrukim alokacijama u četvrtom i problema p-hab centra sa jednostrukim alokacijama u petom poglavlju. Pregled naučnog doprinosa rada i dobijenih rezultata sa zaključkom je dat u 6. poglavlju.

Želela bih da se zahvalim mentoru, dr Đorđu Dugošiji i komentoru dr Jozefu Kratici na rukovođenju pri izradi ove disertacije. Posebnu zahvalnost dugujem komentoru dr Jozefu Kratici koji me je zaineresovao za genetske algoritme i utrošio dosta vremena i energije u razmatranju ideja i sugerisanju puteva za prevazilaženje problema koji su se pojavljivali tokom rada na disertaciji. Takođe bih se zahvalila članovima komisije prof. dr Dušanu Tošiću i dr Milanu Dražiću na vrlo pažljivom čitanju rukopisa i korisnim sugestijama koje su doprinele kvalitetu ovog rada. Isto tako, zahvaljujem se kolegi dr Vladimiru Filipoviću na stručnim savetima i pomoći u toku rada na genetskim algoritmima.

Zahvaljujem se na podršci, pomoći, razumevanju i savetima koju sam dobijala od kolega sa Katedre za Numeričku Matematiku i Optimizaciju Matematičkog fakulteta u Beogradu. Takođe se zahvaljujem na pomoći i kolegama sa Projekta 144007 "Matematicki modeli i metode optimizacije sa primenama".

Naročito se zahvaljujem svojoj porodici i priljateljima na bezgraničnom razumevanju i ogromnoj podšci koja mi je rad na ovoj disertaciji učinila neuporedivo lakšim.

Kandidat

Beograd, 2007. mr Zorica Stanimirović

1 UVOD

U ovom radu pažnja je posvećena hab lokacijskim problemima, koji predstavljaju predmet proučavanja istraživača širom sveta, i sa teorijskog i sa praktičnog aspekta. Hab lokacijski problemi su u poslednje dve decenije doživeli pravu ekspanziju, najviše zahvaljujući svojoj širokoj primeni u praksi. Mreže habova srećemo u različitim segmentima života, i njihovo dizajniranje predstavlja jednu od najprofitabilnijih oblasti primenjene matematike.

Hab lokacijski problemi su uglavnom vrlo teški za rešavanje. Štaviše, većina spada u NP teške probleme [Gar79], [Brs88], [Cre97], [Jun98] i [Dre02]. Takođe, ne postoji opšti matematički model koji dobro opisuje sve hab 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, egzaktni pristupi kao što su metoda grananja i ograničavanja (branch-and-bound) i metoda grananja i sečenja (branch-and-cut) u razumnom vremenu često ne mogu rešiti probleme većih dimenzija. Zato je do sada razvijen niz različitih heurističkih metoda koje su prilagođene karakteristikama i strukturi hab lokacijskih problema. Neke od njih su gramziva heuristika (greedy heuristic) i brojne heuristike zamene (add/remove heuristic, 1-interchange, 2-interchange, fast interchange heuristics).

Osim gore pomenutih metoda, postoje i opšti heuristički pristupi - metaheuristike koji su se pokazali vrlo efikasnim u praksi za rešavanje hab i drugih lokacijskih problema. Najčešće korišćene metaheuristike su: tabu pretraživanje (tabu search) [Glo86], [Glo90], [Her97], Metropolis algoritam [Met53], na čijem konceptu je razvijeno simulirano kaljenje (simulated annealing) [Kir83], [Čer85], Lagranževa relaksacija (Lagrangean relaxation) [Geo74], [BeJ95], metoda promenljivih okolina (variable neighborhood search) [Mla95], [Mla97], [Han99], neuronske mreže (neural networks) [Hop82], [Hop84], [Hop85], mravlji sistemi (ant systems) [Col91], [Dor91], [Dor92], [Dor96], epsilon transformacija [Zha96], [Pem96], [Kra96], genetski algoritmi (genetic algorithms) [Hll75], metoda "rasutog pretraživanja" (scatter search method) [Glo77], metoda povezivanja puta (path relinking) [Glo98], i druge. Čak i u slučajevima kada problem ima više lokalnih ekstrema, ove metaheuristike pronalaze globalni optimum, ali se u većini slučajeva optimalnost ne može dokazati. Moguće je i međusobno kombinovanje heurističkih pristupa u cilju korišćenja dobrih strana svakog od njih, kao i hibridizacija sa egzaktnim metodama čime se povećava efikasnost pri nalaženju optimalnog rešenja [Kid93], [AbH98a], [AbH99], [Pam01]. Većina pomenutih heurističkih metoda se

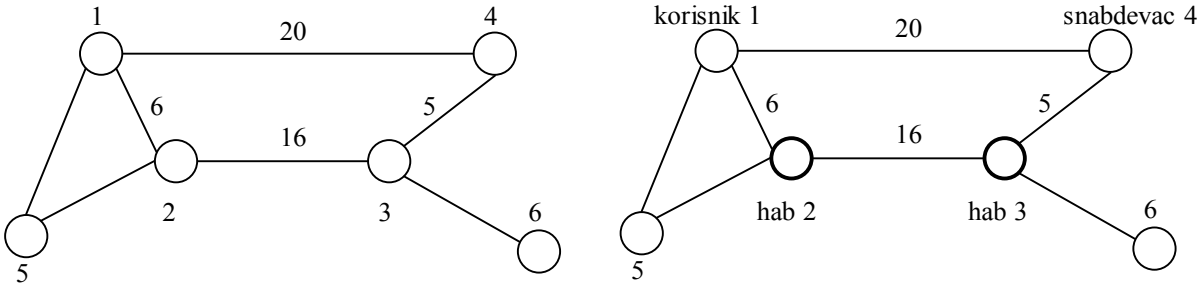
može i paralelizovati na višeprocesorskim računarima. Pregled metaheuristika sa primenama dat je u [Osm96a], [Osm96b], [Vss99], [Rib02], [Glo03].

U ovom radu opisani su razni genetski koncepti za rešavanje nekih NP teških hab lokacijskih problema ograničenih/neograničenih kapaciteta sa jeddnostrukim/višestrukim alokacijama i (eventualno) unapred zadatim brojem habova *p* koje treba uspostaviti. Preciznije, predloženi su genetski (evolutivni) algoritmi za rešavanje četiri hab lokacijska problema: problem p-hab centra neograničenih kapaciteta sa jednostrukim alokacijama (uncapacitated singe allocation p-hub center problem), hab lokacijski problem ograničenih kapaciteta sa jednostrukim alokacijama (capacitated singe allocation hub location problem), problem p-hab medijane ograničenih kapaciteta sa jednostrukim alokacijama (capacitated singe allocation p-hub median problem) i problem p-hab centra ograničenih kapaciteta sa jednostrukim alokacijama (capacitated singe allocation p-hub center problem).

1.1 Hab lokacijski problemi

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 većem broju 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, ali i u budućem vasionskom saobraćaju. 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.



Slika 1.1 Hab mreža sa n=6, p=2. Cena transporta hab-hab je 0.25 po jedinici količini robe, a 1 inače.

Telekomunikacijski sistemi se takođe mogu konfigurisati kao mreže habova. Informacije u računarskim, telefonskim, satelitskim i drugim komunikacijskim

sistemima (kao što je Internet, na primer) 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).

Na slici 1.1 prikazana je mreža od n=6 čvorova sa odgovarajućim rastojanjima. Tačno p=2 od ovih šest čvorova treba izabrati za habove. Neka je cena transporta između habova 0.25, a cena transporta korisnik-snabdevač, snabdevač-hab i hab-korisnik 1 po jedinici količini robe. Ako čvorove 2 i 3 proglasimo za habove, tada će cena transporta od snabdevača 4 do korisnika 1 biti: 6*1+16*0.25+5*1=15. Da je roba transportovana direktno od čvora 4 do čvora 1, cena transporta bi bila 20*1=20.

1.1.1 Osnovni hab lokacijski modeli. Šema višestruke i jednostruke alokacije

U ranim fazama proučavanja hab lokacijskih problema za svaki klasični diskretni lokacijski problem (problem p-medijane, p-centra, prost lokacijski problem sa neograničenim kapacitetima, problem pokrivanja, itd) formulisan je odgovarajući hab lokacijski problem (problem p-hab medijane, p-hab centra, problem lokacije habova sa neograničenim kapacitetima, problem hab pokrivanja, itd). U radovima [Cam94a] i [OK94] nalazi se pregled "prve generacije" hab lokacijskih problema i odgovarajućih modela koji se prvenstveno odnose na minimiziranje ukupnih transportnih troskova korisnik-snabdevač u mreži i troškova uspostavljanja habova. U ovim modelima podrazumeva se jedinstvena niža cena transporta između habova po jedinici količine robe, pa je optimalna mreža uspostavljenih habova zapravo kompletan graf i celokupan transport u mreži se obavlja preko njih. Takođe se može zahtevati da broj hab lokacija koje treba uspostaviti bude tačno *p* (*p-hub location problems*).

Za razliku od klasičnih lokacijskih problema, svaki hab lokacijski problem ima dve varijante, koje koriste različite alokacijske koncepte. Razlikujemo 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.

Šema višestruke alokacije omogućava veću fleksibilnost modela, jer za dati skup habova transport izmedju snabdevača i korisnika može biti realizovan putem sa najnižom cenom transporta, nezavisno od ostalih čvorova. Međutim, većina hab lokacijskih problema podrazumeva šemu jednostruke alokacije, jer je svaki korisnik/ snabdevač obično pridružen svom najbližem habu, odnosno habu sa najnižim troškovima transporta.

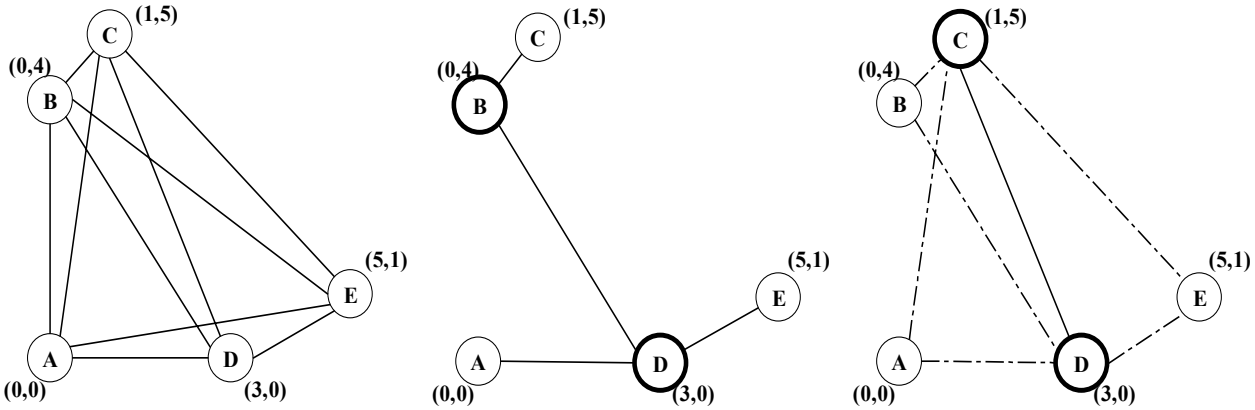
Na primer, putnici u avio saobraćaju koji se nalaze u istom gradu (polaznom čvoru) mogu izabrati da lete preko različitih gradova (habova) u zavisnosti od grada njihove destinacije (dolaznog čvora). Slično, u poštanskim mrežama pošta iz polaznog poštanskog centra može da se distribuira preko različitih poštanskih terminala do dolaznog centra, tako da troškovi transporta budu

minimalni. Međutim, ako se radi o poštanskim mrežama brze isporuke (DHL sistemi) u kojima je važno minimizovati maksimalno vreme isporuke za svaki par korisnik-snabdevač, u odgovarajućim hab modelima koristi se šema jednostruke alokacije. Jasno je da će usled relaksacije uslova u hab modelima sa šemom višestruke alokacije, troškovi transporta u tom slučaju biti niži u odnosu na modele sa šemom jednostruke alokacije.

Razlika između šema jednostruke i višestruke alokacije je ilustrovana na slici 1.2. U mreži sa n=5 čvorova, koji su zadati (x,y) koordinatama u ravni, potrebno je uspostaviti p=2 haba, tako da maksimalni troškovi transporta budu što manji, pri čemu nema ograničenja kapaciteta. Pri tom se transport između svakog para ne-hab čvorova mora odvijati preko jednog ili više hab čvorova (nije dopušten direktni transport korisnik-snabdevač). Neka je cena transporta između ne-hab čvora i haba 1, dok je cena transporta između habova 0.75 po jedinici količine robe. Odgovarajuća matrica rastojanja između parova čvorova u mreži je:

$$\begin{bmatrix} d(A,A) & d(A,B) & d(A,C) & d(A,D) & d(A,E) \\ d(B,A) & d(B,B) & d(B,C) & d(B,D) & d(B,E) \\ d(C,A) & d(C,B) & d(C,C) & d(C,D) & d(C,E) \\ d(D,A) & d(D,B) & d(D,C) & d(D,D) & d(D,E) \\ d(E,A) & d(E,B) & d(E,C) & d(E,D) & d(E,E) \end{bmatrix} = \begin{bmatrix} 0 & 4 & \sqrt{26} & 3 & \sqrt{26} \\ 4 & 0 & \sqrt{2} & 5 & \sqrt{34} \\ \sqrt{26} & \sqrt{2} & 0 & \sqrt{29} & 4\sqrt{2} \\ 3 & 5 & \sqrt{29} & 0 & \sqrt{5} \\ \sqrt{26} & \sqrt{34} & 4\sqrt{2} & \sqrt{5} & 0 \end{bmatrix}$$

Na desnoj strani slike 1.2 prikazano je optimalno rešenje problema sa višestrukom alokacijskom šemom, a u sredini je rešenje istog problema sa šemom jednostruke alokacije.



Slika 1.2 Optimalno rešenje za šemu jednostruke/višestruke alokacije za p-hab problem neograničenih kapaciteta sa n=5, p=2. Cena transporta hab-hab je 0.75 po jedinici količini robe, a 1 inače.

Kao što se može videti, rešenja se razlikuju ne samo u lokacijama habova, već i u pridruživanju uspostavljenih habova čvorovima korisnicima/snabdevačima. U slučaju šeme jednostruke alokacije, uspostavljeni su habovi **B** i **D**, a svaki ne-hab čvor komunicira preko tačno jednog, njemu pridruženog haba (čvorovi A i E preko haba **B**, a čvor B preko haba **C**). Kod šeme višestruke alokacije, čvorovi **C** i **D** su izabrani za habove, a ne-hab čvorovi A, E i B mogu komunicirati preko haba **C** ili preko haba **D** u zavisnosti od toga koja je cena transporta povoljnija. Na primer, ako transportujemo robu

od snabdevača A do korisnika E preko haba **D**, transportni troškovi duž puta "A-**D**-E" su 3+$\sqrt{5}$=5.23. Ako se transport robe realizuje preko haba **C,** transportni troškovi duž puta "A-**C**-E" su: $\sqrt{2}+\sqrt{34}$ = 7.25, tako da za transport robe od A do E biramo hab **D**. Međutim, za transport robe od čvora A do čvora B biramo hab C, jer su troškovi transporta "A-**C**-B" $\sqrt{26}+\sqrt{2}$=6.51 manji od troškova transporta "A-**D**-B" 3+5=8 i "A-**D**-**C**-B" 3+0.75*$\sqrt{29}+\sqrt{2}$=8.453. Maksimalni troškovi transporta pri šemi jednostruke alokacije se ostvaruju za 8.164 za transport "A-**D**-**B**-C" i iznose 3+5*0.75+$\sqrt{2}$=8.164, a u slučaju šeme višestruke alokacije za transport "B-**C**-E" $\sqrt{2}+4\sqrt{2}$=7.071. Ušteda u slučaju šeme višestruke alokacije je oko 13.4%.

1.1.2 Funkcija cilja raznih hab lokacijskih problema

Većina hab lokacijskih problema ima za cilj minimizaciju transportnih i eventualno nekih drugih troškova u mreži (troškovi uspostavljanja habova, lukova, troškovi kašnjenja itd). Tip funkcije cilja hab lokacijskog modela zavisi od tipa realnog problema na koji se dati model odnosi. U zavisnosti od tipa funkcije cilja izdvajaju se:

I Problemi hab medijane, sa funkcijom cilja koja minimizira sumu troškova transporta za svaki par korisnik-snabdevač preko odgovarajućeg haba/habova, pri čemu je broj habova koje treba uspostaviti unapred zadat - *p*. Problemi p-hab medijane odgovaraju klasičnom lokacijskom problemu p-medijane. Specijalni slučajevi sa *p* =1,2 habova predloženi su u [OK86], dok je prva opšta formulacija problema p-hab medijane data u [OK87]. Razne varijante ovog hab lokacijskog problema proučavane su u: [OK96], [Sko96], [Ern98a], [Ern98b], [Bol04], [Pér04], [Pér05] itd.

II Problemi hab centra, koji se bave uspostavljanjem određenog broja habova - *p*, u cilju minimizacije maksimalnog rastojanja/vremena/troškova transporta između bilo koja dva čvora u mreži (*minmax* kriterijum). Problemi p-hab centra su nastali po ugledu na klasičan lokacijski problem *p*-centra. Funkcija cilja hab centra se posebno koristi u slučajevima kada je maksimalno rastojanje između korisnika i snabdevača važno (sistemi brze isporuke). Prva formulacija problema p-hab centra data je u [OK91], iako se različite linearizacije *minmax* formulacije nalaze već u [OK87] u okviru studije o mrežama teretnog saobraćaja. U [OK87] je dokazana i ekvivalentnost problema 1-centra sa poznatim lokacijskim problemom 1-centra. Formulacije problema *p*-hab centra nalazimo zatim u [Cam94], [Kara99a], [Kara00] a ovaj problem je razmatran u brojnim radovima iz literature, kao što su: [Pam01], [Ern04a], [Ern04b], [Ca05], [Kra06], itd.

III Hab lokacijski problemi, kod kojih je potrebno uspostaviti optimalan broj habova, tako da ukupni transportni troškovi u mreži budu minimalni. Ovi problemi predstavljaju "hab analogiju" klasičnog prostog lokacijskog problema. Funkcija cilja hab lokacijskih problema obično uključuje fiksne troškove uspostavljanja i/ili korišćenja habova i/ili lukova u mreži, pa tako nastaju **hab lokacijski problemi sa fiksnim troškovima**. Oni imaju široku primenu u telekomunikacijskim mrežama gde su troškovi uspostavljanja hab centara ili linkova značajni, kao i u transportnim mrežama u kojima su neke stanice ili linije javno ili privatno vlasništvo (aerodromi, putevi, pruge, skladišta...), što podrazumeva naknadu za njihovo korišćenje. Cilj je uspostaviti ravnotežu

između povećavanja fiksnih troškova prilikom uspostavljanja ili korišćenja novih habova i smanjenja troškova transporta u mreži sa većim brojem habova. Hab lokacijski problem je prvi put formulisan u [OK91], a varijante sa ili bez fiksnih troškova proučavane su u: [OK92], [Kli96], [Ern99], [Eb00], [May02], [Lab03], [Bol04], [Car04], [Ma04], itd. Napomenimo da i u slučaju problema *p*-hab medijane i *p*-hab centra, funkcija cilja može uključivati razne fiksne troškove.

IV Hab lokacijski problemi sa promenljivim transportnim troškovima, sa deo po deo linearnom, konkavnom funkcijom cilja koja uključuje različite niže cene transporta između habova koje se menjaju u zavisnosti od količine protoka (što je protok veći, cena transporta opada), ali i odgovarajuće fiksne troškove koji rastu sa snižavanjem cene transporta između habova [Yam04], [Yam03].

V Ostali tipovi funkcije cilja. Funkcija cilja se ne mora obavezno odnositi na minimizaciju troškova/vremena transporta. Na primer, kod hab lokacijskih modela sa takmičenjem (hub location models with competition), funkcija cilja maksimizuje količinu protoka u mreži koji se odvija preko habova [Mar99]. U slučaju hab lokacijskih problema sa pokrivanjem (hub covering problem) zadaje se maksimalno vreme transporta kao uslov problema, a funkcija cilja minimizuje troškove uspostavljanja ovakve hab mreže [Kara99b].

1.1.3 Proširenje osnovnih hab lokacijskih modela

Proširenje osnovnih hab lokacijskih problema se odvija u nekoliko različitih pravaca i ima za posledicu formulisanje novih, kompleksnijih hab lokacijskih modela. Neki od aspekata proširenja su:

I Usložnjavanje mreže habova, pri čemu nastaju:

- modeli sa promenjivim nižim cenama transporta između habova (na primer, u zavisnosti od količine robe koja se transportuje) u [Bry98], [OK98a], [OK98b] i [Bry99]. Ovi modeli imaju primenu u teretnom avio saobraćaju;
- modeli kod kojih se transport preko svake grane u mreži realizuje po nižoj ceni ako količina robe (protok) prelazi određeni prag (flow treshold models). Ovi modeli utiču na povećavanje koncentracije protoka u mreži i korišćenje relativno malog broja grana ([Ayk96], [Sko01], [Pod02], [Pod03], [Sko05]). Posebno su značajni jer se odnose na realne situacije koje srećemo u praksi, najčešće kod dizajniranja brzih telekomunikacijskih mreža;
- modeli koji relaksiraju uslov da se celokupan transport između habova realizuje po nižim cenama, već samo jedan deo ([Cam00a] i [Cam00b]);
- modeli koji koriste grane sa nižim cenama transporta (hab grane-hub arcs). Cilj je uspostaviti fiksiran broj hab grana tako da ukupni transportni troškovi budu minimalni ([Cam03], [Cam05a], [Cam05b]);
- modeli kod kojih se habovi biraju iz unapred zadatih podskupova čvorova mreže [Su01].

II Proširenje komunikacije između ne-hab i hab čvorova u mreži, što daje:

- modele kod kojih se transport od snabdevača do korisnika ne mora odvijati isključivo preko habova u [Ayk95], [Su01];
- modele koji dopuštaju prikupljanje robe iz više čvorova-snabdevača pre kolekcije u određeni hab (multi stop access paths). Na primer, transport se može realizovati: snabdevač1→snabdevač2→hab→korisnik, kao u radu [Ayk95];
- modele sa fiksiranim rutama prikupljanja i distribucije u [Nagy98];

- modeli kod kojih se protok između para snabdevač-korisnik može podeliti na nekoliko puteva [Mar04]. Na primer, 1/3 robe se transportuje jednim putem, a preostalih 2/3 drugim.

III Usložnjavanje nivoa usluge

Nivo usluge (level of service) u mreži se definiše kao (maksimalna) dužina puta ili (maksimalno) vreme transporta od snabdevača do korisnika preko jednog ili više habova. Nivo usluge može označavati i (maksimalan) broj zaustavljanja u habovima na putu snabdevač-korisnik. Kod osnovnih hab lokacijskih modela nivo usluge (maksimalna dužina puta) je 3 luka (snabdevač→hab1→hab2→korisnik), odnosno broj zaustavljanja u habovima najviše 2. Proširivanjem nivoa usluge u mreži mogu se formulisati:

- modeli sa ograničenim (maksimalnim) vremenom transporta, koji imaju za cilj minimizaciju ukupnih troškova uspostavljanja mreže, pri čemu je broj habova koje treba locirati unapred zadat. Ovi modeli se odnose na probleme hab pokrivanja (hub covering problems) [Kara99b];
- hab modeli sa takmičenjem u kojima se habovi međusobno takmiče u opsluživanju korisnika po nižim cenama transporta [Mar99]. Do takmičenja dolazi kad god se uspostavi novi hab, što dovodi do redukcije troškova ili vremena transporta od snabdevača do korisnika. Ovi modeli su naročito korisni za dizajniranje mreže putničkog i teretnog avio saobraćaja;
- modeli sa fiksiranim brojem zaustavljanja u hab čvorovima;
- modeli sa fiksiranim brojem lukova na svakom putu snabdevač-korisnik ([Sas97], [OK98a] i [Sas99]);
- modeli koji uključuju neke aspekte problema raspoređivanja ([Kara99a]), problema maksimalnog pokrivanja ([Bo04]), problema Štajnerovog drveta ([Le96]) itd.

IV Dodavanje novih troškova, osim transportnih, pri čemu nastaju:

- modeli sa fiksnim troškovima lokacije habova [Ern99], [Eb00], [Lab03], [Bol04], [Car04], [Ma04];
- modeli sa fiksnim troškovima uspostavljanja lukova između čvorova u mreži [Car04], [Yam04], [Yam03];
- modeli sa troškovima kašnjenja (delay costs).

V Uključivanje različitih ograničenja kapaciteta u mreži, što dovodi do formulacije hab lokacijskih problema sa ograničenim kapacitetima (capacitated hub location problems). Tako nastaju:

- modeli sa limitiranom količinom robe koja dolazi i/ili prolazi kroz hab čvor [Ayk94], [Ern99], [Eb00], [Lab03], [Bol04], [Car04], [Ma04], [Yam04], [Yam03];
- modeli sa ograničenim protokom između habova kao i/ili između habova i pridruženih korisnika/snabdevača;
- modeli sa ograničenim kapacitetima ne-hab čvorova

Još neke informacije o hab lokacijskim problema mogu se naći u [Cam96] i [Cam02].

1.1.4 Postojeći načini rešavanja

Iako hab lokacijski problemi imaju relativno jednostavnu formulaciju, njihovo rešavanje je uglavnom teže u odnosu na klasične diskretne lokacijske probleme. Većina hab lokacijskih problema spada u klasu NP-teških (NP-kompletnih) problema ([Gar79] i [Cre97]). Čak i neki potproblemi hab

lokacijskih problema su takođe NP-teški. Na primer, ako u problemima p-hab medijane/centra sa jednostrukim alokacijama fiksiramo skup habova, dobijamo potprobleme optimalnog pridruživanja korisnika/snabdevača izabranim habovima, koji su takođe NP-teški [Kara98], [Kara99b], [Kara00], [Ern04a]. Ako u polaznim problemima nema redukcije cene transporta između habova, ponovo dobijamo NP-kompletne potprobleme, što je dokazano u radu [Ern04b].

Ipak, postoje specijalni slučajevi i/ili potproblemi hab lokacijskih problema koji se mogu rešiti u polinomskom vremenu. Na primer, problemi 1-hab medijane i 1-hab centra (p=1) su polinomske složenosti. Takvi su i potproblemi p-hab centra/medijane neograničenih kapaciteta sa višestrukim alokacijama koji nastaju fiksiranjem skupa habova ([Ern98a], [Soh98], [Cam02]).

S obzirom na veliki praktični značaj hab lokacijskih problema, u poslednjih desetak godina razvijeni su brojni algoritmi za njihovo rešavanje. U literaturi nalazimo različite egzaktne metode koje uglavnom daju dobre rezultate za hab lokacijske probleme manjih dimenzija. Za rešavanje problema velikih dimenzija koje srećemo u praksi, koriste se razne heuristike, metaheuristike i njihove hibridizacije sa egzaktnim metodama.

U narednim sekcijama dat je kratak pregled postojećih egzaktnih i heurističkih pristupa za rešavanje nekih hab lokacijskih problema.

1.1.5 Egzaktne metode

Hab lokacijskim problemaima se najčešće pristupa kao problemima mešovitog celobrojnog linearnog programiranja (Mixed Integer Linear Programming-MILP). Jedan od osnovnih načina za egzaktno rešavanje MILP problema je pretraživanje prostora rešenja. Međutim, sa porastom dimenzije problema, potpuno pretraživanje prostora rešenja najčešće nije izvodljivo, čak ni pomoću najmoćnijih računara. Zato su razvijene metode inteligentnog pretraživanja koje, pomoću procena, usmeravaju pretraživanje u određenom pravcu i eliminišu beskorisna pretraživanja. Dve takve metode, metoda grananja i ograničavanja (Branch and Bound-BnB) i grananja i odsecanja (Branch and Cut-BnC) se intenzivno koriste za rešavanje hab lokacijskih problema. Detaljni opisi metoda zasnovanih na pretraživanju za rešavanje MILP problema mogu se naći u [Sch86] i [Pad90].

1.1.5.1 Metoda grananja i ograničavanja

Metoda grananja i ograničavanja [La60] je zasnovana na ideji da nije potrebno pretraživati one delove prostora rešenja u kojima se, po proceni, ne nalazi rešenje koje je bolje od nekog unapred poznatog ili već nađenog rešenja. Početni problem (P) se razlaže (grana) na sve manje, eventualno prostije, probleme sa ciljem da se neki od tih problema ne rešavaju, ako se proceni da njihov dopustivi skup ne sadrži optimalno rešenje polaznog problema.

- Razlaganje problema (P) se obično vrši tako što se dopustivi skup rešenja (X) pokrije unijom nekih svojih (ne obavezno disjunktnih) delova X_k, i problem (P) zameni spiskom problema (P_k), definisanih na odgovarajućim delovima X_k. Dobijeni potproblemi mogu biti lakši za rešavanje (npr. ako su delovi X_k dovoljno mali), ali ne moraju.
- Ako je potproblem (P_k) i dalje težak za rešavanje, razmatra se relaksirani problem (Q_k) problema (P_k). Relaksacirani problem (Q_k) se dobija proširivanjem dopustivog skupa X_k do nekog skupa Y_k (recimo

izostavljanjem nekih ograničenja) i/ili izborom nove funkcije cilja koja je minoranta funkcije cilja polaznog problema (pod pretpostavkom da je polazni problem bio minimizacijskog tipa).

- Optimalna vrednost problema (Q_k) nije veća od optimalne vrednosti problema (P_k). Ako su ove vrednosti jednake, problem (P_k) je rešen, u suprotnom dobijamo donju granicu optimalne vrednosti (P_k).
- Ako nam je poznata neka majoranta optimalne vrednosti polaznog problema P (majoranta se obično dobija nekom heuristikom) ili tekući minimum (najmanja optimalna vrednosti do sada rešenih problema (P_k)), svaki naredni problem (P_j) možemo skinuti sa spiska (i dalje ne razmatrati), ukoliko je donja granica njegovog rešenja veća od tekućeg minimuma ili poznate majorante.
- Ukoliko se spisak problema nije ispraznio, biramo neki naredni problem i na sličan način ga obrađujemo. Postupak se završava kad se spisak problema obradi i optimalna vrednost polaznog problema (P) je jednaka tekućem minimumu, a optimalno rešenje je dopustiva tačka iz prostora X u kojoj se dostiže tekući minimum.

Postupak pretraživanja se može predstaviti grafom, koje nazivamo drvetom pretrazivanja (BnB tree), kome je koren polazni problem (P), a krajnji čvorovi (listovi) potproblemi čijim daljim grananjem ne možemo poboljšati tekući minimum.

U slučaju da se u svakom grananju generiše konačno mnogo potproblema i da se oni ne ponavljaju, algoritam će se zaustaviti posle konačno mnogo iteracija. Računska složenost BnB algoritma je eksponencijalna, ali se u nekim slučajevima do rešenja može doći vrlo brzo. Postoje razne varijante metode grananja i ograničavanja koje su prilagođene specifičnoj strukturi problema koji se rešava. Algoritam može koristiti različite načine za izbor sledećeg problema koji će se obrađivati, različita razbijanja dopustivog prostora X, načine relaksacija i obrade potproblema. Detaljan opis BnB metode može se naći u [Ber97].

1.1.5.2 Metoda grananja i odsecanja

Metoda grananja i odsecanja nastaje kombinovanjem metode grananja i ograničavanja i metode odsecanja ravni.

Osnovna ideja metode odsecanja ravni [Dan54] sastoji se u odsecanju delova dopustivog skupa linearne relaksacije koji ne sadrži celobrojne tačke. Konstrukcijom jedne linearne nejednakosti koju zadovoljava svako dopustivo rešenje problema, a ne zadovoljava dobijeno rešenje linearne relaksacije, dobijamo tzv. "odsecajuću ravan" ili "rez" koja odseca dobijena nezadovoljavajuća rešenja. Dopisivanjem date nejednakosti skupu ograničenja polaznog problema, dobijamo novi problem koji je ekvivalentan polaznom. Optimalno rešenje necelobrojne relaksacije novog problema, ako postoji, je različito od rešenja necelobrojne relaksacije polaznog problema, pa je ili optimalno za polazni probem, ili se ceo postupak odsecanja (dopisivanjem "odsecajućih" nejednakosti) može ponoviti. U literaturi nalazimo razne varijante metode odsecanja ravni, koje se razlikuju po načinu dobijanja "odsecajuće ravni", kao i relaksacija problema, a njihov pregled je dat u [Jün94].

Teorijski, metoda odsecanja ravni posle konačno mnogo koraka nalazi optimalno rešenje polaznog problema celobrojnog programiranja, ali se u praksi može desiti da je pri rešavanju problema većih dimenzija broj potrebnih koraka

veoma veliki. U cilju što efikasnijeg rešavanja problema, ova metoda se može kombinovati sa BnB algoritmom, pa tako dobijamo metodu grananja i odsecanja - BnC. U svakom čvoru BnB drveta, BnC algoritam traži "odsecajuće ravni", radi poboljšavanja granica LP relaksacije problema. Na ovaj način se mogu dobiti znatno bolje lokalne donje granice LP relaksacija, što povlači znatno manji broj neophodnih potproblema u BnB algoritmu.

U praksi, BnC algoritam se pokazao veoma uspešnim za rešavanje raznih NP-teških problema kombinatorne optimizacije: [Pad90], [Pad91], [Jün96], [Jün97] i [Mut01].

1.1.5.3 Egzaktne metode za rešavanje hab lokacijskih problema

Pri rešavanju hab lokacijskih problema kao problema MILP klase, koriste se dva osnovna tipa formulacija. Na primer, u [Cam94], [Cam96], [OK96], [Ham04] koriste se formulacije raznih hab lokacijskih problema sa $O(n^4)$ promenljivih i $O(n^3)$ ograničenja. Međutim, formulacije ovog tipa nisu pogodne za standardne metode grananja i ograničavanja (Branch and Bound-BnB) i grananja i sečenja (Branch and Cut-BnC), jer broj promenljivih i uslova brzo raste sa povećavanjem dimenzije problema. Ali, zbog dobrih ocena donje i gornje granice rešenja, fomulacije sa $O(n^4)$ promenljivih i $O(n^3)$ ograničenja se mogu koristiti za generisanje početnog rešenja za dualnu metodu penjanja (dual ascent method) i Lagranževe relaksacije (Lagrangean relaxation). Drugi tip formulacija [Ern98b] redukuje broj promenljivih na $O(n^3)$, što znači da se LP relaksacijom može rešiti znatno brže. Međutim, granice rešenja su znatno slabije, pa je veličina drveta grananja i ograničavanja (BnB tree) dosta veća. Imajući u vidu gore navedeno, može se objasniti zašto prvi tip formulacija daje bolje rezultate na problemima manjih dimenzija, a drugi tip na problemima većih dimenzija.

Algoritmi prebrojavanja (enumerative algoritms) se uglavnom koriste za rešavanje hab lokacijskih problema sa višestrukim alokacijama kada je broj habova koje treba uspostaviti relativno mali ($p \leq 5$). Za svaku dopustivu kombinaciju lociranih habova dobija se potproblem koji se može rešiti metodom najkraćeg puta (shortest path method) u polinomskom vremenu ([Cam00b] i [Ern98b]).

Ernst i Krishnamoorthy [Ern98a] takođe koriste metodu najkraćeg puta za prebrojavanje svih mogućih lokacija habova pri rešavanju problema p-hab medijane neograničenih kapaciteta sa višestrukim alokacijama. Predloženi algoritam je je eksponencijalan po p i polinomski po n, tako da na instancama problema sa relativno malim brojem habova koje treba uspostaviti ($p \leq 5$) daje tačna rešenja u razumnom vremenu računanja. Za veće dimenzije istog problema, metoda najkraćeg puta se kombinuje sa BnB algoritmom [Ern04b]. Skup čvorova mreže se deli u klastere, a zatim se računaju sve moguće alokacije tačno p habova u svim klasterima. Metoda najkraćeg puta obezbeđuje dobijanje donjih granica koje se dalje koriste u branch-and-bound šemi za dobijanje egzaktnog rešenja. Opisani hibridni pristup se može koristiti i za rešavanje hab lokacijskog problema sa jednostrukim alokacijama ([AbH98b] i [Ern04b]).

U [Soh98] predložena formulacija p-hab lokacijskog problema sa jednostrukim alokacijama sa $O(n^4)$ promenljivih, od kojih su n^2 binarne, i $O(n^3)$ ograničenja. Autori opisuju egzaktnu metodu za rešavanje problema kojim se

najpre relaksiraju binarna ograničenja postavljena na hab promenljive a zatim se rešava relaksirani problem, čime se dobijaju dosta dobre granice rešenja problema koje, u većini slučajeva, vode celobrojnom rešenju. Ako dobijeno rešenje nije celobrojno, pomoću drveta implicitnog prebrojavanja (sa samo nekoliko čvorova), lako se stiže do optimuma.

Klincewicz u [Kli96] rešava hab lokacijski problem neograničenih kapaciteta tako što implementira dualnu metodu penjanja (dual ascent method) i odgovarajuću dualnu metodu poravnjavanja (dual adjustment methods) u okviru BnB algoritma. Klincewicz najpre relaksira LP formulaciju problema datu u [Cam94b], a zatim koristi dualnu metodu penjanja za rešavanje dualnog relaksiranog problema. Dobijeno rešenje predstavlja dobru donju granicu polaznog problema, a zatim se na njega primenjuje metoda dualnog poravnavanja u cilju nalaženja dobre gornje granice. Koristeći dobijene granice, BnB algoritmom se optimalno rešavaju instance problema sa manjim brojem čvorova ($n \leq 25$).

Mayer i Wagner su u [May02] opisali hibridizaciju dualne metode penjanja i BnB algoritma za rešavanje hab lokacijskog problem neograničenih kapaciteta sa višestrukim alokacijama. Oni polaze od slabije formulacije duala polaznog problema iz [Cam94b], a zatim relaksirani dualni problem optimalno rešavaju kao varijantu transportnog problema. Dobijeno optimalno rešenje se zatim koristi kao polazna tačka za dualnu metodu penjanja kojom se rešava jača formulacija dualnog problema. Rezultat dualne metode penjanja je dobra donja granica polaznog problema, koja omogućava BnB metodi da dođe do optimalnog rešenja za instance problema sa $n \leq 40$ čvorova. Predložena hibridna metoda je efikasnija od pomenute hibridne metode u [Kli96], ali i od egzaktnog pristupa koji koristi MILP formulaciju iz [Cam02], posebno za veće dimenzije problema.

U [Bol04] date su nove MILP formulacije tri hab lokacijska problema sa višestrukim alokacijama: problem p-hab medijane neograničenih kapaciteta i hab lokacijski problem ograničenih i neograničenih kapaciteta. Kako sve tri predložene MILP formulacije imaju po $O(n^3)$ promenljivih, njihovi LP-relaksirani problemi se relativno brzo rešavaju, ali daju loše donje granice za dalju primenu BnB algoritma. Autori dokazuju nekoliko osobina razmatranih hab lokacijskih problema, a zatim ih koriste u tehnikama preprocesiranja i sečenja (preprocessing and cutting techniques) za dobijanje novih ograničenja koja "pojačavaju" LP relaksacije novih formulacija bez povećavanja broja promenljivih. Rešavanjem "pojačanih" LP-relaksiranih problema dobijaju se bolje donje granice, koje značajno povećavaju efikasnost egzaktne BnB metode. Testiranja na instancama problema sa $n \leq 50$ čvorova i $p \leq 5$ habova, pokazala su da korišćenje novih formulacija značajno smanjuje vreme izvršavanja BnB metode, ali i da su instance sa po n=100,200 čvorova i dalje nedostižne za egzaktne MILP pristupe.

Još neki egzaktni pristupi za rešavanje hab lokacijskih problema opisani su i u: [Le96], [Kli00], [Su01], [Lab03], [Ma04], [Yam02], [Yam04] itd.

1.1.5.4 Heuristike

Egzaktne metode, koje su do sada predložene u literaturi, ne mogu dati rešenja u slučaju hab lokacijskih problema velikih dimenzija. Zbog toga su razvijeni razni heuristički pristupi koji za realne dimenzije problema daju dobre

rezultate u razumnom vremenu izvršavanja. Iako heuristike često dostižu optimalna rešenja, njihov nedostatak je u tome što se optimalnost ne može dokazati.

U [OK87] data je prva formulacija problema p-hab medijane sa jednostrukim alokacijama (sa $O(n^4)$ celobrojnih promenljivih) i predložene dve heuristike koje za svaku moguću kombinaciju od *p* uspostavljenih habova određuju najbolje alokacije ne-hab čvorova uspostavljenim habovima. Prva heuristika-HEUR1 pridružuje svaki ne-hab čvor najbližem habu, dok druga heuristika-HEUR2 za svaki ne-hab čvor poredi alokacije najbližem habu i drugom najbližem habu, uzimajući onu koja daje bolje rešenje.

Klicewicz u [Kli91] za rešavanje istog problema predlaže višekriterijumsku šemu alokacije ne-hab čvorova uspostavljenim habovima, a zatim koristi heuristike jednostruke i dvostruke zamene (single and double location exchange heuristic) u cilju popravljanja rešenja. U istom radu Klincewicz opisuje i heuristiku zasnovanu na ideji grupisanja čvorova (clustering heuristics), koja najpre deli skup čvorova na p-podskupova (p-clusters), a zatim u svakom od njih bira po jedan čvor za hab.

Metoda tabu pretraživanja (tabu search method-TS) i GRASP heuristika primenjene su u [Kli92] za rešavanje problema p-hab medijane sa jednostrukim alokacijama. Isti pristupi predloženi su u [Kli00] za rešavanje hab lokacijskih problema sa konkavnom funkcijom cilja. Skorin-Kapov i saradnici u [Sko94] razvili su efikasniju varijantu heuristike tabu pretraživanja-dvofazni TS algoritam za problem p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama. 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.

Ernst i Krishnamoorthy u [Ern98b] rešavaju problem p-hab medijane neograničenih kapaciteta sa višestrukim alokacijama primenom heuristike jednostruke zamene. Pomoću ove heuristike dobija se skup uspostavljenih habova, a zatim se metodom najkraćeg puta određuju optimalne alokacije korisnika/snabdevača izabranim habovima. Testiranja su pokazala da je ovaj algoritam efikasan za male dimenzije problema ($n \leq 50$). U slučaju većih dimenzija, algoritam takođe daje rešenja, ali vreme izvršavanja znatno raste sa povećanjem broja habova koje treba locirati (*p*). Isti hab lokacijski problem sa varijantama višestruke/jednostruke alokacije rešavan je heuristikama simuliranog kaljenja (simulated annealing) u [AbH93], [Ern96], neuronskih mreža (neural networks) u [Sm95], [Dom03] i gramzivom heuristikom (greedy heuristics) u [Per98].

U [Pam01] takođe se opisuje heuristički pristup problemu p-hab centra neograničenih kapaciteta sa jednostrukim alokacijama. Autori predlažu četiri različite strategije, zasnovane na metodi najkraćeg puta, za izbor skupa od *p* habova. Za pridruživanje ne-hab čvorova uspostavljenim habovima, koriste se tri modifikacije heuristike jednostruke zamene HEUR1, MAXFLO i ACWA. U cilju sprečavanja preuranjene konvergencije, algoritam je hibridizovan sa metodom tabu pretraživanja, a dobijeno rešenje se eventualno dodatno poboljšava gramzivom heuristikom (greedy heuristic). Predložena metoda je testirana na instancama problema sa $n \leq 25$ čvorova i $p \leq 5$ habova, za razne

kombinacije strategija inicijalne lokacije habova i heuristika alokacije u cilju njihovog poređenja.

U [Ern04a] dokazana je NP-kompletnost potproblema hab cenra neograničenih kapaciteta sa jednostrukim alokacijama i predloženo nekoliko heuristika za njegovo rešavanje. Svaka heuristika je zatim hibridizovana sa evolutivnom metodom pretrage prostora rešenja (problem space search method -PSS) iz [Sto96]. Heuristike su testirane pojedinačno i u hibridizaciji sa PSS algoritmom na instancama problema sa $n \leq 100$ čvorova.

Metoda Lagranževe relaksacije primenjena je u:

- [Ayk94] za rešavanje problema p-hab medijane ograničenih kapaciteta sa višestrukim alokacijama,
- [Gav92] za hab lokacijski problem neograničenih kapaciteta sa jednostrukim alokacijama,
- [Sko96] za p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama i
- [Pir98] za problem p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama.

Dualna metoda penjanja (dual ascent method) se često koristi za rešavanje varijanti hab lokacijskih problema u telekomunikacijskim sistemima kada je potrebno minimizovati samo fiksne troškove ([Kim92], [Lee93], [Kim95] i [Yo98]).

Campbell u [Cam96] predlaže dve MILP formulacije problema p-hab medijane za obe alokacijske šeme.

- Za slučaj višestrukih alokacija, predložena je gramziva heuristika zamene koja enumerativnom tehnikom nalazi optimalni par habova, zatim postepeno dodaje habove, sve dok ih ne bude tačno *p*. Heuristika dalje vrši zamenu habova i pridruženih ne-hab čvorova sve dok ima poboljšanja rešenja.
- Za slučaj jednostrukih alokacija, Campbell koristi opisanu heuristiku zamene za dobijanje početnog rešenja, koje dalje transformiše heuristikama MAXFLO i ALLFLO.
- Heuristika MAXFLO vrši pridruživanje svakog ne-hab čvora onom habu koji najvećim delom (u smislu količine protoka) opslužuje dati čvor u slučaju višestruke alokacijske šeme istog problema.
- Heuristika ALLFLO razmatra sve moguće alokacije ne-hab čvorova uspostavljenim habovima pri šemi jednostruke alokacije i traži najbolju kombinaciju.

Za hab lokacijske probleme ograničenih kapaciteta predložene su:

- gramziva heuristika u [Bo04],

hibridizacija simuliranog kaljenja i metode promenljivog spusta (randomized descent) u [Ern99],

- metoda povezivanja puta (path relinking) u [Pér04] i [Pér05],
- metoda lokalnog pretraživanja (local search) u [Car04] i [Yam04] ,
- konstruktivna heuristika sa ograničenom zamenom (constructive heuristic with limited interchange) u [Eb00].

Neke od navedenih heuristika biće detaljnije opisane u narednim poglavljima.

Među heurističkim pristupima za rešavanje hab lokacijskih problema značajno mesto imaju genetski (evolutivni) algoritmi. Pregled primena

postojećih evolutivnih pristupa na neke hab lokacijske probleme dat je u sekciji 1.3.

1.2 Genetski algoritmi

1.2.1 Opšte karakteristike GA

Iako su i pre 70-tih godina XX veka postojali radovi u kojima su predlagana rešenja raznovrsnih problema koja simuliraju ponašanje, odnose i veze kao u prirodi, nastanak genetskih algoritama (GA) se vezuje za 1975. godinu i Hollandovu knjigu "Adaptation in natural and artificial systems" ([Hll75]). Holland je proučavao GA radi praćenja procesa prilagođavanja kod prirodnih sistema i radi razvoja sistema veštačke inteligencije koji oponašaju modele prilagođavanja. Osnovna ideja GA je simuliranje procesa prirodne evolucije jedne populacije jedinki pod dejstvom genetskih operatora. Holland je prvi uveo pojam genetskog algoritma, a postavke iz njegovih najranijih radova iz ove oblasti i danas 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], [Kra01a] i [Tošo4].

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 programi zasnovani na toj heuristici relativno brzo izvršavaju i da se 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], [Tošo4]) 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 se 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 1.3 dat je shematski zapis osnovnih elemenata GA:

```
Unošenje_Ulaznih_Podataka();
Generisanje_Početne_Populacije();
while not Kriterijum_Zaustavljanja_GA()do
    for i=1 to N_pop do
       obj[i] = Funkcija_Cilja(i);
    endfor
    Funkcija_Prilagođenosti();
    Selekcija();
    Ukrštanje();
    Mutacija();
endwhile
Štampanje_Izlaznih_Podataka();
```

Slika 1.3 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 u rešavanju problema kod kojih je funkcija prilagođenosti neprekidna i glatka, to važi i za ostale metode. Prednosti GA u odnosu na ostale metode uglavnom dolaze do izražaja tek kada to nije ispunjeno. Pošto GA ne zahteva neprekidnost i glatkost funkcije prilagođenosti, on se može primeniti i u slučajevima kada nisu ispunjeni uslovi za primenu klasičnih metoda.

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 izvršavanja 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 [Kra07] ili smanjiti njihovu prilagođenost za određenu vrednost pomoću kaznenih funkcija [Đu07]. Jedna od metoda, koja se pokazala uspešnom 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. Ova metoda 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*, koja je zasnovana na principu *turnira*. Turniri su mala takmičenja između jedinki populacije, koji se nadmeću radi preživljavanja i učešća u sledećoj generaciji. Broj jedinki koje učestvuju na turniru je *veličina turnira* (u oznaci N), koja predstavlja parametar turnirske selekcije (koji se najčešće unapred zadaje). Turnirska selekcija se realizuje tako što se najpre na slučajan način generišu podskupovi od po N jedinki, a zatim se u svakom podskupu bira najbolja jedinka koja učestvuje u stvaranju nove generacije. Obično je problem izabrati veličinu turnira 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] i [Fil06].

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 počevši od pozicije k+1 do poslednje pozicije n–1 u genetskim kodovima jedinki-roditelja uzajamno menjaju mesta, stvarajući pri tom 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 na slučajan način 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 značajno da utiče na performanse GA. Ako GA koristi binarno kodiranje i populacija nema nekorektnih jedinki, obično se implementira operator *proste mutacije* koji na slučajan način (sa unapred zadatom verovatnoćom p_{mut}) bira neke bitove jedinke i mutira ih. 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 $B(N_{bit}, p_{mut})$, gde je N_{bit} 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 $\{0,..., N_{bit}-1\}$. U slučaju dugačkog genetskog koda može doći do greške izračunavanja broja N_{mut}. Ova pojava se dešava ako je N_{bit} veliko, a p_{mut} blisko nuli, pa u proizvodu $\binom{N_{bit}}{k} * {p_{mut}}^{k} * (1-p_{mut})^{N_{bit}-k}$ prvi činilac je mnogo veliki, a drugi mnogo mali. Ovaj problem je poznat u teoriji verovatnoće i rešava se korišćenjem centralne granične teoreme, pri čemu se slučajna promenljiva X_{ind} aproksimira pomoću normalne raspodele $N(N_{bit}*p_{mut}; N_{bit}*p_{mut}*(1-p_{mut}))$. Kao i u prethodnom slučaju, koristimo inverznu funkciju funkcije normalne raspodele da bismo izračunali broj gena koje treba mutirati N_{mut}, a njihove pozicije određujemo na gorepomenuti način. Razvijena je varijanta ovog operatora za primenu na celoj populaciji, jer se pri mutaciji genetski kodovi svih jedinki mogu statistički 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 brojem bliskim jedinici (*geometric creep*) itd. Za oba pomenuta operatora mutacije potrebne vrednosti (*creep values*) su slučajne i mogu imati uniformnu, eksponencijalnu, Gausovu ili binomnu raspodelu (videti [BeD93a] i [BeD93b]).

Postoje slučajeve kada nije moguće direkno primeniti standardne genetske operatore ili njihove modifikacije. U drugim slučajevima, moguće ih je primeniti, ali daju relativno loše rezultate. To može biti posledica specifičnog načina kodiranja ili postojanja velikog broja ograničenja (constrained problems) koja se ne mogu sva implementirati u genetski kod. Tada se često primenjuju genetski operatori mutacije i ukrštanja zavisni od prirode problema. Ovakvi operatori mutacije i ukrštanja uglavnom 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 se mogu izvršavati 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 hab i drugih lokacijskih problema

Jedna od važnih primena genetskih algoritama je rešavanje NP-kompletnih 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 [Boz02] predložen je GA koncept za rešavanje klasičnog lokacijskog problema p-medijane koji koristi: celobrojnu reprezentaciju jedinki, nekoliko modifikovanih operatora ukrštanja, standardnu mutaciju i operator "invazije" koji ima za cilj povećavanje raznovrsnosti genetskog materijala. Dvorett u [Dv99] takođe rešava problem p-medijane genetskim algoritmom sa binarnom šemom kodiranja i novim operatorom ukrštanja koji koristi "princip kompatibinosti roditelja". U radu [Alk04], heuristika "najbliža četiri suseda" (The Nearest Four Neighbors-N4N) je implementirana u GA za rešavanja nekih problema na grafovima koji se odnose na izdvajanje skupa čvorova u cilju minimizacije ili maksimizacije funkcije cilja (tu spada i lokacijski problem p-medijane).

Za rešavanje problema particije skupa čvorova u datom grafu (clustering problems), u [Lor00] je predložen "konstruktivni" GA (Constructive GA-CGA) koji koristi kodiranje nad skupom {0,1 i #}, standardne operatore selekcije i ukrštanja, dok ulogu mutacije ima heuristika zamene. Koncept CGA je u [Lor00] primenjen na problem p-medijane ograničenih kapaciteta, problem k-bojenja grafova (k-colouring problem), 2-D sečenja (2-D cutting problem), minimalnog sečenja (MinCut problem), itd.

Poznati lokacijski problem p-centra takođe je rešavan primenom genetkih algoritama. U [Mih03] opisan je genetski koncept sa celobrojnom reprezentacijom jedinki, turnirskom selekcijom i standardnim operatorom mutacije. Za svaku jedinku-roditelja formira se lista gena koji se mogu kopirati u kod drugog roditelja a da ne dođe do dupliranja, tako da operator ukrštanja koji se realizuje zamenom slučajno izabranih gena iz svake liste, čuva korektnost jedinki.

I neki od domaćih istraživača su uspešno primenili unapređene verzije GA na prost lokacijski problem [Kra96a], [Kra98], [Kra00] i [Kra01], problem dizajniranja mreže [Kra00] i [Kra02], problem selekcije indeksa [Kra00] i [Kra03]. U navedenim radovima su predložene sekvencijalne/paralelne GA implementacije koje sadrže fleksibilno realizovane razne varijante genetskih operatora selekcije, ukrštanja i mutacije. Osim njih, dato je i nekoliko funkcija prilagođenosti, razni kriterijumi zaustavljanja GA i različite politike zamene generacija. Sve zajedničke GA promenljive su grupisane u strukturu, pa je laka dopuna funkcijama koje zavise od prirode samog problema, što obezbeđuje relativno jednostavno rešavanje novih problema.

Problem 2-povezanosti grafova (biconnectivity augmentation problem) je u [Lju00] rešavan evolutivnim konceptom, čije su karakteristike prilagođene prirodi problema. U [Lju00a] je za problem ivične 2-povezanosti grafova (edge-biconnectivity augmentation problem) predložena hibridizacija genetskog algoritma iz [Lju00] sa efikasnom heuristikom koja popravlja nekorektne jedinke u GA populaciji. Problem čvorovske 2-povezanosti grafova (vertex-biconnectivity augmentation problem) se u [Lju01] rešava hibridnim GA koji koristi stohastičku tehniku penjanja (stohastic hill-climbing procedure) u okviru modifikovanih genetskih operatora. Isti problem je u [Lju02] rešavan efikasnijim hibridnim GA, koji uključuje efikasnu tehniku preprocesiranja i brzu heuristiku

lokalnog pretraživanja. Preprocesiranjem se značajno smanjuje dimenzija prostora pretrage, tako da se i genetski operatori i heuristika izvršavaju znatno efikasnije. Pošto se predložena heuristika primenjuje na jedinku pre njenog uključivanja u populaciju, nema pojave nekorektnih jedinki.

Kombinacija GA sa egzaktnom BnC metodom za rešavanje problema čvorovske 2-povezanosti grafova opisana je u [Lju04]. Za veće dimenzije problema predložena je hibridizacija GA sa egzaktnom metodom grananja, sečenja i procene (Branch-and-Cut-and-Price), koja se pokazala uspešnijom od ostalih egzaktnih i heurističkih pristupa na svim testiranim instancama problema.

U [Sta04] i [Sta06a] predložene su dve efikasne evolutivne metaheuristike HGA1 i HGA2 za rešavanje Diskretno uređenog problema medijane (DOMP), koji je uopštenje nekoliko klasičnih diskretnih lokacijskih problema (p-medijane, p-centra, (k_1+k_2)-uređene sredine, μ-centdian problema, itd). Obe heuristike predstavljaju hibridizaciju genetskog algoritma i heuristike brze zamene (fast interchange heuristic) i koriste dva različita načina kodiranja: binarno u HGA1 i celobrojno u HGA2. Testiranja i poređenja na instancama problema velikih dimenzija (do n=900 čvorova) pokazuju da je heuristika HGA1 efikasnija od HGA2, dok obe heuristike daju bolje rezultate od ostalih metoda za rešavanje DOMP-a. U [Mar03] autori daju proširenje genetskog koncepta predloženog u [MV96] (za problem p-centra) i kombinuju ga sa gramzivom heuristikom koja služi za generisanje početne GA populacije. Predloženi hibridni GA je primenjen za rešavanje ne samo problema p-centra, već i za DOMP, ali je lošiji u poređenju sa gorepomenutim HGA1 i HGA2 konceptima.

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 efikasan za rešavanje problema sa malo ograničenja i relativno složenom funkcijom cilja [Ree93].

Neki od hab lokacijskih problema su takođe rešavani primenom genetskih algoritama: [AbH98a], [AbH99], [Ern04a]. U [AbH98a] i [AbH99] opisana je hibridizacija genetskog algoritma i tabu pretraživanja za hab lokacijski problem neograničenih kapaciteta sa jednostrukim alokacijama. Dva efikasna genetska algoritma za rešavanje problema p-hab medijane/centra neograničenih kapaciteta sa višestrukim alokacijama predloženi su redom u [Sta07] i [Kra06a]. Oba algoritma koriste binarno kodiranje i nove genetske operatore koji su prilagođeni prirodi problema. Testiranja na standardnim hab instancama pokazuju da oba genetska algoritma dostižu sva poznata optimalna rešenja iz literature i takođe daju nova rešenja na hab instancama problema realnih dimenzija $n\leq 200$ i $p\leq 20$.

U [Kra06b] opisana su dva genetska algoritma sa različitim načinom kodiranja (binarno i celobrojno) i odgovararajućim modifikovanim genetskim operatorima za rešavanje problema p-hab medijane neograničenih kapaciteta sa jednostrukim alokacijama. Obe metode su uspešno primenjene na hab instancama velikih dimenzija ($n\leq 200$ i $p\leq 20$), pri čemu GA sa binarnim kodiranjem dostiže nešto bolja rešenja u odnosu na GA sa celobrojnim kodiranjem. Genetski algoritam u [Kra05] takođe daje dobre rezultate za hab lokacijski problem neograničenih kapaciteta sa višestrukim alokacijama.

Još neki genetski koncepti primenjeni na hab lokacijske probleme opisani su u [Pér00], [Pér04], [Top05] i [Fil06].

2 PROBLEM P-HAB CENTRA NEOGRANIČENIH KAPACITETA SA JEDNOSTRUKIM ALOKACIJAMA

2.1 Matematička formulacija problema

U literaturi postoje različite formulacije problema *p*-hab centra neograničenih kapaciteta sa jednostrukim alokacijama (Uncapacitated Single Allocation p-hub Center Problem-USApHCP). U ovom radu koristi se linearna formulacija data u [Kara00] koja ima manje promenljivih u odnosu na ostale poznate formulacije, što je čini pogodnom za rešavanje velikih dimenzija problema.

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}, pri čemu je $C_{ij} = C_{ji}$ i pretpostavlja se da rastojanja zadovoljavaju nejednakost trougla: $C_{ij} \leq C_{ik} + C_{kj}$, za svako *i,j,k=1,...,n.* Ako ne postoji grana od čvora *i* do čvora *j,* uzima se $C_{ij} = \infty$. Problem ima šemu jednostruke alokacije, što znači da svaki korisnik/snabdevač može biti pridružen samo jednom habu. Broj habova koje treba locirati je unapred zadat (*p*). Binarna promenljiva Z_{ik} uzima vrednost 1 akko je čvor *i* pridružen habu *k*, u suprotnom ima vrednost 0. Promenljiva *z* predstavlja funkciju cilja, odnosno troškove transporta.

Protok od čvora-snabdevača *i* do čvora-korisnika *j* sastoji se od tri komponente: transfer od snabdevača *i* do prvog haba *k*, transport između habova *k* i *l* i distribucija od haba *l* do korisnika *j*. Cena transporta jedinice količine robe duž puta $i \rightarrow k \rightarrow l \rightarrow j$ računa se kao $\chi * C_{ik} + \alpha * C_{kl} + \delta * C_{lj}$. Parametri χ, α i δ redom označavaju troškove (cenu) kolekcije, transfera i distribucije robe po jedinici količine. Ukoliko su $\chi = \delta = 1$, tada $1-\alpha$ predstavlja koeficijent uštede za transport između habova.

Koristeći gornju notaciju, USApHCP se matematički može zapisati kao:

$$\min z \tag{2.1}$$

uz uslove

$$\sum_{k=1}^{n} Z_{kk} = p \tag{2.2}$$

$$\sum_{k=1}^{n} Z_{ik} = 1, \qquad \text{za svako i=1,…,n} \tag{2.3}$$

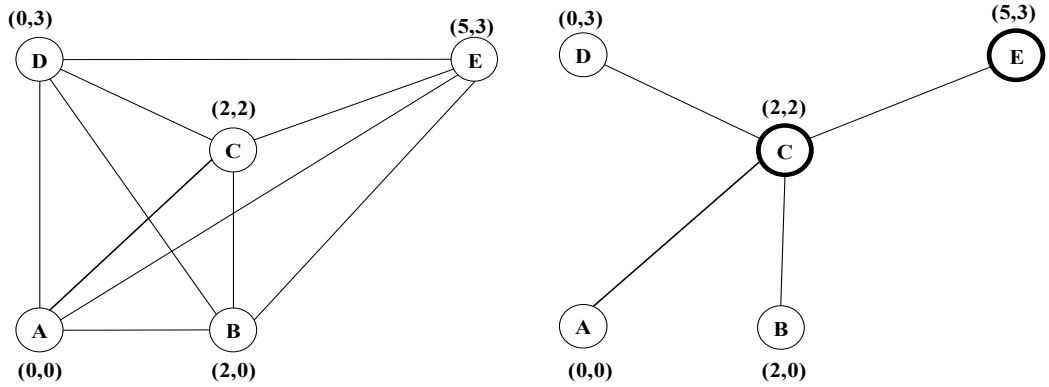
$$Z_{ik} \leq Z_{kk}, \qquad \text{za svako i, k=1,…,n} \tag{2.4}$$

$$z \geq \sum_{k=1}^{n} (\chi C_{ik} + \alpha C_{kl}) Z_{ik} + \delta C_{lj} Z_{jl}\ , \quad \text{za svako i, j ,l=1,…,n} \qquad (2.5)$$

$$Z_{ik} \in \{0,1\}\ , \quad \text{za svako I, k=1,…,n} \qquad (2.6)$$

Cilj USApHCP je minimizacija maksimalnih troškova transporta između bilo koja dva čvora u mreži (2.1). Uslov (2.2) određuje tačan broj habova koje treba uspostaviti (*p)*. Ograničenja (2.3) i (2.4) obezbeđuju da svaki korisnik/snabdevač bude pridružen tačno jednom, prethodno lociranom habu. Donja granica za promenljivu z funkcije cilja je data uslovom (2.5). Konačno, (2.6) ukazuje na binarnu reprezentaciju promenljivih Z_{ik}.

USApHCP je NP-težak, što je dokazano u [Kara99b] i [Ern04b]. Za unapred zadat skup habova, dobija se potproblem optimalnog pridruživanja habova ne-hab čvorovima koji je takođe NP-težak [Ern04a]. Ako u istom problemu pretpostavimo da nema redukcije cene transporta između habova *(α=0)*, ponovo dobijamo NP-kompletan potprobem [Ern04b].



Slika 2.1 Hab mreža sa n=5, p=2 i *χ=δ=1,* α=0.25

Primer 1: Na levoj strani slike 2.1 data je mreža sa pet čvorova A, B, C, D, E (čvorovi su zadati Dekartovim koordinatama u ravni) od kojih je potrebno izabrati dva haba. Odgovarajuća matrica rastojanja je:

$$C = \begin{bmatrix} 0 & 2 & 2\sqrt{2} & 3 & \sqrt{34} \\ 2 & 0 & 2 & \sqrt{13} & 3\sqrt{2} \\ 2\sqrt{2} & 2 & 0 & \sqrt{5} & \sqrt{10} \\ 3 & \sqrt{13} & \sqrt{5} & 0 & 5 \\ \sqrt{34} & 3\sqrt{2} & \sqrt{10} & 5 & 0 \end{bmatrix}$$

Neka je količina protoka od čvora-snabdevača do čvora-korisnika ista i jednaka 1. Optimalno rešenje USApHCP koje se dobija za vrednosti parametara χ=1,α=0.25 i δ=1 dato je na desnoj strani slike 2.1. Habovi su uspostavljeni u čvorovima **C** i **E**, dok su ne-hab čvorovi A, B i D pridruženi habu **C**. Svaki od habova **C** i **E** je pridružen samom sebi. Troškovi transporta između svakog para čvorova u mreži su: "A-**C**-A" $2\sqrt{2}$ +2$\sqrt{2}$ = 5.657, "A-**C**-B" $2\sqrt{2}$ +2=4.823, "A-**C**" $2\sqrt{2}$ =2.823, "A-**C**-D" $2\sqrt{2}+\sqrt{5}$ =5.064, "A-**C**-**E**" $2\sqrt{2}+\sqrt{10}$ *0.25=3.619, "B-**C**-B" 2+2=4, "B-**C**" 2, "B-**C**-D" 2+$\sqrt{5}$ =4.236, "B-**C**-**E**" 2+$\sqrt{10}$ *0.25=2.791, "**C**-**C**" 0, "**C**-

D" $\sqrt{5}$, "**C**-**E**" $\sqrt{10}$*0.25=0.791, "D-**C**-D" $2\sqrt{5}$=4.472, "D-**C**-**E**" $\sqrt{5}+\sqrt{10}$*0.25=3.027, "**E**-**E**" 0 (hab čvorovi u stazama su boldovani). Vrednost funkcije cilja (maksimalni transportni troškovi) se dobija za transport "A-**C**-A" iznosi 5.657.

2.2 Postojeći načini rešavanja USApHCP

Sudeći po zastupljenosti u literaturi, problemi p-hab centra su manje proučavani u poređenju sa problemima p-hab medijane. Dve najstarije definicije problema p-hab centra nalaze se u [OK91] i [Cam94b]. U poslednjem, Campbell daje formulaciju sa $O(n^4)$ 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.

U [Kara98] predloženo je nekoliko linearizacija modela USApHCP iz [Cam96], kao i novi linearni model problema koji koristi jednu realnu i n^2 binarnih promenljivih i $n^3 + n^2 + n + 1$ uslova. Novi model USApHCP se pokazao znatno pogodnijim u rešavanju standardnih hab problema sa $n \leq 25$ čvorova i $p \leq 5$ habova u odnosu na ostale linearizacije problema.

Pamuk i Sepil [Pam01] kombinuju tri različite varijante heuristike jednostruke realokacije i tri metode tabu pretraživanja. Heuristika iz tekućeg rešenja isključuje jedan i uključuje drugi hab, a u svakom koraku traži najbolje alokacije ne-hab čvorova uspostavljenim habovima. Kada dalje poboljšanje rešenja nije moguće, tabu pretraživanje usmerava algoritam u nove regione dopustivog prostora rešenja i tako smanjuje mogućnost da algoritam završi u lokalnom optimumu. Gramziva heuristika lokalne pretrage na kraju dodatno poboljšava dobijeno najbolje rešenje. U radu su prikazana rezultati testiranja devet hibridnih heuristika na hab instancama malih dimenzija $n \leq 25$ i $p \leq 5$.

U [Ern04a] dokazana je NP-kompletnost problema hab centra sa jednostrukim alokacijama (Uncapacitated Single Allocation Hab Center Problem-HCSAP), kao potproblema USApHCP koji se dobija fiksiranjem skupa habova. Predloženo je nekoliko novih LP formulacija HCSAP i pet heuristika za njegovo rešavanje. U cilju poboljšanja kvaliteta rešenja, svaka heuristika je hibridizovana sa metodom pretrage prostora (problem space search method), koji je zasnovan na evolutivnom konceptu. Rezultati testiranja na instancama sa $n \leq 100$ čvorova pokazuju da je hibridizacija predložene "heuristike najbližeg haba" (closest hub heuristic) sa evolutivnim PSS algoritmom najuspešnija u smislu kvaliteta dobijenih rešenja i vremena izvršavanja.

Linearizacija modela USApHCP, zasnovana na konceptu "radijusa habova", data je u [Ern04b]. Nova formulacija koristi $n^2 + n + 1$ promenljivih, od kojih su n^2 binarne a ostale celobrojne, i $3n^2 + n + 1$ uslova, što je manje u odnosu na model iz [Kara98]. U radu su opisana dva heuristička pristupa za rešavanje USApHCP. Testiranja na standardnim instancama sa $n \leq 100$ čvorova i $p \leq 10$ habova su pokazala da je nova formulacija USApHCP efikasnija za rešavanje problema većih dimenzija u odnosu na formulaciju iz [Kara98]. Ni jedna od predloženih heuristika nije dala rešenja zadovoljavajućeg kvaliteta.

U [Sol05] primenjena je hibridizacija genetskog algoritma i gramzive heuristike lokalnog pretraživanja (greedy local search heuristic). Predloženi genetski algoritam koristi celobrojno kodiranje i standardne genetske operatore. Gramziva heuristika (preuzeta iz [Pam01]) ima za cilj poboljšanje alokacije ne-hab čvorova kad god operator mutacije izbaci jedan, a ubaci drugi hab čvor

iz genetskog koda. Predložena hibridna metoda je testirana samo na instancama rešenja sa $n \leq 25$ i $p \leq 10$, ali čak i na instancama malih dimenzija dobijena najbolja rešenja odstupaju od optimalnih rešenja poznatih iz literature.

2.3 Predloženi hibridni genetski algoritmi za USApHCP

Postojeći genetski koncepti za rešavanje nekih drugih hab lokacijskih problema sa višestrukim alokacijama opisani u [Sta04], [Kra05], [Kra06a] i [Sta07] ne mogu se primeniti na hab lokacijske probleme sa jednostrukim alokacijama. Jedna od suštinskih razlika između pomenutih hab modela se ogleda u sledećem. Kod problema sa višestrukim alokacijama, potproblemi dobijeni fiksiranjem skupa habova su obično polinomske složenosti. Nasuprot tome, u slučaju problema sa jednostrukim alokacijama razmatranih u ovom radu čak su i potproblemi sa fiksiranim habovima NP-teški.

Iako se genetski algoritam za vrlo sličan problem USApHMP opisan u [Kra06b], može prilagoditi i za rešavanje USApHCP, praktični rezultati su daleko od optimalnih, čak i za probleme malih dimenzija. Zbog toga nije bila dovoljna modifikacija gore navedenih genetskih koncepata, već se moralo krenuti od početka u svim ključnim segmentima, kao što su kodiranje, funkcija cilja i genetski operatori.

U ovoj sekciji opisana su dve evolutivne metaheuristike HGA1 i HGA2 koje predstavljaju hibridizaciju genetskog algoritma i efikasne heuristike lokalnog pretraživanja za poboljšanje kvaliteta rešenja (Fast Improving Heuristic-FIH). Osnovna razlika je u načinu kodiranja, što ima za posledicu različite genetske operatore i ostale aspekte GA. Za razliku od HGA1 reprezentacije koja isključuje mogućnost pojavljivanja nekorektnih jedinki, u slučaju HGA2 reprezentacije bilo je neophodno razviti nove genetske operatore koji čuvaju korektnost jedinki u populaciji. Oba algoritma koriste FIH heuristiku u cilju poboljšanja kvaliteta GA rešenja, kao i GA keširanje koje značajno smanjuje vreme izvršavanja algoritama.

Na slici 2.2 prikazana je osnovna struktura HGA implementacija:

```
Unošenje_Ulaznih_Podataka();
Generisanje_Početne_Populacije();
while not Kriterijum_Zaustavljanja_GA()do
    for i=1 to N_pop do
      obj[i] = Funkcija_Cilja(i);
      Heuristika_Lokalnog_Pretraživanja(i);
    endfor
    Funkcija_Prilagođenosti();
    Selekcija();
    Ukrštanje();
    Mutacija();
endwhile
Štampanje_Izlaznih_Podataka();
```

Slika 2.2 Osnovna struktura HGA

N_{pop} predstavlja broj jedinki u populaciji, dok je *obj*[*i*] vrednost funkcije cilja *i*-te jedinke

2.3.1 Reprezentacija jedinki

Genetski kod jedinke u HGA1 implementaciji sastoji se od dva segmenta. Prvi segment je string dužine *p,* čiji elementi (geni) odgovaraju indeksima uspostavljenih habova. Elementi ovog stringa uzimaju vrednosti iz skupa {0,1,...,n-1}, u skladu sa numeracijom čvorova u mreži. Drugi segment genetskog koda je string sa *n-p* gena, pri čemu svaki gen odgovara jednom ne-hab čvoru i sadrži indeks uspostavljenog hab čvora kojem je dati ne-hab čvor alociran.

Za svaki ne-hab čvor formira se niz uspostavljenih habova, koji se zatim sortira u neopadajućem poretku prema rastojanjima habova do datog ne-hab čvora. Svaki hab čvor je očigledno pridružen samom sebi. Motivacija za kreiranje i sortiranje niza habova, tzv. "uređenje najbližeg suseda" potiče iz činjenice da optimalno rešenje problema obično ne podrazumeva alokaciju svakog ne-hab čvora njemu najbližem habu. Indeksi "bližih" habova često se javljaju u optimalnom rešenju, dok su indeksi "daljih" habova retki. Zbog toga je GA pretraga usmerena ka "bližim" habovima, dok se "dalji" habovi razmatraju sa malom verovatnoćom. Sortiranjem niza uspostavljenih habova prema rastojanjima od tekućeg ne-hab čvora, obezbeđuje se da "bliži" habovi imaju veći prioritet pri alokacijama ne-hab čvorova.

Genetski kod svake jedinke u HGA2 sadrži *n* gena, pri čemu svaki gen odgovara jednom čvoru u mreži. Prvi bit svakog gena uzima vrednost 1, ako je dati čvor uspostavljeni hab, 0 ako nije. Preostali bitovi se odnose na alokaciju datog čvora nekom od uspostavljenih habova, pri čemu je svaki hab alociran samom sebi. Koristeći vrednosti prvih bitova u svakom od *n* gena, formira se niz uspostavljenih habova koji se za svaki ne-hab čvor sortira koristeći "uređenje najbližeg suseda", kao u slučaju HGA1 kodiranja.

2.3.2 Funkcija cilja

U slučaju HGA1 reprezentacije, indeksi uspostavljenih habova dobijaju se iz prvog segmenta genetskog koda. Ako se neki od indeksa ponovo pojavi u genetskom kodu, uzimamo sledeći indeks koji se nije javljao ranije. U slučaju da su svi naredni indeksi već iskorišćeni, uzimamo prvi prethodni "slobodan" indeks. Pošto je broj habova *p* značajno manji od broja čvorova u mreži *n*, brzo se nalazi slobodan indeks koji će zameniti duplirani. Na ovaj način izbegavamo problem ponavljanja indeksa, tako da iz prvog dela genetskog koda čitamo tačno *p* različitih indeksa uspostavljenih habova. Vremenska složenost opisanog koraka je $O(p^2)$.

Ukoliko su podaci u instanci potpuno nezavisni (nisu uređeni ni po kakvom pravilu), tada je gorepomenuti postupak traženja sledećeg/prethodnog "slobodnog" indeksa ekvivalentan sa slučajnim traženjem "slobodnog" indeksa. Međutim, treba očekivati je da ulazni podaci u instanci nisu potpuno nezavisni (da im je koeficijent korelacije veći od nule), pa je gorepomenuti postupak u prednosti u odnosu na slučajno traženje "slobodnog" indeksa.

Za svaki ne-hab čvor dobijeni niz indeksa habova sortira se u neopadajućem poretku rastojanja od datog čvora. Iz drugog dela genetskog koda za svaki ne-hab čvor uzima se odgovarajući gen koji se odnosi na alokaciju datog čvora nekom od habova. Ako dati gen ima vrednost *r*, tekući ne-hab čvor je pridružen *r*-tom habu iz sortiranog niza habova, pri čemu $r \in \{0, 1, ..., p-1\}$. Sortiranje niza uspostavljenih habova vrši se *n-p* puta za svaku jedinku u svakoj generaciji

genetskog algoritma, što zahteva O((n-p)*p*log p) operacija. Pošto je p relativno malo u testiranim instancama problema ($p \leq 20$), sortiranje ne utiče značajno na ukupno CPU vreme algoritma. Iako je ukupno vreme izvršavanja nešto duže, testiranja su pokazala da se korišćenjem sortiranja postižu značajna poboljšanja kvaliteta GA rešenja.

Nakon opisanog postupka lokacije habova i alokacija ne-hab čvorova, vrednost funkcije cilja se računa jednostavnim sumiranjem rastojanja snabdevač-hab, hab-hab i hab-korisnik pomnoženih odgovarajućim parametrima χ, α i δ.

Primer 2*:* Za n=5, p=3 (*i*,*j* = 0, 1, ..., *n*-1) i matricu rastojanja

$$C = \begin{bmatrix} 0 & 12 & 4 & 7 & 8 \\ 12 & 0 & 9 & 11 & 6 \\ 4 & 9 & 0 & 8 & 17 \\ 7 & 11 & 8 & 0 & 12 \\ 8 & 6 & 17 & 12 & 0 \end{bmatrix},$$

genetski kod

```
1 1 4 | 2 0
```

označava da su čvorovi 1, 2 i 4 izabrani za habove. Indeks 1 je dupliran u genetskom kodu, pa uzimamo sledeći "slobodan" indeks (2). Za ne-hab čvor 0, sortirani niz indeksa uspostavljenih habova je: 2, 4, 1 sa rastojanjima 4, 8, 12 redom. U slučaju ne-hab čvora 3, odgovarajući sortirani niz habova je: 2, 1, 4 sa rastojanjima 8, 11, 12. Iz drugog segmenta genetskog koda čitamo da je čvor 0 pridružen habu koji je treći po udaljenosti (čvor 1), dok je čvor 3 alociran najbližem habu (čvor 2). Svaki od habova 1, 2 i 4 je pridružen samom sebi. Za α=0.2 i $\chi = \delta = 1$, funkcija cilja je: 1* (3*12+0.75*9+2*8) = 58.75.

Kod HGA2 indeksi uspostavljenih habova dobijaju se iz prvih bitova svakog od *n* gena, kao što je prethodno objašnjeno. U ovom slučaju očigledno nemamo problem ponavljanja indeksa uspostavljenih habova. Za svaki ne hab čvor, dobijeni niz habova se takođe uređuje po principu "najbližeg suseda", kao kod HGA1.

Primer 3. Uzimajući iste vrednosti za n i p i istu matricu rastojanja C iz primera 2, genetski kod

```
02|10|10|00|10
```

predstavlja HGA2 reprezentaciju istog rešenja koje je dato u primeru 2.

Prvi bitovi u svakom od pet gena (0, 1, 1, 0, 1) označavaju uspostavljene habove (1, 2 i 4), dok ostali delovi gena (2, 0, 0, 0, 0) ukazuju na pridruživanja ne-hab čvorova habovima.

2.3.3 Heuristika lokalnog pretraživanja

Rešenje dobijeno genetskim algoritmom se dodatno poboljšava efikasnom heuristikom lokalnog pretraživanja- FIH.

U opisu implementirane heuristike koristi se sledeće oznake:

- *imax, jmax*: indeksi čvora-snabdevača i čvora-korisnika u tekućem najboljem rešenju;
- *h*(*imax*), *h*(*jmax*): indeksi habova koji su pridruženi čvorovima *imax* i *jmax*;
- *hi, hj*: indeksi habova;
- *obj*, obj_{new}: tekuća i potencijalna vrednost funkcije cilja;

- *stop*: logička promenljiva koja označava da nije postignuto poboljšanje najboljeg rešenja.
- promenljiva *switched* označava da li je u tekućem koraku poboljšano tekuće rešenje. Ukoliko nije dobijeno poboljšanje, heuristika završava rad.

Primenjena FIH heuristika ima sledeću osnovnu šemu:

```
repeat
    switched := 0;
   for hi:=1 to p do
      for hj:=1 to p do
       if(χC_imax,hi+α*C_hi,hj+δC_hj,jmax<χC_imax,h(imax)+α*C_h(imax),h(jmax)+δC_h(jmax),jmax)
             then
                 h(imax) = hi;
                 h(jmax) = hj;
                 switched := 1;
       endif
      endfor
     endfor
      stop := true;
      if (switched)
      then
        obj_new := Funkcija_Cilja();
        if(obj_new < obj)
        then
            obj := obj_new;
            stop := false;
        endif
      endif
 until (stop);
```

Opisana heuristika se primenjuje na svaku jedinku u populaciji u svakoj generaciji genetskog algoritma, pre primene genetskih operatora. Kako je vremenska složenost dva ugnježdena **for** ciklusa $O(p^2)$ manja od složenosti funkcije cilja O((n-p)*p*log p), složenost opisane FIH u svakom repeat-until ciklusu je O((n-p)*p*log p).

2.3.4 Genetski operatori

2.3.4.1 Selekcija

Oba predložena HGA koncepta koriste fino-gradiranu turnirsku selekciju (FGTS), opisanu u [Fil98] i [Fil03], 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 $\lfloor F_{tour} \rfloor$, dok se drugi tip realizuje k_2 puta sa $\lceil F_{tour} \rceil$ jedinki-učesnika turnira ($\lfloor x \rfloor = r$ i $\lceil x \rceil = s \Leftrightarrow r \le x \le s$ i $r,s \in Z$, $x \in R$), te je $F_{tour} \approx (k_1 * \lfloor F_{tour} \rfloor + k_2 * \lceil F_{tour} \rceil) / (k_1 + k_2)$. U obe HGA implementacije parametar F_{tour} uzima vrednost 5.4, jer se u praksi ova vrednost pokazala optimalnom.

Ako se, na primer, FGTS operator primenjuje na 50 neelitnih jedinki, u svakoj generaciji realizuju se dva tipa turnira veličine 5 i 6 u kojima učestvuju k_1=30 i k_2=20 ne-elitnih jedinki respektivno.

Vreme izvršavanja FGTS operatora je O(N_{nel} *F_{tour}), gde je N_{nel} broj ne-elitnih jedinki koje učestvuju u selekciji. U praksi, parametri F_{tour} i N_{nel} se smatraju konstantnim (ne zavise od veličine problema), što daje konstantnu vremensku složenost. Detaljne informacije o FGTS i njegovim teoretskim aspektima mogu se naći u [Fil98], [Fil00], [Fil03] i [Fil06].

2.3.4.2 Ukrštanje

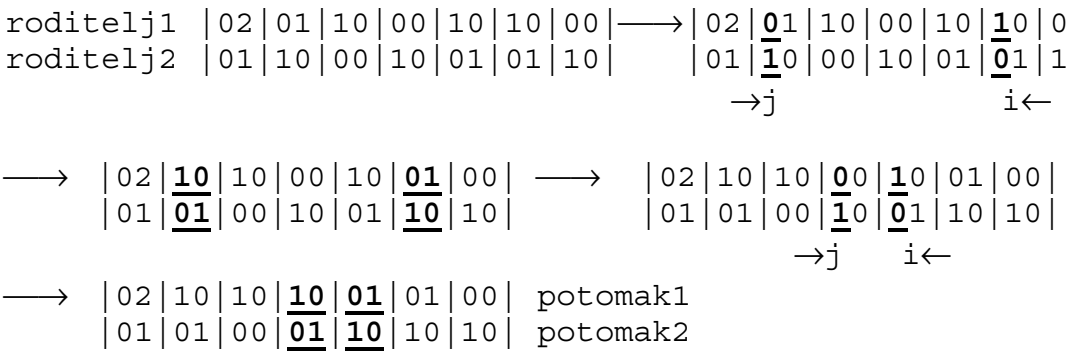
Posle selekcije parova jedinki-roditelja, na njih se primenjuje operator ukrštanja stvarajući dve jedinke-potomka. Osnovni tip ukrštanja razmenjuje delove genetskih kodova roditelja počevši od slučajno izabrane tačke ukrštanja. S obzirom na prirodu kodiranja koje koriste HGA1 i HGA2, standardno jednopoziciono ukrštanje se ne može primeniti ni u jednom ni u drugom slučaju. Genetski kod jedinki-roditelja u algoritmu HGA1 sastoji se od dva segmenta različite prirode, tako u ovoj implementaciji koristimo dvostruko jednopoziciono ukrštanje koje daje dve jedinke-potomke. U svakom segmentu genetskog koda roditelja na slučajan način biramo "tačku ukrštanja" a zatim jedinke razmenjuju sve gene počevši od izabrane pozicije (slika 2.3). Pošto se oba segmenta "nezavisno" ukrštaju, broj uspostavljenih habova ostaje isti, a primenjeni način kodiranja uspešno razrešava problem dupliranja indeksa habova, tako da opisani operator ukrštanja čuva korektnost jedinki. Eksperimenti sa različitim vrednostima verovatnoće ukrštanja pokazali su da GA daje najbolje rezultate ako se ukrštanje realizuje sa verovatnoćom p_{cross}=0.85, pa je ova vrednost uzeta i u HGA1 implementaciji. To znači da oko 85% jedinki iz populacije učestvuje u stvaranju jedinki-potomaka, dok se u oko 15% slučajeva ukrštanje ne izvršava i jedinke-potomci su zapravo identične jednom od roditelja.

```
roditelj1   0 1 1 3| 2 0 0 1 1 0 ⟶  0 1 1 3| 2 0 0 1 1 0
roditelj2   2 2 5 9| 0 2 2 1 0 0      2 2 5 9| 0 2 2 1 0 0
                                          i            j
⟶  0 1 5 9| 2 0 0 1 0 0 potomak1
    2 2 1 3| 0 2 2 1 1 0 potomak2
```

Slika 2.3 Modifikovani operator ukrštanja u HGA za n=10, p=4

U slučaju HGA2 kodiranja, prosta zamena segmenata genetskog koda jedinki-roditelja može proizvesti nekorektne potomke za USApHCP. Primenom standardnog ukrštanja, broj jedinica na prvoj poziciji svakog gena u genetskom kodu jedinki-potomaka može postati različit od p, iako su oba roditelja bila korektna. Da bi se ovaj problem prevazišao, standardni operator ukrštanja je modifikovan u HGA2 implementaciji (slika 2.4). Operator istovremeno prolazi kroz genetske kodove obe jedinke-roditelja s desna na levo tražeći gen *i* u kojem prvi roditelj ima vrednost 1 na prvoj poziciji datog gena, a drugi 0. Jedinke razmenjuju cele gene na nađenoj poziciji *i* (koja predstavlja tačku ukrštanja), a zatim se isti proces sprovodi počevši od leve strane genetskog koda na desno. Operator sada traži gen *j* čiji prvi bit ima vrednost 0 u kodu prvog roditelja, a 1 u kodu drugog. Posle zamene celog gena na *j*-toj poziciji

broj uspostavljenih habova ostaje nepromenjen. Opisani proces (korak) se ponavlja do ispunjenja uslova $j \geq i$. Modifikovani operator ukrštanja u HGA2 takođe se izvršava sa verovatnoćom p_{cross}=0.85.



Slika 2.4 Modifikovani operator ukrštanja u HGA2 za n=7, p=3

2.3.4.3 Mutacija

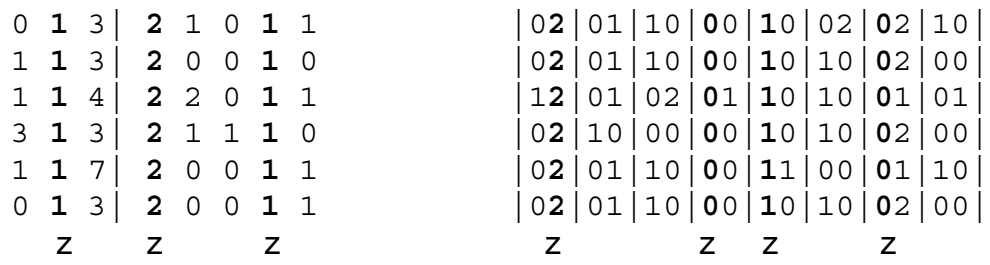
Jedinke-potomci generisane operatorom ukrštanja podležu operatoru mutacije, koji menja vrednost slučajno izabranog gena u genetskom kodu jedinke. Standardni operator mutacije je modifikovan u skladu sa tipom kodiranja i osobinama problema u oba HGA koncepta.

S obzirom na različitu prirodu segmenata genetskog koda HGA1 implementacije, trebalo bi uzeti različite vrednosti nivoa mutacije za prvi i drugi segment koda. Pošto su geni drugog segmenta uređeni po strategiji "najbližeg suseda", mutacija bitova na različitim pozicijama u svakom genu ima različit uticaj na promenu poretka u nizu uspostavljenih habova. Na primer, mutacija prvog bita u genu menja poredak u nizu habova za $\pm$ 1. Promena bita $0 \rightarrow 1$ ima uticaj +1, što znači da tekući ne-hab čvor pridružujemo sledećem habu iz niza uspostavljenih habova. Promena $1 \rightarrow 0$ rezultuje pridruživanjem datog čvora prethodnom habu (-1 uticaj). Mutacija bita na drugoj poziciji gena ima uticaj $\pm$ 2 na poredak habova u nizu, na trećoj poziciji $\pm$ 4, na četvrtoj $\pm$ 8, itd. Pošto se dejstvo mutacije bitova povećava sa koeficijentom 2, odlučili smo se za smanjenje nivoa mutacije bitova za isti faktor idući kroz svaki gen drugog segmenta. Imajući u vidu gore navedene činjenice, osnovni nivo mutacije u HGA1 implementaciji je izabran na sledeći način:

- Svi bitovi prvog segmenta genetskog koda imaju isti nivo mutacije 0.7/*n*.
- U svakom genu drugog segmenta, prvi bit je promenjen sa verovatnoćom 0.1/n. Svaki naredni bit istog gena se menja sa dva puta manjim nivoom mutacije u odnosu na svog bita-prethodnika (0.05/*n*, 0.025/*n*, ...).

Tokom izvršavanja GA može se desiti da (skoro) sve jedinke u populaciji imaju isti bit na određenoj poziciji. Takvi bitovi (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 bitovima, ali ne više od nekoliko puta.



Slika 2.5 Zaleđeni bitovi u HGA2 i HGA2 reprezentaciji za n=8, p=3

U HGA1 implementaciji, mutacija zaleđenih bitova u prvom segmentu genetskog koda se realizuje sa 2.5 puta većom verovatnoćom u odnosu na nezaleđene bitove, tj. 1.75/*n* umesto 0.7/*n*. U drugom segmentu koda, nivo mutacije svakog zaleđenog gena je 1.5 puta veći od odgovarajućeg osnovnog nivoa mutacije, što znači da se bitovi svakog zaleđenog gena mutiraju sa verovatnoćama 0.15/n, 0.075/n, ...itd. Za drugi segment genetskog koda koristimo niži osnovni nivo mutacije i nivo mutacije za zaleđene bitove, jer je tu poželjno imati veći broj nula (nula u genu označava da je dati ne-hab čvor pridružen svom najbližem habu). Svi navedeni nivoi mutacije su konstantni tokom svih generacija genetskog algoritma.

Dvofazni modifikovani operator mutacije sa zaleđenim bitovima implementiran je u HGA2. U svakom od *n* gena genetskog koda, osnovni nivo mutacije je:

- 0.4/*n* za bit na prvoj poziciji u genu,
- 0.1/*n* za bit na drugoj poziciji, dok se naredni bitovi u genu mutiraju se sa dva puta manjom verovatnoćom u odnosu na prethodni bit (0.05/*n*, za treći, 0.025/*n* za četvrti, itd).

Zaleđeni bitovi na prvoj poziciji svakog gena imaju 2.5 puta veći nivo mutacije u odnosu na osnovni nivo (1.0/*n* umesto 0.4/*n*). Verovatnoće mutacije zaleđenih bitova na ostalim pozicijama u genu 1.5 puta veće od odgovarajućih osnovnih vrednosti (*0.15/n* umesto 0.1/*n*, *0.075/n* umesto 0.05/*n, 0.0375/n* umesto 0.025/*n, ...*).

Operator mutacije u HGA2 za svaku mutiranu jedinku prebrojava i upoređuje brojeve promenjenih nula i jedinica u bitovima na prvoj poziciji svakog gena. U slučaju da se ti brojevi razlikuju, neophodno je dodatno mutirati odgovarajuće vodeće bitove (nule ili jedinice), da bismo ih izjednačili. Na ovaj način operator mutacije primenjen u HGA2 održava tačno p uspostavljenih habova i čuva korektnost jedinki.

2.3.5 Ostale karakteristike algoritma

U obe HGA implementacije, početna populacija broji 150 jedinki i generiše se na slučajan način. Ovaj pristup obezbeđuje maksimalnu raznovrsnost genetskog materijala i veći gradijent funkcije cilja tokom generacija u odnosu na generisanje korišćenjem nekih heuristika. Jedna trećina populacije (N_{nel}=50) se zamenjuje u svakoj generaciji, dok najboljih 100 jedinki direktno prolazi u

narednu generaciju, čuvajući visoko prilagođene gene populacije. Funkcije cilja elitnih jedinki se računaju samo u prvoj generaciji, pa njihov broj praktično ne utiče na vreme izvršavanja celog GA. Ovaj koncept je u literaturi poznat pod nazivom *stacionarni GA sa elitističkom strategijom* ([Kra00], [Kra01]).

U HGA1 konceptu, svaki gen u prvom segmentu genetskog koda jedinke generiše se korišćenjem uniformne raspodele na skupu {0, ... , n-1}. U skladu sa primenjenim "uređenjem najbližeg suseda" i minimizacijskim tipom funkcije cilja, potrebno je favorizovati pridruživanje "bližih" habova svakom ne-hab čvoru. Zbog toga je poželjno da drugi segment genetskog koda sadrži mnogo nula (svaka nula označava pridruživanje najbližeg haba tekućem korisniku/snabdevaču).

Kao u slučaju mutacije, u drugom segmentu genetskog koda "1-bitovi" imaju različit uticaj u zavisnosti od njihove pozicije u datom genu. Ako se nalazi na prvoj poziciji u genu, "1-bit" ima uticaj +1, na drugoj poziciji uticaj "1-bita" je +2, na trećoj + 4, na četvrtoj +8, itd. Pošto se uticaj "1-bita" povećava sa koeficijentom 2, verovatnoća generisanja "1-bita" smanjuje se sa istim faktorom idući kroz svaki gen drugog segmenta. Vrednost gena (videti sekciju 2.3.1) je, očigledno, jednaka sumi uticaja njegovih "1-bitova". Verovatnoća generisanja celog gena je proizvod verovatnoća generisanja "1-bitova" i "0-bitova" datog gena (verovatnoća "1-bita" = 1 - verovatnoća "0-bita"). Rastući poredak gena po njihovim vrednostima ne mora da odgovara neopadajućem poretku gena po odgovarajućim verovatnoćama generisanja. Zbog toga se geni uređuju tako da njihove verovatnoće generisanja obrazuju neopadajući niz. Ovo uređenje (sortiranje) gena se izvršava samo jednom, pošto su verovatnoće generisanja gena konstantne tokom generisanja početne populacije genetskog algoritma.

Iz navedenih razloga, odučili smo se za sledeću strategiju generisanja drugog segmenta genetskog koda:

- "1-bit" na prvoj bit poziciji u svakom genu pojavljuje se sa verovatnoćom *1.0*/*n.*
- Svaki naredni "1-bit" generiše se sa dvostruko manjom verovatnoćom u odnosu na svog "1-bit" prethodnika ($\frac{1}{2n}$ za drugu poziciju u genu, $\frac{1}{4n}$ za treću, $\frac{1}{8n}$ za četvrtu, itd).

Opisana strategija obezbeđuje da je u početnoj populaciji svaki ne-hab čvor jedinke-rešenja često pridružen najbližem ili "bliskom" habu, a retko "udaljenom" habu.

Sličan pristup primenjen je u generisanju početne populacije HGA2 implementacije:

- U cilju dobijanja što većeg broja korektnih jedinki u početnoj populaciji, verovatnoća genersanja "1-bita" na prvoj poziciji gena u genetskom kodu je postavljena na *p/n*.
- "1-bit" na drugoj poziciji gena pojavljuje se sa verovatnoćom 1.0/n, a svaki sledeći "1-bit" generiše se sa dva puta manjom verovatnoćom ($\frac{1}{2n}, \frac{1}{4n}, \frac{1}{8n}$, ...).

Iako je verovatnoća generisanja jedinica na vodećoj bit poziciji gena postavljena na *p/n,* u praksi ipak može doći do pojavljivanja nekorektnih jedinki u početnoj populaciji. Jedinke koje imaju $k \neq p$ jedinica na prvim bit pozicijama

gena popravljaju se do korektnih brišući/dodajući |*p*-*k*| jedinica na vodećim bit pozicijama počevši od gena sa kraja genetskog koda. Implementirani genetski operatori čuvaju fiksiran broj habova, tako da se nekorektne jedinke ne pojavljuju u narednim generacijama.

HGA1 metoda koristi šemu kodiranja koja rešava problem pojavljivanja nekorektnih jedinki u početnoj populaciji (videti sekciju 2.3.1). Genetski operatori imlementirani u HGA1 čuvaju fiksiran broj različitih indeksa uspostavljenih habova - *p*, tako da su jedinke u svakoj narednoj generaciji algoritma korektne.

Jedinke koje se više puta pojavljuju u populaciji uklanjaju se u svakoj generaciji postavljanjem prilagođenosti 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 obe HGA metode. 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 HGA1 i HGA2 implementaciji).

Vreme izvršavanja HGA1 i HGA2 se može poboljšati keširanjem [Kra99], [Kra01]. Izračunate vrednosti funkcije cilja jedinki se čuvaju u heš-red tabeli koja se kreira primenom tehnike najstarijeg korišćenog člana (Least Recently Used Strategy-LRU). Kada se tokom primene GA ponovo dobije isti genetski kod, umesto direktnog izračunavanja, vrednost funkcije cilja se uzima iz heš-tabele. U ovom konceptu GA broj keširanih vrednosti funkcije cilja u heš-red tabeli je ograničen na N_{cache}=5000.

2.4 Rezultati testiranja i poređenja

U ovoj sekciji predstavljeni su rezultati predloženih HGA1 i HGA2 implementacija. U literaturi postoji nekoliko heuristika za rešavanje USApHCP, ali poređenja su izvršena samo sa najboljom od njih [Ern04b]. Sva testiranja su izvedena na računaru sa AMD Sempron sa 1597MHz procesorom i 256 MB RAM memorije pod Linux (Knoppix 3.7) operativnim sistemom. Algoritam je kodiran u programskom jeziku C. Za testiranje HGA metoda korišćena su dva tipa standardnih ORLIB instanci [Bea96] za hab lokacijske probleme:

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. U svim AP instancama, parametri χ, α i δ uzimaju vrednosti 3, 0.75 i 2 respektivno. Matrica protoka nije simetrična i elementi na njenoj glavnoj dijagonali nisu obavezno nule, pošto polazni i odredišni poštanski region pošiljke može biti isti (sa istim poštanskim brojem). U ovim instancama

važi nejednakost trougla za rastojanja između poštanskih centara. Skup AP instanci se takođe može preuzeti iz [Bea96].

Parametri genetskog algoritma o kojima je bilo reči u prethodnoj sekciji, a koji su se pokazali robusnim i pogodnim za USApHCP su korišćeni u predloženim HGA implementacijama. Iste vrednosti parametara uzete su u HGA1 i HGA2, kako bi poređenje predloženih metoda bilo korektno. Maksimalni broj generacija je N_{gen}=500, a algoritam se takođe zaustavlja ako je najbolja jedinka ili najbolja vrednost funkcije cilja ostala nepromenjena tokom N_{rep}=200 uzastopnih generacija. Na svim testiranim instancama, ovi kriterijumi zaustavljanja su obezbeđivali konvergenciju HGA metoda ka visoko kvalitetnim rešenjima problema. Samo mala poboljšanja finalnog rešenja HGA1 i HGA2 mogu se očekivati u slučaju produžavanja izvršavanja algoritma, što se može videti iz Tabela 2.1-2.4. Glavni cilj u ovom radu bio je nalaženje rešenja koja odgovaraju najboljim rešenjima USApHCP poznatim u literaturi do sada, tako da su vremena izvršavanja algoritma bila u drugom planu razmatranja.

Tabela 2.1 Rezultati HGA1 na CAB instancama

n	p	α	Opt.reš.	HGA_{najb}	t[s]	t_{tot}[s]	gen	agap [%]	σ [%]	eval	keš [%]
20	2	0.2	1892.9908	opt	0.007	0.112	206	0.000	0.000	3931	62.4
20	2	0.4	2160.7480	opt	0.003	0.117	201	0.000	0.000	3955	61.2
20	2	0.6	2274.6700	opt	0.005	0.104	208	0.000	0.000	3137	70.3
20	2	0.8	2501.9240	opt	0.003	0.141	201	0.000	0.000	4638	54.5
20	2	1.0	2609.1750	opt	0.001	0.103	201	0.000	0.000	2569	74.8
20	3	0.2	1551.2500	opt	0.045	0.247	243	0.000	0.000	8357	31.9
20	3	0.4	1760.1456	opt	0.023	0.221	219	0.000	0.000	7109	35.9
20	3	0.6	1997.7928	opt	0.032	0.240	225	0.000	0.000	7111	37.9
20	3	0.8	2263.5439	opt	0.060	0.269	248	0.000	0.000	7608	39.7
20	3	1.0	2600.0780	opt	0.060	0.320	246	0.010	0.026	9183	26.4
20	4	0.2	1355.4126	opt	0.088	0.352	275	0.025	0.113	11263	18.8
20	4	0.4	1472.7071	opt	0.076	0.336	258	0.575	1.886	9717	25.4
20	4	0.6	1834.8309	opt	0.084	0.356	264	0.027	0.123	10024	24.7
20	4	0.8	2152.9973	opt	0.121	0.435	280	0.006	0.028	11479	19.2
20	4	1.0	2600.0780	opt	0.095	0.398	266	0.002	0.002	11289	16.2
25	2	0.2	2131.1980	opt	0.001	0.143	201	0.000	0.000	3968	61.1
25	2	0.4	2402.5454	opt	0.004	0.153	201	0.000	0.000	4155	59.3
25	2	0.6	2558.7357	opt	0.002	0.138	201	0.000	0.000	3668	64.0
25	2	0.8	2714.9259	opt	0.001	0.134	201	0.000	0.000	3329	67.4
25	2	1.0	2827.1579	opt	0.007	0.134	205	0.000	0.000	2954	71.5
25	3	0.2	1923.1181	opt	0.082	0.349	263	0.212	0.332	8027	39.9
25	3	0.4	2100.4651	opt	0.004	0.272	201	0.000	0.000	6693	34.4
25	3	0.6	2340.2543	opt	0.005	0.285	201	0.000	0.000	6685	34.5
25	3	0.8	2554.1310	opt	0.031	0.334	221	0.000	0.000	7604	32.1
25	3	1.0	2758.3939	opt	0.060	0.356	240	0.737	0.555	7770	36.2
25	4	0.2	1619.4827	opt	0.079	0.401	249	0.105	0.323	8997	29.0
25	4	0.4	1884.8444	opt	0.074	0.386	241	0.000	0.000	8042	34.5
25	4	0.6	2182.4910	opt	0.105	0.462	252	0.743	0.522	9420	26.4
25	4	0.8	2454.3486	opt	0.077	0.463	241	1.074	0.463	9394	23.2
25	4	1.0	2726.2805	opt	0.241	0.612	323	0.214	0.324	12859	21.3

Tabela 2.2 Rezultati HGA2 na CAB instancama

n	p	α	Opt.reš.	HGA_{najb}	t[s]	t_{tot}[s]	gen	agap [%]	σ [%]	eval	keš [%]
20	2	0.2	1892.9908	opt	0.003	0.123	201	0.000	0.000	3880	62.0
20	2	0.4	2160.7480	opt	0.007	0.135	208	0.000	0.000	3947	62.6
20	2	0.6	2274.6700	opt	0.002	0.131	201	0.000	0.000	3666	64.1
20	2	0.8	2501.9240	opt	0.004	0.134	203	0.000	0.000	3794	63.1
20	2	1.0	2609.1750	opt	0.004	0.141	203	0.000	0.000	3578	65.3
20	3	0.2	1551.2500	opt	0.013	0.183	212	0.000	0.000	5119	52.5
20	3	0.4	1760.1456	opt	0.008	0.187	205	0.000	0.000	5154	50.5
20	3	0.6	1997.7928	opt	0.009	0.196	207	0.399	1.785	5241	50.1
20	3	0.8	2263.5439	opt	0.013	0.193	213	0.238	1.063	5051	53.2
20	3	1.0	2600.0780	opt	0.067	0.246	272	0.003	0.002	6273	54.1
20	4	0.2	1355.4126	opt	0.021	0.226	219	0.000	0.000	5778	47.8
20	4	0.4	1472.7071	opt	0.014	0.225	210	0.187	0.836	5180	51.3
20	4	0.6	1834.8309	opt	0.022	0.238	217	0.385	0.259	5792	47.4
20	4	0.8	2152.9973	opt	0.017	0.233	214	0.019	0.047	5544	48.8
20	4	1.0	2600.0780	opt	0.043	0.233	243	0.003	0.002	6249	49.0
25	2	0.2	2131.1980	opt	0.005	0.180	205	0.000	0.000	4244	59.1
25	2	0.4	2402.5455	opt	0.002	0.160	201	0.000	0.000	4021	60.6
25	2	0.6	2558.7357	opt	0.001	0.157	201	0.000	0.000	3713	63.6
25	2	0.8	2714.9260	opt	0.004	0.164	201	0.000	0.000	3725	63.5
25	2	1.0	2827.1579	opt	0.004	0.176	201	0.000	0.000	3639	64.3
25	3	0.2	1923.1181	opt	0.042	0.301	228	0.601	0.259	6309	45.4
25	3	0.4	2100.4651	opt	0.004	0.248	202	0.000	0.000	5067	50.7
25	3	0.6	2340.2543	opt	0.009	0.265	205	0.000	0.000	5407	48.0
25	3	0.8	2554.1310	opt	0.015	0.255	209	0.000	0.000	4824	54.6
25	3	1.0	2758.3939	opt	0.038	0.288	228	0.851	0.504	5877	48.9
25	4	0.2	1619.4827	opt	0.033	0.315	220	0.474	1.159	5847	47.5
25	4	0.4	1884.8444	opt	0.069	0.364	247	2.076	1.926	6272	50.0
25	4	0.6	2182.4910	opt	0.084	0.401	250	1.841	1.543	6865	45.8
25	4	0.8	2454.3486	opt	0.033	0.312	219	1.201	0.283	5283	52.4
25	4	1.0	2726.2805	opt	0.078	0.356	254	0.145	0.063	6716	47.4

U Tabelama 2.1 i 2.2 prikazani su rezultati HGA pristupa na CAB instancama, dok Tabela 2.3 i Tabela 2.4 sadrže rezultate istih metoda dobijene na AP instancama problema. Da bi rezultati mogli biti korektno upoređeni, svaka HGA implementacija je izvršena 20 puta na svim instancama.

U prvoj koloni Tabela 2.1-2.4 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 HGA dato je u sledećoj koloni, sa oznakom *opt* u slučajevima kada HGA dostiže unapred poznato optimalno rešenje. Prosečno vreme potrebno algoritmu da prvi put dobije najbolju vrednost je dato u *t[s]* koloni, dok t_{tot}*[s]* predstavlja ukupno vreme izvršavanja HGA za svih 500 generacija. U proseku, najbolje rešenje algoritam je našao posle *gen* generacija.

Kvalitet HGA rešenja u svih 20 izvršavanja se računa kao srednje procentualno odstupanje $agap = \frac{1}{20}\sum_{i=1}^{20} gap_i$, gde je gap_i procentualno odstupanje dobijenog rešenja u *i*-tom izvršavanju algoritma (*i* = 1, 2, . . . , 20). Pri tome se $gap_i = 100 * \frac{sol_i - Opt.sol}{Opt.sol}$ računa kao procentualno odstupanje u odnosu na unapred poznato optimalno rešenje *Opt.reš,* ili u odnosu na $HGA_{najb.}$

(najbolja vrednost HGA1, odnosno HGA2) $gap_i = 100 * \frac{sol_i - Best.sol}{Best.sol}$, ukoliko optimalno rešenje za datu instancu nije poznato (sol_i označava HGA rešenje dobijeno u *i*-tom izvršavanju algoritma). Standardna devijacija odstupanja $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(gap_i - agap)^2}$ predstavljena je u koloni *σ[%]* tabela 2.1-2.4.

Poslednje dve kolone u tabelama 2.1-2.4. 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 izračunavanja funkcije cilja, iskorišćeno je i do 76.3% vrednosti iz heš-tabele. Pošto se heš-tabela pretražuje i ažurira u konstantnom O(1) vremenu, procentualna ušteda vremena *kеš[%]* je veoma slična praktičnoj uštedi u vremenu izvršavanja.

Tabela 2.3 Rezultati HGA1 na AP instancama

n	p	Opt.reš.	HGA_{najb}	t[s]	t_{tot}[s]	gen	agap [%]	σ [%]	eval	keš [%]
10	2	40382.654	opt	0.002	0.067	201	0.000	0.000	4735	53.6
10	3	34772.375	opt	0.001	0.133	201	0.000	0.000	7657	24.9
10	4	32574.203	opt	0.002	0.138	201	0.000	0.000	8585	15.8
10	5	32531.213	opt	0.004	0.171	203	0.000	0.000	9424	8.3
20	2	45954.151	opt	0.044	0.210	252	0.571	1.395	6742	47.1
20	3	43400.446	opt	0.002	0.264	201	0.000	0.000	7052	30.9
20	4	38607.295	opt	0.033	0.287	222	2.076	3.253	8452	24.9
20	5	37868.148	opt	0.030	0.375	216	0.000	0.000	9145	16.4
20	10	37868.148	opt	0.009	0.545	201	0.000	0.000	9900	2.9
25	2	53207.459	opt	0.021	0.245	219	1.738	0.892	5848	47.2
25	3	46608.314	opt	0.036	0.369	225	0.000	0.000	7651	32.9
25	4	45552.497	opt	0.014	0.374	206	0.000	0.000	7906	24.5
25	5	45552.497	opt	0.005	0.403	201	0.000	0.000	8375	17.9
25	10	45552.497	opt	0.010	0.642	201	0.000	0.000	9745	4.5
40	2	61682.501	opt	0.069	0.698	228	0.244	0.595	5888	49.0
40	3	58192.761	opt	0.079	0.788	221	0.117	0.525	7543	32.8
40	4	52265.274	opt	0.106	1.128	227	2.028	0.477	8221	28.5
40	5	49741.201	opt	0.218	1.271	245	0.196	0.479	9863	20.5
40	10	49741.201	opt	0.034	1.179	204	0.000	0.000	9581	7.6
50	2	65523.366	opt	0.175	1.031	244	1.365	1.027	6516	47.0
50	3	60132.137	opt	0.070	1.330	211	0.501	0.171	7192	32.9
50	4	52905.770	opt	0.327	1.673	250	3.086	2.486	8965	29.0
50	5	50707.866	opt	0.651	2.407	279	2.412	2.823	10817	23.3
50	10	50707.866	opt	0.045	1.847	204	0.000	0.000	9451	8.8
100	2	65914.820	opt	0.369	4.193	223	0.393	0.329	5956	47.5
100	3	60658.884	opt	1.695	8.691	253	3.155	1.429	8148	36.5
100	4	-	56124.741	1.209	6.667	255	2.024	2.297	8855	31.6
100	5	54243.497	opt	1.783	8.198	264	0.801	0.806	9839	26.3
100	10	51860.026	opt	1.504	9.837	246	0.154	0.475	10739	13.8
100	15	-	51860.026	0.382	7.296	211	0.000	0.000	9779	8.8
100	20	-	51860.026	0.266	7.561	206	0.000	0.000	9875	5.7
200	2	-	68231.970	1.558	21.015	223	4.502	1.547	5734	49.0
200	3	-	64237.355	9.564	36.367	281	1.487	0.816	8951	37.3
200	4	-	60958.515	7.352	39.473	254	3.223	1.561	8423	34.5
200	5	-	59081.314	18.046	53.060	296	3.944	2.165	10442	29.9
200	10	-	55958.751	14.570	48.671	296	2.934	2.036	12177	18.5
200	15	-	55958.751	5.018	35.315	240	0.566	0.792	10672	12.1
200	20	-	55958.751	2.476	32.079	218	0.081	0.362	10063	8.8

Koncept HGA ne može dokazati optimalnost dobijenog rešenja, a takođe ne postoji adekvatan krriterijum zaustavljanja algoritma koji bi obezbedio fino podešavanje kvaliteta rešenja. Iz tog razloga, obe HGA metode se izvršavaju dodatno t_{tot} - t time, dok se ne zadovolji jedan od dva gore navedena kriterijuma zaustavljanja, iako je algoritam već dostigao optimalno rešenje (videti kolonu t_{tot} u tabelama 2.1-2.4). I pored toga, vreme izvršavanja HGA1 i HGA2 je relativno kratko, s obzirom da oba algoritma dostižu sva optimalna ili najbolja do sada poznata rešenja.

Rezultati prikazani u Tabelama 2.1 i 2.2 pokazuju da oba predložena hibridna algoritma HGA1 i HGA2 brzo dostižu sva do sada poznata optimalna rešenja na CAB instancama problema. Iz Tabela 2.3 i 2.4 može se videti da i na AP instancama HGA1 i HGA2 takođe dostižu sva poznata optimalna rešenja iz literature. Oba algoritma daju i rešenja na AP instancama velikih dimenzija, koje do sada nisu rešene: n=100, p=4,15,20 i n=200, p=2,3,4,5,10,15,20

Tabela 2.4 Rezultati HGA2 na AP instancama

n	p	Opt.reš.	HGA_{najb}	t[s]	t_{tot} [s]	gen	agap [%]	σ [%]	eval	keš [%]
10	2	40382.654	opt	0.000	0.056	201	0.000	0.000	2416	76.3
10	3	34772.375	opt	0.002	0.087	201	0.000	0.000	4462	56.3
10	4	32574.203	opt	0.002	0.095	201	0.000	0.000	4082	60.0
10	5	32531.213	opt	0.004	0.115	201	0.000	0.000	4701	53.9
20	2	45954.151	opt	0.005	0.136	204	0.000	0.000	3849	62.8
20	3	43400.446	opt	0.004	0.195	201	0.000	0.000	4396	56.9
20	4	38607.295	opt	0.064	0.268	250	0.000	0.000	6465	48.9
20	5	37868.148	opt	0.026	0.289	215	0.000	0.000	6114	43.9
20	10	37868.148	opt	0.006	0.442	201	0.000	0.000	7625	25.2
25	2	53207.459	opt	0.062	0.263	262	0.000	0.000	5683	57.0
25	3	46608.314	opt	0.024	0.294	215	0.000	0.000	5473	49.7
25	4	45552.497	opt	0.028	0.327	216	0.000	0.000	6125	44.0
25	5	45552.497	opt	0.020	0.350	208	0.000	0.000	6500	38.5
25	10	45552.497	opt	0.007	0.564	201	0.000	0.000	7938	22.2
40	2	61682.501	opt	0.061	0.503	226	0.304	1.360	5315	53.6
40	3	58192.761	opt	0.029	0.617	207	0.000	0.000	5616	46.6
40	4	52265.274	opt	0.447	1.187	325	0.960	1.089	9197	43.7
40	5	49741.201	opt	0.319	1.044	284	3.681	3.225	9180	35.7
40	10	49741.201	opt	0.015	1.051	201	0.000	0.000	7977	21.8
50	2	65523.366	opt	0.241	0.867	282	0.242	0.469	6713	52.7
50	3	60132.137	opt	0.230	1.220	244	0.594	0.278	7152	42.1
50	4	52905.770	opt	0.410	1.616	275	0.000	0.000	8231	40.7
50	5	50707.866	opt	0.548	2.043	290	0.000	0.000	9077	38.0
50	10	50707.866	opt	0.071	1.594	207	0.000	0.000	8053	23.4
100	2	65914.820	opt	0.784	3.723	253	0.821	1.025	6916	45.8
100	3	60658.884	opt	2.061	7.231	283	2.076	1.175	8905	37.6
100	4	-	56124.741	3.205	7.812	344	0.473	1.004	11381	34.4
100	5	54243.497	opt	3.808	9.356	329	0.698	0.882	11287	31.8
100	10	51860.026	opt	2.002	8.331	268	0.000	0.000	10366	23.4
100	15	-	51860.026	0.690	7.550	221	0.000	0.000	9289	17.2
100	20	-	51860.026	0.087	7.126	201	0.000	0.000	8752	14.2
200	2	-	68231.970	5.163	19.584	273	2.423	2.486	7833	43.1
200	3	-	64237.355	11.571	39.046	318	1.637	2.141	10695	33.4
200	4	-	60123.783	28.119	48.984	420	2.659	1.507	14708	30.4
200	5	-	58918.616	26.369	49.457	391	1.231	1.501	14067	28.5
200	10	-	55958.751	25.648	58.986	353	0.090	0.361	14093	20.6
200	15	-	55958.751	7.515	40.649	252	0.000	0.000	11061	13.3
200	20	-	55958.751	4.389	38.683	232	0.000	0.000	10314	12.1

U Tabelama 2.5 i 2.6 prikazano je poređenje HGA1 i HGA2 sa sledećim metodama:

- Egzaktnim pristupom iz [En04b], zasnovanim na linearizaciji modela USApHCP, koji je testiran sa CPLEX 7.0 na DEC Alpha procesoru (oznaka ERN u tabeli),
- Linearizacijom modela USApHCP, koja je predložena u [Kara98] i testirana sa CPLEX 5.0 na 8 CPU, 50Mhz Super Sparc Station sa 16MB memorije (oznaka KT u tabeli),
- Genetskim algoritmom GA za rešavanje USApHCP, opisanim u radu [Sol05] i testiranom na Intel Pentium III 733 MHz PC sa 256 MB RAM-a (oznaka GA u tabeli).

S obzirom na različitu prirodu metoda (neke su egzaktne, a neke heuristike), kao i na različite performanse računara na kojima su testirane, vremena izvršavanja ovih metoda se ne mogu direktno porediti. I pored toga, može se steći predstava o kvalitetu i performansama ovih algoritama.

Pošto su rezultati devet hibridnih heuristika na CAB instancama $n \leq 25$, $p \leq 5$ znatno lošiji u poređenju sa rezultatima GA iz [Sol05] (videti poređenja u istom radu), kao i (optimalnim) rezultatima linearizacija iz [Kara98] i [Ern04b], poređenja sa ovim heuristikama nisu prikazana u Tabeli 2.5. Kako je genetski algoritam iz [Sol05] testiran samo na CAB instancama problema, u Tabeli 2.6 nalaze se poređenja HGA1 i HGA2 sa linearizacijama ERN i KT na AP instancama.

Tabela 2.5 Poređenja na CAB instancama

Inst. n_p_α	Opt.reš.	HGA1		HGA2		ERN	KT	GA
		Najb.reš.	CPU AMD 1597MHz	Najb.reš.	CPU AMD 1597MHz	CPU DEC Alpha	CPU Sparc 50Mhz	agap[%]
20_2_0.2	1892.9908	opt	0.112	opt	0.123	2.08	74.23	0.04
20_2_0.4	2160.7480	opt	0.117	opt	0.135	7.18	173.99	0.00
20_2_0.6	2274.6700	opt	0.104	opt	0.131	3.34	104.42	0.00
20_2_0.8	2501.9240	opt	0.141	opt	0.134	3.74	125.67	0.00
20_2_1.0	2609.1750	opt	0.103	opt	0.141	1.81	96.22	0.00
20_3_0.2	1551.2500	opt	0.247	opt	0.183	1.48	102.19	0.00
20_3_0.4	1760.1456	opt	0.221	opt	0.187	5.03	344.25	0.00
20_3_0.6	1997.7928	opt	0.240	opt	0.196	2.66	270.15	0.00
20_3_0.8	2263.5439	opt	0.269	opt	0.193	3.44	126.12	0.30
20_3_1.0	2600.0780	opt	0.320	opt	0.246	2.56	155.81	0.00
20_4_0.2	1355.4126	opt	0.352	opt	0.226	0.91	136.94	0.00
20_4_0.4	1472.7071	opt	0.336	opt	0.225	7.73	185.69	0.75
20_4_0.6	1834.8309	opt	0.356	opt	0.238	3.89	105.02	0.49
20_4_0.8	2152.9973	opt	0.435	opt	0.233	3.31	108.71	0.20
20_4_1.0	2600.0780	opt	0.398	opt	0.233	1.83	111.21	0.00
25_2_0.2	2131.1980	opt	0.143	opt	0.180	4.94	819.51	0.00
25_2_0.4	2402.5455	opt	0.153	opt	0.160	13.14	1257.46	0.00
25_2_0.6	2558.7357	opt	0.138	opt	0.157	9.26	1287.06	0.00
25_2_0.8	2714.9260	opt	0.134	opt	0.164	5.34	1415.57	0.00
25_2_1.0	2827.1579	opt	0.134	opt	0.176	7.98	421.78	0.00
25_3_0.2	1923.1181	opt	0.349	opt	0.301	6.59	742.42	0.00
25_3_0.4	2100.4651	opt	0.272	opt	0.248	17.81	-	0.00
25_3_0.6	2340.2543	opt	0.285	opt	0.265	12.94	-	0.00
25_3_0.8	2554.1310	opt	0.334	opt	0.255	10.88	1881.75	0.00
25_3_1.0	2758.3939	opt	0.356	opt	0.288	7.56	1271.09	0.18
25_4_0.2	1619.4827	opt	0.401	opt	0.315	3.01	1314.26	0.74
25_4_0.4	1884.8444	opt	0.386	opt	0.364	16.68	-	0.95
25_4_0.6	2182.4910	opt	0.462	opt	0.401	9.83	-	0.29
25_4_0.8	2454.3486	opt	0.463	opt	0.312	18.49	-	0.76
25_4_1.0	2726.2805	opt	0.612	opt	0.356	4.18	542.01	0.15

Tabela 2.6 Poređenja na AP instancama

Inst. n_p	Opt.reš.	HGA1			HGA2			ERN	KT
		Najb. reš.	agap[%]	CPU AMD 1597MHz	Najb. reš.	agap[%]	CPU AMD 1597MHz	CPU DEC Alpha	CPU Sparc 50Mhz
10_2	40382.654	opt	0.000	0.067	opt	0.000	0.056	0.21	1.54
10_3	34772.375	opt	0.000	0.133	opt	0.000	0.087	0.28	1.41
10_4	32574.203	opt	0.000	0.138	opt	0.000	0.095	0.29	1.63
10_5	32531.213	opt	0.000	0.171	opt	0.000	0.115	0.16	1.54
20_2	45954.151	opt	0.571	0.210	opt	0.000	0.136	0.83	14.89
20_3	43400.446	opt	0.000	0.264	opt	0.000	0.195	3.54	83.51
20_4	38607.295	opt	2.076	0.287	opt	0.000	0.268	3.51	102.22
20_5	37868.148	opt	0.000	0.375	opt	0.000	0.289	1.53	116.76
20_10	37868.148	opt	0.000	0.545	opt	0.000	0.442	1.81	137.52
25_2	53207.459	opt	1.738	0.245	opt	0.000	0.263	1.91	155.84
25_3	46608.314	opt	0.000	0.369	opt	0.000	0.294	7.21	216.89
25_4	45552.497	opt	0.000	0.374	opt	0.000	0.327	6.08	94.12
25_5	45552.497	opt	0.000	0.403	opt	0.000	0.350	5.54	718.02
25_10	45552.497	opt	0.000	0.642	opt	0.000	0.564	8.49	1422.42
40_2	61682.501	opt	0.244	0.698	opt	0.304	0.503	4.39	-
40_3	58192.761	opt	0.117	0.788	opt	0.000	0.617	80.33	-
40_4	52265.274	opt	2.028	1.128	opt	0.960	1.187	66.39	-
40_5	49741.201	opt	0.196	1.271	opt	3.681	1.044	63.68	-
40_10	49741.201	opt	0.000	1.179	opt	0.000	1.051	9.84	-
50_2	65523.366	opt	1.365	1.031	opt	0.242	0.867	72.11	-
50_3	60132.137	opt	0.501	1.330	opt	0.594	1.220	97.94	-
50_4	52905.770	opt	3.086	1.673	opt	0.000	1.616	453.43	-
50_5	50707.866	opt	2.412	2.407	opt	0.000	2.043	175.52	-
50_10	50707.866	opt	0.000	1.847	opt	0.000	1.594	131.76	-
100_2	65914.820	opt	0.393	4.193	opt	0.821	3.723	935.64	-
100_3	60658.884	opt	3.155	8.691	opt	2.076	7.231	342.88	-
100_4	-	56124.741	2.024	6.667	56124.741	0.473	7.812	1287.62	-
100_5	54243.497	opt	0.801	8.198	opt	0.698	9.356	3300	-
100_10	51860.026	opt	0.154	9.837	opt	0.000	8.331	-	-
100_15	-	51860.026	0.000	7.296	51860.026	0.000	7.550	-	-
100_20	-	51860.026	0.000	7.561	51860.026	0.000	7.126	-	-
200_2	-	68231.970	4.502	21.015	68231.970	2.423	19.584	-	-
200_3	-	64237.355	1.487	36.367	64237.355	1.637	39.046	-	-
200_4	-	60958.515	3.223	39.473	60123.783	2.659	48.984	-	-
200_5	-	59081.314	3.944	53.060	58918.616	1.231	49.457	-	-
200_10	-	55958.751	2.934	48.671	55958.751	0.090	58.986	-	-
200_15	-	55958.751	0.566	35.315	55958.751	0.000	40.649	-	-
200_20	-	55958.751	0.081	32.079	55958.751	0.000	38.683	-	-

U Tabelama 2.5 i 2.6 za svaku CAB/AP instancu dato je optimalno rešenje - **Opt.reš.** (ako je poznato). Za svaku od metoda HGA1, HGA2, ERN, KT i GA prikazano je vreme izvršavanja **CPU** (CPU= t_{tot} u slučaju HGA1 i HGA2). Za heuristike HGA1, HGA2 i GA dati su i: najbolje rešenje metode za datu CAB/AP instancu (**Najb.reš.**) i srednje procentualno odstupanje (a**gap**) rešenja od optimalnog/najboljeg rešenja za tekuću instancu. Napomenimo da za GA heuristiku iz [Sol05] nisu prezentovana vremena izvršavanja na CAB instancama, već je dato prosečno i najduže CPU vreme izvrsavanja za sve instance i ona inose 1.64 s i 14.245 s respektivno.

Testiranja na standardnim ORLIB hab instancama [Bea96] sa $n \leq 200$ čvorova i $p \leq 20$ habova pokazuju da obe predložene hibridne metode HGA1 i HGA2 dostižu sva poznata optimalna rešenja iz literature i takođe daju rešenja za 10 do sada nerešenih AP instanci realnih dimenzija.

Rezultati dati u Tabeli 2.6 pokazuju da HGA2 daje nešto bolje rezultata na AP instancama problema u poređenju sa HGA1. U slučaju instanci AP 200_4 i 200_5, HGA2 dostiže bolja rešenja, ali i u slučajevima kada obe implementacije daju ista najbolja ili optimalna rešenja, srednje odstupanje HGA2 je manje od srednjeg odstupanja HGA1 implementacije (videti kolone **agap**). Ovi rezultati se mogu objasniti na sledeći način: HGA1 koristi genetski kod sa dva segmenta, dok genetski kod jedinke u HGA2 implementaciji ima samo jedan segment (videti sekciju 2.3.1). Reprezentacija korišćena u HGA1 ima za posledicu da su informacije o tome da li je neki čvor hab ili ne i alokacija datog čvora nekom od uspostavljenih habova razdvojene u različitim segmentima genetskog koda. U slučaju HGA2 reprezentacije, svaki gen genetskog koda se odnosi na tačno jedan čvor u mreži i sadrži sve informacije o datom čvoru (prvi bit gena određuje da li je dati čvor hab ili ne, a ostatak gena određuje hab pridružen datom čvoru). Dakle, šema kodiranja primenjena u HGA2 koristi kraće gradivne blokove (building-blocks) u odnosu na kodiranje u HGA1 implementaciji. Hipoteza o gradivnim blokovima (building-block hypothesis) i teorema o šemi (schema theorem) favorizuju kraće gradivne blokove i binarno kodiranje (videti [Bä00a] i [Bä00b]), što može biti objašnjenje za dobijene rezultate na AP instancama.

3 PROBLEM P-HAB MEDIJANE OGRANIČENIH KAPACITETA SA JEDNOSTRUKIM ALOKACIJAMA

Problemi dizajniranja hab mreža koje srećemo u praksi retko omogućavaju transport neograničenih količina jedinica robe između korisnika i snabdevača preko mreže habova u neograničenom vremenu. Ograničenja kapaciteta mogu biti različite prirode i mogu se odnositi na čvorove i/ili na lukove u mreži, na količinu protoka koji dolazi i/ili prolazi kroz čvor, protok koji se odvija u mreži habova itd. Hab lokacijski problemi ograničenih kapaciteta imaju veliki praktični značaj, ali su zbog dodatnih ograničenja dosta teži za rešavanje od odnosu na hab lokacijske probleme neograničenih kapaciteta.

Ovo poglavlje se bavi rešavanjem problema p-hab medijane ograničenih kapaciteta sa jednostrukim alokacijama (Capacitated Single Allocation p-Hub Median Problem - CSApHMP). Model CSApHMP je nastao na osnovu problema koji se javlja u poštanskim i drugim sistemima isporuke. Posmatramo *n* poštanskih regiona koji odgovaraju različitim poštanskim brojevima i predstavljaju čvorove u mreži. Ovi poštanski regioni svakodnevno razmenjuju pošiljke, pri čemu sve pošiljke koje se transportuju od polaznog poštanskog regiona (čvor snabdevač) do odredišnog poštanskog regiona (čvor korisnik), moraju proći kroz jedan ili najviše dva centra konsolidacije pošte (habova). Habovi takođe služe kao centri kolekcije i distribucije poštanskih pošiljki. CSApHMP model ima šemu jednostruke alokacije, što znači da se sva pošta iz polaznog poštanskog regiona može sakupljati samo u jednom habu, odnosno distribuirati u odredišni poštanski region iz samo jendnog hab čvora. Zbog vremenskog ograničenja rada poštanskih centara, količina pošiljki koja se sakuplja i sortira u svakom hab centru je ograničena. Broj habova koje je potrebno uspostaviti da bi rad poštanskog sistema bio što efikasniji je unapred poznat (*p*). Cilj CSApHMP je minimizacija ukupnih transportnih troškova u poštanskoj mreži, i eventualno fiksnih troškova uspostavljanja hab centara.

3.1 Formulacija problema

Formulacija problema, preuzeta iz [Cam94b], koju koristimo u ovom poglavlju koristi sledeće oznake:

C_{ij} = rastojanje između čvorova *i* i *j* (u smislu metrike),

W_{ij} = količina protoka (broj jedinica količine robe) od čvora-snabdevača *i* do čvora-korisnika *j,*

Γ_k = kapacitet haba *k,*

O_i = količina protoka koji polazi iz čvora *i*, tj. $O_i = \sum_{j=1}^{n} W_{ij}$,

D_j = količina protoka koji dolazi u čvor *j,* tj. $D_j = \sum_{i=1}^{n} W_{ij}$.

Promenljive CSApHMP modela su:

Z_{ij} = 1 ako je čvor *I* pridružen habu *k*, 0 inače,

Y_{kl}^{i} = količina protoka koja polazi od čvora-snabdevača *I,* sakuplja se u habu *k* I distribuira preko haba *l.*

Koristeći gornju notaciju, CSApHMP se može matematički zapisati na sledeći način:

$$\min \sum_{i=1}^{n}\sum_{k=1}^{n} C_{ik} Z_{ik} (\chi O_i + \delta D_i) + \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \alpha C_{kl} Y_{kl}^{i} \qquad (3.1)$$

uz uslove:

$$\sum_{k=1}^{n} Z_{kk} = p \qquad (3.2)$$

$$\sum_{k=1}^{n} Z_{ik} = 1, \quad \text{za svako i=1,…,n} \qquad (3.3)$$

$$\sum_{j=1}^{n} W_{ij} Z_{jk} + \sum_{l=1}^{n} Y_{kl}^{i} = \sum_{l=1}^{n} Y_{lk}^{i} + O_i Z_{ik}, \quad \text{za svako i,k=1,…,n} \qquad (3.4)$$

$$Z_{ik} \le Z_{kk}, \quad \text{za svako i,k=1,…,n} \qquad (3.5)$$

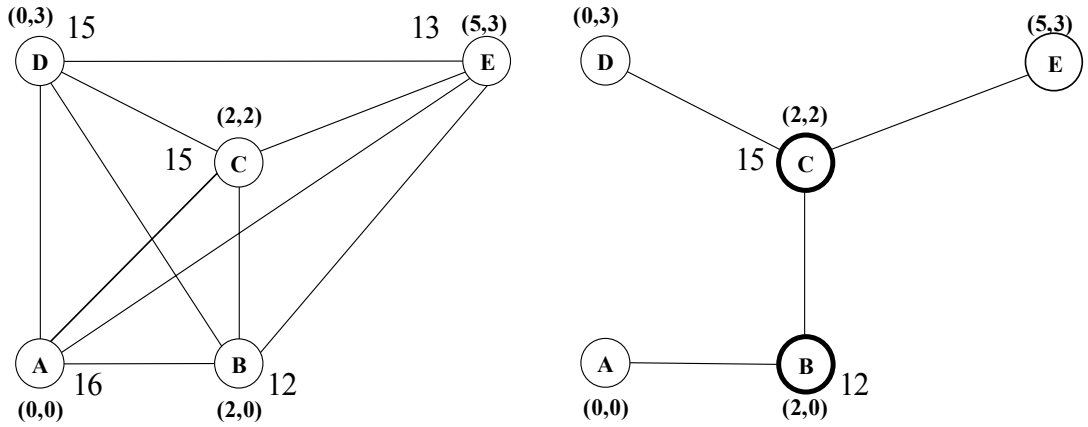
$$\sum_{i=1}^{n} O_i Z_{ik} \le \Gamma_k Z_{kk}, \quad \text{za svako k=1,…,n} \qquad (3.6)$$

$$Y_{kl}^{i} \ge 0, \quad \text{za svako i,k,l=1,…,n} \qquad (3.7)$$

$$Z_{ik} \in \{0,1\}, \quad \text{za svako i,k=1,….n} \qquad (3.8)$$

Cilj CSApHMP je minimizacija ukupnih troškova transporta u mreži (3.1). Uslov (3.2) određuje tačan broj habova koje treba uspostaviti *p*. Ograničenja (3.3) i (3.5) obezbeđuju da svaki korisnik/snabdevač bude pridružen tačno jednom, prethodno lociranom habu, sprečavajući direktnu komunikaciju između ne-hab čvorova. Jednačina konzervacije protoka u mreži data je uslovom (3.4), dok (3.6) ograničava količinu protoka koji se sakuplja u svakom habu. Konačno, (3.7) i (3.8) ukazuju na ne-negativnu, odnosno binarnu reprezentaciju promenljivih Y_{kl}^{i} i Z_{ik}. Koeficijenti *χ, α* i *δ* imaju isto značenje kao u prethodnom poglavlju. CSApHMP je NP-težak problem, jer je njegov potproblem sa neograničenim kapacitetima USApHMP NP-težak, što je dokazano u [Lo98].

Posmatrajmo ponovo mrežu sa pet čvorova A, B, C, D i E sa istim matricama rastojanja i protoka kao u prethodnom poglavlju. Neka su kapaciteti čvorova mreže 16, 12, 15, 15 i 13 respektivno. Potrebno je izabrati dva haba tako da ukupni troškovi transporta u mreži budu minimalni i da ukupna količina robe koja dolazi u svaki hab bude manja ili jednaka njegovom kapacitetu. U ovom slučaju, optimalno rešenje je sledeće: čvorovi **B** i **C** su izabrani za habove, ne-hab čvorovi D i E su pridruženi habu **C**, a ne-hab čvor A habu **B** (slika 3.1).



Slika 3.1 Hab mreža sa n=5, p=2 i $\chi=\delta=1$, $\alpha=0.25$

Troškovi transporta između svakog para čvorova u mreži su: "A-**B**-A" 2+2=4, "A-**B**-**C**" 2+2*0.25=2.5, "A-**B**-**C**-D" 2+2*0.25+$\sqrt{5}$=4.736, "A-**B**-**C**-E" 2+2*0.25+$\sqrt{10}$=5.662, "**B**-**B**" 0, "B-**C**" 2, "**B**-**C**" 2*0.25=0.5, "**B**-**C**-D" 2*0.25+$\sqrt{5}$=2.736 "**B**-**C**-E" 2*0.25+$\sqrt{10}$=3.662, "**C**-**C**" 0, "**C**-D" $\sqrt{5}$, "**C**-E" $\sqrt{10}$, "D-**C**-D" $\sqrt{5}+\sqrt{5}$=4.472, "D-**C**-E" $\sqrt{5}+\sqrt{10}$=5.398, "E-**C**-E" $\sqrt{10}+\sqrt{10}$=6.324 (hab čvorovi u stazama su boldovani). Vrednost funkcije cilja (ukupni transportni troškovi) je 79.983.

3.2 Postojeći načini rešavanja

Do sada u literaturi nalazimo samo jedan rad koji se bavi problemom p-hab medijane ograničenih kapaciteta sa jednostrukim alokacijama [Pér05]. U ovom radu autori predlažu evolutivnu Path Relinking (PR) metodu, GRASP heuristiku slučajne pretrage, kao i njihovu hibridizaciju za rešavanje CSApHMP, ali su prezentovani rezultati čudni i nekozistentni iz sledećih razloga.

U literaturi nalazimo nekoliko radova koji se bave rešavanjem potproblema CSApHMP sa neograničenim kapacitetima-USApHMP. Jedan od njih je [Pér04], iste grupe autora kao [Pér05], u kome je takođe primenjena PR heuristika za rešavanje USApHMP. Predložena metoda je testirana na istim AP instancama sa n≤200 čvorova i p≤20 habova, kao i u [Pér05]. Pošto CSApHMP uključuje ograničenja kapaciteta habova za količinu protoka koja se sakuplja u habu i distribuira iz haba, prirodno se očekuje da su za istu rešavanu AP instancu ukupni transportni troškovi za USApHMP manji ili jednaki ukupnim transportnim troškovima za CSApHMP. Ali, poredeći rezultate prezentovane u [Pér04] i [Pér05], može se primetiti da su u slučaju AP instance n=10, p=2, ukupni transportni troškovi dobijeni za CSApHMP iznose 35476.08, a za USApHMP 167493.06. Ista situacija je kod AP instanci n=10, p=3: 28635.25 za CSApHMP i 136008.13 za USApHMP, n=10, p=4: 30793.10 za CSApHMP i 112396.07 za USApHMP, n= 10, p=5: 30080.613 za CSApHMP i 91105.37 za USApHMP, itd. Ova nekonzistentnost rezultata bi se mogla objasniti korišćenjem različitih vrednosti paramentara χ, α i δ u [Pér04] i [Pér05], ali bi u tom slučaju dobijeni rezultati za USApHMP i CSApHMP bili približno proporcionalni, što nije slučaj (videti, na primer rezultate na AP instancama n=10,p=3 i n=10, p=5).

Druga nelogičnost se odnosi na rezultate za CSApHMP koji su dobijeni na AP instancama sa istim brojem čvorova *n,* a različitim brojem habova *p*. Ako se broj habova poveća, prirodno očekujemo da su ukupni transportni troškovi manji ili jednaki ukupnim transportnim troškovima koji su dobijeni za manje vrednosti p. Ali, u [Pér05] nalazimo da je rezultat PR metode za AP instancu n=10, p=2 (28635.25) strogo manji od rezultata za AP instancu n=10, p=3 (30793.10). Istu nelogičnost možemo primetiti i na sledećim AP instancama: n=25, p=2 (37359.92) i n=25, p=3 (42811.63); n=40, p=2 (46734.69) i n=40, p=3 (59999.84); n=100, p=15 (75699.27) i n=100, p=20 (75699.27).

Imajući u vidu gore navedeno, kao i činjenicu da su sva pisma koje sam uputila autorima [Pér05] sa molbom da mi razjasne navedene primedbe ostala bez odgovora, u ovom radu su izostavljena poređenja rezultata predloženog GA sa PR metodom iz [Pér05].

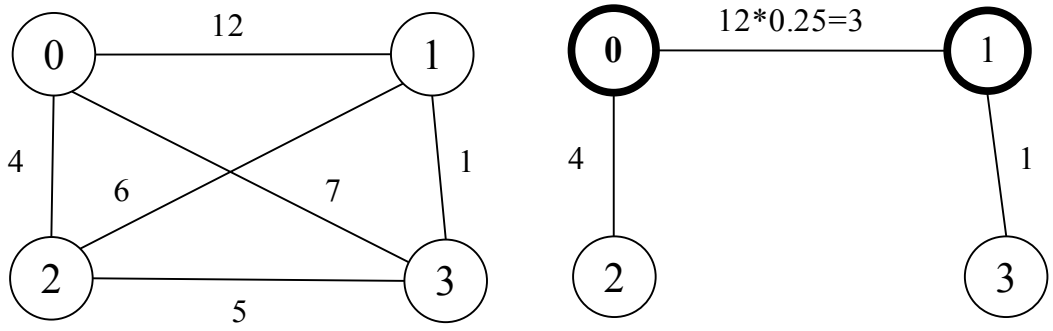
3.3 Predloženi genetski algoritmi

Genetski algoritmi za rešavanje CSApHMP imaju neke slične aspekte kao genetski koncepti koji su opisani u prethodnom poglavlju. Zbog toga, da bi se izbeglo ponavljanje sličnih delova, u ovoj sekciji će biti detaljno opisani samo GA aspekti koji se bitno razlikuju u odnosu na ranije opisane. Slični GA aspekti će biti samo kratko naznačeni.

3.3.1 Reprezentacija jedinki i funkcija cilja

Predložene GA1 i GA2 implementacije koriste iste šeme kodiranja koje su primenjene u HGA1 i HGA2 metodama respektivno (videti sekciju 2.3.1). Obe GA metode koriste već opisani princip "uređenja najbližeg suseda". Za svaki ne-hab čvor formira se niz uspostavljenih habova, koji se zatim sortira koristeći strategiju "najbližeg suseda" kojom se obezbeđuje češće pridruživanje ne-hab čvorova "bližim" habovima, dok je svaki hab čvor pridružen samom sebi.

Na levoj strani slike 3.2. prikazana je mreža četiri čvora 0, 1, 2, 3 od kojih je potrebno izabrati dva haba. Odgovarajuća matrica rastojanja je C=[0 12 4 7;12 0 6 1; 4 6 0 9; 7 1 9 0]. Količina protoka od čvora-snabdevača *i* do čvora-korisnika *j* je W_{ij}*=1 (i, j=0,1,2,3).* Optimalno rešenje koje se dobija za vrednosti parametara χ=1,α=0.25 i δ=1 dato je na desnoj strani slike 3.2. Habovi su uspostavljeni u čvorovima 0 i 1, dok su ne-hab čvorovi 2 i 3 pridruženi habovima 0 i 1 redom.



Slika 3.2 Hab mreža sa n=4, p=2 i $\chi=\delta=1$, α=0.25

Primer 4. U GA2 implementaciji, genetski kod 00 |10 |10 | 00 odgovara optimalnom rešenju prikazanom na desnoj strani slike 3.2 za n=4, p=2. Prvi bit svakog gena (0, 1, 1, 0) označava uspostavljene habove (čvorovi sa indeksima 1 i 2), dok preostali delovi gena (0, 0, 0, 0) određuju alokacije čvorova. Za ne-hab čvor 0, niz uspostavljenih habova (sortiran po principu "najbližeg suseda") je 2,1, sa rastojanjima 4 i 12 redom. Za ne-hab čvor 3, pridruženi niz habova je 1,2 sa odgovarajućim rastojanjima 1 i 9. Oba ne-hab čvora 0 i 3 pridružena su svojim najbližim uspostavljenim habovima: čvor 0 pridružen je habu 2, a čvor 3 habu 1. Habovi 1 i 2 su pridruženi samom sebi.

Primer 5. Imajući u vidu reprezentaciju jedinki koju koristi GA1 implementacija, genetski kod 11| 00 odgovara istom optimalnom rešenju sa slike 3.2. Prvi segment genetskog koda sadrži indekse uspostavljenih habova: 1 i 2. Dupliranje indeksa je rešeno na isti način kao u HGA1 implementaciji, pa kako je indeks 1 već "zauzet", uzimamo prvi sledeći "slobodan" indeks iz skupa {0,1,2,3}, tj. 2. Drugi segment genetskog koda ima *n-p* gena sa po *p-1* bitova koji određuju alokacije ne-hab čvorova. Dve nule u drugom segmentu genetskog koda označavaju da su ne-hab čvorovi 0 i 3 pridruženi svojim najbližim uspostavljenim habovima, 2 i 3 respektivno.

Šema kodiranja primenjena u GA2 implementaciji obezbeđuje da se podaci o tome da li je neki čvor hab ili ne, kao i njegovoj alokaciji određenom habu, nalaze u istom genu. Na ovaj način su informacije o svakom čvoru grupisane u genetskom kodu, što obezbeđuje kraće "gradivne blokove" genetskog koda. Hipoteza o gradivnim blokovima i teorema o šemi ([Bä00a] i [Bä00a]) favorizuju kraće "gradivne blokove" i binarno kodiranje, tako da primenjeni način kodiranja značajno utiče na efikasnost predloženog GA. Eksperimentalni rezultati i poređenja u Sekciji 3.1.4 potvrđuju gorenavedene pretpostavke i jasno ukazuju na prednosti korišćenja GA2 kodiranja u odnosu na GA1 kodiranje za rešavanje CSApHMP.

Nakon dobijanja indeksa uspostavljenih habova i alokacija svakog ne-hab čvora odgovarajućem habu, računa se vrednost funkcije cilja. Imajući u vidu (4.1), vrednost funkcije cilja se dobija jednostavnim sumiranjem rastojanja snabdevač-hab, hab-hab, hab-korisnik, pomnoženim odgovarajućim parametrima χ, α i δ.

Može se desiti da je neki ne-hab čvor pridružen habu čiji (preostali) kapacitet nije dovoljan da bi zadovoljio potrebe datog čvora. U tom slučaju, iz sortiranog niza uspostavljenih habova za dati ne-hab čvor uzimamo prvi sledeći hab, koji ima dovoljno kapaciteta da opsluži dati ne-hab čvor. Ako takav hab ne postoji, datu jedinku smatramo nekorektnom i uklanjamo je iz populacije postavljajući njenu funkciju prilagođenosti na nulu. Ovakvi slučajevi se obično dešavaju kada je suma kapaciteta habova manja od ukupne količine protoka u mreži, a to se u praksi retko dešava. Verovatnoća generisanja nekorektnih jedinki (u smislu nedovoljnih kapaciteta habova) u početnoj populaciji je veoma mala.

Opisana strategija popravljanja nekorektnih jedinki može neznatno uticati na kvalitet GA rešenja, ali sa druge strane, ona čuva raznovrsnost genetskog materijala. Ako bi sve nekorektne jedinke bile uključene u početnu populaciju, one mogu postani dominantne u narednim generacijama GA. U tom slučaju, može se desiti da algoritam ne daje nijedno rešenje ili da završi u lokalnom optimumu. Sa druge strane, ako sve nekorektne jedinke isključujemo iz

populacije u svim generacijama genetskog algoritma, mogućnost preuranjene konvergencije algoritma se značajno povećava.

Navedeni razlozi predstavljaju objašnjenje zašto heuristike lokalne pretrage (local-search heuristics) uglavnom daju loše rezultate za hab lokacijske probleme ograničenih kapaciteta, čak i u slučaju malih dimenzija problema. Ograničenja kapaciteta dovode do velikog broja nekorektnih rešenja, što znatno otežava lokalno pretraživanje kroz prostor rešenja, pa ova metoda često daje suboptimalno ili nijedno rešenje problema. Predložena strategija popravljanja nekorektnih jedinki (rešenja) do korektnih u početnoj populaciji genetskog algoritma, kao i implementacija genetskih operatora koji čuvaju koreknost jedinki, dovode do toga da se nekorektne jedinke ne pojavljuju u narednim generacijama GA.

3.3.2 Genetski operatori

Predložene GA1 i GA2 metode koriste fino gradiranu turnirsku selekciju FGTS, opisanu u sekciji 2.3.4.1. Realni parametar F_{tour} (željena srednja veličina turnira) takođe uzima vrednost 5.4 u obe GA implementacije.

Zbog različite prirode segmenata genetskog koda jedinki-roditelja, u GA1 implementirano je dvostruko jednopoziciono ukrštanje (videti sekciju 2.3.4.2). Dobijene jedinke-potomci su uvek korektne, jer operator ukrštanja čuva broj uspostavljenih habova *p,* a primenjena šema kodiranja ne dopušta dupliranje indeksa habova.

Standardni operator ukrštanja nije pogodan za način kodiranja primenjen u GA2, jer može proizvesti nekorektne jedinke (broj uspostavljenih habova je različit od *p*), čak i u slučaju da oba roditelja imaju tačno *p* jedinica na prvim bitovima gena genetskog koda. Iz tog razloga, predloženi GA2 koristi modifikovani operator ukrštanja, opisan u sekciji 2.3.4.2. koji takođe čuva broj uspostavljenih habova *p*. Modifikovani operatori ukrštanja u GA1 i GA2 izvršavaju se sa verovatnoćom p_{cross}=0.85.

U oba predložena GA, implementiran je modifikovani operator mutacije sa zaleđenim bitovima. S obzirom na primenjene načine kodiranja i "uređenje najbližeg suseda", osnovni nivo mutacije, kao i nivo mutacije zaleđenih bitova u svakom genu genetskog koda razlikuju se u odnosu na poziciju datog bita (videti sekciju 2.3.4.3 u prethodnom poglavlju).

Nivoi mutacije u GA1 implementaciji su:

- Osnovni nivo mutacije je 0.7/*n* za sve gene prvog segmenta genetskog koda. Zaleđeni bitovi na istoj poziciji mutiraju se sa 2.5 većom verovatnoćom, tj. 1.75/*n*
- U svakom genu drugog segmenta, osnovni nivo mutacije vodećeg bita je 0.1/n, dok se svaki naredni bit mutira se sa dva puta manjom verovatnoćom u odnosu na svog prethodnika (0.05/*n*, 0.025/*n*, ...). Nivo mutacije svakog zaleđenog bita u genu je 1.5 puta veći u odnosu na odgovarajući osnovni nivo (0.15/n, 0.075/n, ...)

Nivoi mutacije u GA2 implementaciju su:

- Bit na prvoj poziciji u svakom genu genetskog koda ima osnovni nivo mutacije 0.4/*n,* a ukoliko je zaleđen, nivo mutacije je 2.5 puta veći, tj. 1.0/*n,*
- Osnovni nivo mutacije je 0.1/*n* za gen na drugoj poziciji, dok se naredni bitovi u genu mutiraju se sa dva puta manjom verovatnoćom u odnosu na

svog prethodnika (0.05/*n*, 0.025/*n,...*). Verovatnoće mutacije zaleđenih bitova na istim pozicijama su 1.5 puta veće od odgovarajućih osnovnih vrednosti (*0.15/n* umesto 0.1/*n*, *0.075/n* umesto 0.05/*n, 0.0375/n* umesto 0.025/*n, ...*).

Zbog primenjenog načina kodiranja, sve mutirane jedinke u GA1 implemetaciji su korektne. U slučaju GA2, operator mutacije dodatno menja vodeće bitove gena u genetskom kodu nekorektne mutirane jedinke, tako da broj jedinica na prvim pozicijama gena njenog genetskog koda bude tačno *p* (videti sekciju 2.3.4.3).

3.3.3 Ostale karakteristike GA

Početne populacije obe predložene GA metode imaju po 150 jedinki. Generisanje početne populacije u GA1 (GA2) vrši se na isti način kao u slučaju HGA1 (HGA2) implementacije. Verovatnoća generisanja svakog bita zavisi od njegove pozicije u datom genu genetskog koda (videti sekciju 2.3.6). Ostali aspekti GA1(GA2) su takođe isti kao aspekti HGA1(HGA2):

- politika zamene generacija (stacionarni GA sa elitističkom strategijom i 100 elitnih jedinki)
- popravljanje nekorektnih jedinki, sa brojem uspostavljenih habova koji je različit od *p,* do korektnih (u slučaju GA2)
- uklanjanje dupliranih jedinki iz populacije
- ograničavanje broja jedinki koje imaju istu vrednost funkcije cilja, a različite genetske kodove (N_{rv}=40)
- keširanje

Maksimalan broj generacija u oba GA je N_{gen}=5000 za sve instance problema. Algoritmi se takođe zaustavljaju ako je najbolja jedinka ili najbolja vrednost funkcije cilja ostala nepromenjena tokom N_{rep}=2000 uzastopnih generacija. Maksimailan broj keširanih vrednosti funkcije cilja u heš-red tabeli je ograničen na N_{cache}=5000 u obe GA implementacije.

3.4 Rezultati i poređenja

U ovoj sekciji prikazani su rezultati predloženih GA metoda za rešavanje CSApHMP i poređenja sa rezultatima dobijenim korišćenjem solvera CPLEX 8.1.0. GA implementacije su kodirane u programskom jeziku C i testirane na AMD Sempron procesoru na 1597 MHz sa 256 MB RAM memorije pod Linux (Knoppix 3.7) operativnom sistemu. Testiranja sa CPLEX-om su izvršena na Intel procesoru na 1.8 GHz sa 256 MB RAM memorije. Sva tri algoritma su testirana na standardnim AP hab instancama [Bea96] sa n≤200 čvorova i p≤20 habova. Korišćena su dva tipa ograničenja kapaciteta na habovima: jaka (tight -T) i slaba (loose-L). Na svakoj AP instanci problema predloženi GA1 i GA2 izvršeni su po 20 puta. U tabelama nisu navedene instance koje nisu korektne (ne postoji nijedno dopustivo rešenje) jer je zbir kapaciteta habova manji od ukupne količine protoka koja se sakuplja u habovima.

Optimalna rešenja dobijena CPLEX algoritmom na AP instancama manjih dimenzija n≤50 prikazana su u Tabeli 3.1. Prva kolona Tabele 3.1 sadrži ime AP instance, njene dimenzije (*n* i *p*), kao i tip ograničenja kapaciteta. Na primer, oznaka 40L_5 odnosi se na AP instancu sa n=40 čvorova i p=5 habova koje

treba uspostaviti, dok "L" označava slaba ograničenja kapaciteta na habovima. Naredne tri kolone sadrže: optimalno rešenje tekuće instance dobijeno CPLEX-om, zatim odgovarajuće vreme izvršavanja algoritma t[s] (u sekundama) i broj iteracija algoritma n_{iter}.

U Tabelama 3.2 i 3.3 prikazani su rezultati GA1 i GA2 pristupa na AP instancama manjih dimenzija n≤50. Najbolje GA1(GA2) rešenje dato je u trećoj koloni Tabele 3.1, sa oznakama *opt* u slučajevima kada GA u bar jednom od 20 izvršavanja dostiže unapred poznato optimalno rešenje dobijeno CPLEX algoritmom. Ako GA1(GA2) ne daje nikakvo rešenje za neku od instanci, u odgovarajuće polje upisano je "-". Naredne kolone Tabela 3.2 i 3.3 sadrže:

- prosečno vreme (u sekundama) potrebno algoritmu da dobije najbolju vrednost *t[s],*
- ukupno vreme izvršavanja GA (u sekundama) za svih 5000 generacija t_{tot}*[s]*,
- srednji broj generacija *gen* potrebnih algoritmu da dostigne najbolje rešenje
- srednje procentualno odstupanje a*gap[%]* u odnosu na unapred poznato optimalno rešenje ili najbolje rešenje GA,
- standardnu devijaciju odstupanja *σ [%],*
- prosečan broj neophodnih izračunavanja,
- ušteda vremena *keš[%]* ostvarenu primenom tehnike keširanja.

Kao što se može videti iz Tabela 3.2 i 3.3, za sve AP instance manjih dimenzija oba predložena GA brzo dostižu sva optimalna rešenja, koja su prethodno dobijena CPLEX algoritmom, sem u slučaju instance 20T_2 za koju ni GA1 ni GA2 nisu dali nijedno rešenje.

Tabela 3.1 Rezultati CPLEX-a na AP instancama manjih dimenzija

Inst.	CPLEX			Inst.	CPLEX		
	Opt.reš.	t[s] Intel 1.8 GHz	N_{iter}		Opt.reš.	t[s] Intel 1.8 GHz	N_{iter}
10L_2	167493.06497	0.01	120	25T_3	195500.88511	9.88	4792
10L_3	136008.12591	0.18	290	25T_4	168226.97385	20.66	12684
10L_4	122396.06809	0.21	322	25T_5	145318.64862	10.25	5881
10L_5	91105.37068	0.17	262	25T_10	99700.96103	4.78	2719
10T_2	198060.79558	0.1	299	40L_2	178602.58220	10.21	2134
10T_3	166219.04818	0.22	299	40L_3	161641.90679	73.36	7695
10T_4	137344.52962	0.19	368	40L_4	145931.30224	60.22	6402
10T_5	122903.55479	0.42	983	40L_5	135908.09368	110.11	13131
20L_2	172816.68972	0.55	558	40L_10	104631.51100	108.21	12933
20L_3	151533.08378	2.45	1481	40T_3	179795.59780	34.29	4434
20L_4	135624.88360	2.85	1797	40T_4	155041.20503	35.33	4379
20L_5	123130.09462	4.41	3043	40T_5	141412.38851	29.03	3197
20L_10	81870.54858	2.13	1359	40T_10	107089.58661	72.65	9848
20T_2	194077.66234	1.16	712	50L_2	178548.98256	41.42	3467
20T_3	165290.85005	4.51	3982	50L_3	161008.12675	200.26	9813
20T_4	139415.28208	1.7	1010	50L_4	145578.81661	334.5	23473
20T_5	123453.01027	1.42	800	50L_5	132651.43955	190.22	10346
20T_10	84661.06551	1.62	1093	50L_10	102706.85734	365.61	30055
25L_2	177869.30164	1.92	1079	50T_3	185779.91419	563.4	51631
25L_3	155256.32315	4.03	1380	50T_4	153334.47291	292.06	24079
25L_4	139197.16909	7.38	2813	50T_5	139682.21196	630.55	51642
25L_5	123574.28868	4.73	1908	50T_10	104470.15626	441.52	32852
25L_10	81870.54858	2.12	1359				

Testiranja na AP instancama velikih dimenzija n=100,200, p≤20 pokazala su da CPLEX algoritam ne daje rešenja za ove instance u doglednom vremenu izvršavanja. Na ovim instancama CPLEX algoritam se nije izvršio do kraja ni posle nekoliko dana neprekidnog izvršavanja, iako je u većini slučajeva našao

gornje granice problema. Zbog toga su u Tabelama 3.4 i 3.5 prikazani samo rezultati predloženih GA metoda. Rezultati GA1 i GA2 su prikazani na isti način kao u Tabelama 3.2 i 3.3. Kao što se može videti iz Tabela 3.4 i 3.5, obe predložene GA metode u relativno kratkom CPU vremenu daje rešenja za sve AP instance velikih dimenzija.

Tabela 3.2 Rezultati GA1 na AP instancama manjih dimenzija

Inst.	Opt.reš. CPLEX	GA1							
		Najb.reš. GA1	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
10L_2	167493.06497	opt	0.004	0.323	2013	0.000	0.000	6293.4	93.8
10L_3	136008.12591	opt	0.007	0.672	2013	0.000	0.000	63644.8	36.9
10L_4	122396.06809	opt	0.007	0.785	2019	0.000	0.000	73312.9	27.5
10L_5	91105.37068	opt	0.02	0.973	2039	0.000	0.000	88264.0	13.6
10T_2	198060.79558	opt	0.003	0.319	2008	0.000	0.000	7816.8	92.2
10T_3	166219.04818	opt	0.003	0.636	2006	0.000	0.000	59116.2	41.2
10T_4	137344.52962	opt	0.004	0.757	2013	0.000	0.000	72390.8	28.2
10T_5	122903.55479	opt	0.053	0.942	2116	0.000	0.000	89917.5	15.1
20L_2	172816.68972	opt	0.01	0.465	2019	0.000	0.000	10284	89.8
20L_3	151533.08378	opt	0.012	1.145	2016	0.000	0.000	52799.8	47.7
20L_4	135624.88360	opt	0.026	1.377	2030	0.000	0.000	60032.1	41.0
20L_5	123130.09462	opt	0.171	2.01	2182	0.009	0.040	79966.8	26.9
20L_10	81870.54858	opt	0.075	3.029	2048	0.000	0.000	96791.0	5.7
20T_2	194077.66234	opt	0.003	0.351	1713	2.148	5.474	9149.2	76.7
20T_3	165290.85005	opt	0.077	1.025	2151	0.603	0.256	58049.5	46.4
20T_4	139415.28208	opt	0.073	1.284	2112	0.000	0.000	62112.2	41.2
20T_5	123453.01027	opt	0.031	1.639	2033	0.000	0.000	75966.5	25.4
20T_10	84661.06551	opt	0.147	2.102	2135	0.012	0.053	100471.8	6.0
25L_2	177869.30164	opt	0.175	0.722	2558	0.702	1.714	16566	87.0
25L_3	155256.32315	opt	0.132	1.423	2182	1.017	1.432	48573.8	55.6
25L_4	139197.16909	opt	0.064	1.571	2068	0.362	0.214	50415.6	51.3
25L_5	123574.28868	opt	0.249	2.379	2228	0.010	0.025	70221	36.9
25L_10	81870.54858	opt	0.09	4	2044	0.198	0.484	93295.1	8.9
25T_3	195500.88511	opt	0.059	1.116	2092	0.492	0.120	49237.1	53.1
25T_4	168226.97385	opt	0.137	1.521	2196	0.256	0.078	54449.2	50.3
25T_5	145318.64862	opt	0.136	1.912	2132	0.909	1.454	67376.4	36.9
25T_10	99700.96103	opt	0.35	3.286	2240	0.032	0.144	101261.9	9.7
40L_2	178602.58220	opt	0.069	0.831	2116	0.356	0.870	12767.8	88.0
40L_3	161641.90679	opt	0.142	2.192	2113	0.415	0.852	43729.9	58.6
40L_4	145931.30224	opt	0.284	2.649	2223	0.001	0.001	48896.8	56.1
40L_5	135908.09368	opt	0.239	3.672	2122	0.222	0.076	63384.7	40.4
40L_10	104631.51100	opt	4.003	10.543	3202	0.560	0.556	136900	14.6
40T_3	179795.59780	-	0	0.002	1	0.000	0.000	150.0	0
40T_4	155041.20503	opt	0.58	2.692	2517	0.646	1.721	62542.8	50.4
40T_5	141412.38851	opt	0.454	3.644	2274	0.496	1.090	72224.4	36.4
40T_10	107089.58661	opt	2.768	8.369	2972	0.278	0.377	128713.6	13.4
50L_2	178548.98256	opt	0.093	1.033	2120	0.000	0.000	12162.8	88.6
50L_3	161008.12675	opt	0.216	2.778	2139	0.397	0.629	38637.7	64.0
50L_4	145578.81661	opt	0.211	3.253	2105	0.478	0.707	42664.8	59.5
50L_5	132651.43955	opt	0.218	4.557	2078	0.299	0.800	55052.6	47.1
50L_10	102706.85734	opt	2.312	10.961	2524	0.776	0.644	99828.4	21.0
50T_3	185779.91419	-	0	0.003	1	0.000	0.000	150.0	0
50T_4	153334.47291	opt	0.185	2.962	2099	2.971	1.916	45353.2	56.8
50T_5	139682.21196	opt	0.506	4.708	2216	0.618	0.879	59852.4	45.9
50T_10	104470.15626	opt	1.581	9.368	2396	1.076	0.649	95205.9	20.7

Tabela 3.3 Rezultati GA2 na AP instancama manjih dimenzija

Inst.	Opt.reš. CPLEX	GA2							
		Najb.reš. GA2	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
10L_2	167493.06497	opt	0.005	0.368	2013	0.000	0.000	3564.8	96.5
10L_3	136008.12591	opt	0.004	0.5	2010	0.000	0.000	17984.5	82.1
10L_4	122396.06809	opt	0.01	0.519	2022	0.000	0.000	18871.5	81.4
10L_5	91105.37068	opt	0.015	0.637	2037	0.000	0.000	24984.8	75.5
10T_2	198060.79558	opt	0.004	0.364	2018	0.000	0.000	4102.9	95.9
10T_3	166219.04818	opt	0.004	0.491	2015	0.000	0.000	18148.3	82.0
10T_4	137344.52962	opt	0.009	0.508	2015	0.000	0.000	19029.5	81.1
10T_5	122903.55479	opt	0.052	0.673	2151	0.001	0.003	30402.3	71.8
20L_2	172816.68972	opt	0.025	0.629	2053	0.000	0.000	11259.1	89.0
20L_3	151533.08378	opt	0.028	0.986	2041	0.000	0.000	28273.5	72.3
20L_4	135624.88360	opt	0.024	1.077	2029	0.000	0.000	30892.3	69.6
20L_5	123130.09462	opt	0.136	1.488	2180	0.000	0.000	41536.8	62.2
20L_10	81870.54858	opt	0.05	1.926	2038	0.013	0.056	46326.6	54.6
20T_2	194077.66234	-	0	0.001	1	0.000	0.000	150.0	0.0
20T_3	165290.85005	opt	0.09	0.978	2175	0.497	0.259	32762.0	69.9
20T_4	139415.28208	opt	0.037	1.031	2054	0.000	0.000	31449.9	69.4
20T_5	123453.01027	opt	0.029	1.262	2031	0.000	0.000	36050.0	64.6
20T_10	84661.06551	opt	0.044	1.698	2040	0.000	0.000	45794.0	55.2
25L_2	177869.30164	opt	0.065	0.837	2114	0.000	0.000	17481.0	83.5
25L_3	155256.32315	opt	0.063	1.322	2078	0.000	0.000	33468.4	67.9
25L_4	139197.16909	opt	0.107	1.562	2126	0.137	0.217	40405.8	62.0
25L_5	123574.28868	opt	0.153	1.983	2158	0.007	0.021	47502.7	56.0
25L_10	81870.54858	opt	0.114	2.636	2068	0.000	0.000	47681.3	54.0
25T_3	195500.88511	opt	0.065	1.144	2096	0.518	0.460	36549.2	65.2
25T_4	168226.97385	opt	0.122	1.39	2159	0.223	0.066	39530.6	63.4
25T_5	145318.64862	opt	0.341	1.989	2392	0.386	0.942	54966.8	54.2
25T_10	99700.96103	opt	0.088	2.244	2063	0.000	0.000	44977.8	56.5
40L_2	178602.58220	opt	0.148	1.495	2169	0.000	0.000	27426.0	74.8
40L_3	161641.90679	opt	0.147	2.475	2102	0.000	0.000	43089.8	59.1
40L_4	145931.30224	opt	0.476	3.222	2322	0.000	0.000	53733.7	53.7
40L_5	135908.09368	opt	0.472	3.742	2264	0.012	0.056	55454.3	51.1
40L_10	104631.51100	opt	0.642	5.397	2238	0.045	0.137	58528.5	47.8
40T_3	179795.59780	opt	0.52	1.954	2277	5.376	11.836	47284.8	50.1
40T_4	155041.20503	opt	0.473	2.846	2363	0.000	0.000	53030.1	55.2
40T_5	141412.38851	opt	0.169	3.29	2093	0.106	0.376	50690.9	51.6
40T_10	107089.58661	opt	0.953	5.45	2389	0.066	0.296	62842.3	47.5
50L_2	178548.98256	opt	0.401	2.281	2345	0.000	0.000	35509.4	69.8
50L_3	161008.12675	opt	0.322	3.638	2163	0.000	0.000	48118.4	55.6
50L_4	145578.81661	opt	0.409	4.217	2184	0.000	0.000	52359.7	52.1
50L_5	132651.43955	opt	0.522	5.114	2201	0.054	0.242	57019.3	48.3
50L_10	102706.85734	opt	1.559	8.339	2424	0.284	0.356	67992.6	44.0
50T_3	185779.91419	opt	0.889	3.106	2764	1.130	1.413	61518.3	55.5
50T_4	153334.47291	opt	1.725	5.13	2956	0.654	1.008	70842.1	52.1
50T_5	139682.21196	opt	0.775	5.165	2323	0.080	0.267	60252.3	48.2
50T_10	104470.15626	opt	1.772	8.165	2516	0.354	0.412	69817.2	44.6

Tabela 3.4 Rezultati GA1 na AP instancama velikih dimenzija

Inst.	Najb.reš. GA1	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
100L_2	189351.55220	0.384	2.856	2126	0.347	1.199	14192.4	86.7
100L_3	165714.45524	1.057	7.961	2240	0.720	0.883	38666.7	65.7
100L_4	146197.75674	1.852	9.481	2417	0.206	0.559	42393.8	65.1
100L_5	137398.84844	3.920	14.384	2686	1.033	1.445	61273.1	54.4
100L_10	107381.77382	5.835	24.946	2540	1.419	1.003	84794.1	33.2
100L_15	91285.26316	13.926	42.328	2952	1.375	0.864	119215.6	19.3
100L_20	81230.96016	30.163	66.603	3561	1.315	0.775	159109.2	10.7
100T_4	157073.05820	2.640	9.459	2651	2.973	1.751	50284.2	61.8
100T_5	143567.78336	3.410	14.184	2575	2.180	1.732	62705.4	51.3
100T_10	109633.40777	9.521	29.443	2902	1.665	0.825	99184.7	31.8
100T_15	92947.09085	16.151	44.241	3061	1.303	1.113	124071.0	19.1
100T_20	82643.69838	27.020	64.463	3439	1.089	0.774	154505.6	10.2
200L_2	182459.25446	6.146	17.105	2669	5.063	4.992	21121.3	84.4
200L_3	162887.03135	4.047	30.655	2203	0.660	1.081	33642.2	69.5
200L_4	147814.83950	7.266	35.712	2338	0.822	1.176	35847.9	69.4
200L_5	140770.40692	10.095	51.264	2390	0.641	0.446	48529.7	59.5
200L_10	111323.49637	40.799	111.881	3025	1.908	1.222	87113.2	42.6
200L_15	95477.43756	76.780	172.681	3323	1.753	1.170	114854.8	30.8
200L_20	86652.80247	125.229	253.171	3704	1.112	0.687	145170.4	21.7
200T_4	158157.03172	12.823	40.427	2729	1.824	1.868	45351.3	66.9
200T_5	146861.18489	21.502	60.706	2940	1.752	1.555	59698.4	59.4
200T_10	114559.73125	60.012	129.488	3539	2.090	1.375	101708.8	42.5
200T_15	97145.07242	91.676	185.515	3632	2.413	1.883	127026.5	30.1
200T_20	87457.05109	116.015	241.654	3626	1.313	0.949	143964.7	20.7

Tabela 3.5 Rezultati GA2 na AP instancama velikih dimenzija

Inst.	Najb.reš. GA2	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
100L_2	189351.55220	3.386	9.238	3062	0.042	0.075	63514.5	58.6
100L_3	165714.45524	5.937	16.604	3069	0.006	0.019	81070.6	47.2
100L_4	146197.75674	4.573	16.566	2711	0.014	0.035	75511.4	44.4
100L_5	137232.52162	5.046	18.872	2678	0.087	0.162	79411.3	40.8
100L_10	107207.72249	10.969	29.531	3076	0.353	0.523	95923.6	37.7
100L_15	91285.26316	12.915	36.091	3052	0.573	0.33	96059.8	37.1
100L_20	81034.59491	15.808	43.294	3084	0.609	0.53	97763.2	36.7
100T_4	156486.75661	5.772	15.498	3103	0.487	0.513	86243.9	44.4
100T_5	143567.78336	8.336	21.672	3205	0.718	0.946	94624.1	41.0
100T_10	109633.40777	13.528	32.119	3332	1.191	0.777	103623.7	37.8
100T_15	92635.94839	11.458	34.64	2925	0.596	0.387	92245.7	37.0
100T_20	82425.42707	15.370	42.901	3050	0.689	0.498	96779.6	36.6
200L_2	182459.25446	30.658	58.915	3927	0.024	0.059	92531.9	52.9
200L_3	162887.03135	43.092	93.141	3655	0.002	0.004	105562.4	42.3
200L_4	147814.83950	50.356	107.518	3650	0.024	0.041	111076.5	39.2
200L_5	140563.76616	54.518	118.703	3615	0.262	0.158	114759.5	36.6
200L_10	110400.53915	102.042	182.995	4262	0.861	0.689	142324	33.3
200L_15	94525.39494	123.002	223.727	4347	0.611	0.412	147403.2	32.2
200L_20	85290.57664	143.854	261.82	4342	0.548	0.328	148658.9	31.6
200T_4	158161.97064	90.691	115.712	4886	0.368	0.512	145076.5	40.7
200T_5	143422.42519	97.293	139.989	4672	1.702	1.235	146129.9	37.5
200T_10	112056.60524	152.022	204.915	4919	2.505	1.374	162483.5	34.0
200T_15	96448.95340	128.705	223.062	4452	1.378	1.514	149817.9	32.7
200T_20	86496.09061	154.482	260.028	4389	1.04	0.551	149937.3	31.7

Tabela 3.6 Poređenje GA1 i GA2 na AP instancama velikih dimenzija

Inst.	GA1			GA2			bolji
	Najb.reš.GA1	agap[%]	t_{tot}[s]	Najb.reš.GA2	agap[%]	t_{tot}[s]	
100L_2	189351.55220	0.347	2.856	189351.55220	0.042	9.238	Isti
100L_3	165714.45524	0.720	7.961	165714.45524	0.006	16.604	Isti
100L_4	146197.75674	0.206	9.481	146197.75674	0.014	16.566	Isti
100L_5	137398.84844	1.033	14.384	137232.52162	0.087	18.872	GA2
100L_10	107381.77382	1.419	24.946	107207.72249	0.353	29.531	GA2
100L_15	91285.26316	1.375	42.328	91285.26316	0.573	36.091	Isti
100L_20	81230.96016	1.315	66.603	81034.59491	0.609	43.294	GA2
100T_4	157073.05820	2.973	9.459	156486.75661	0.487	15.498	GA2
100T_5	143567.78336	2.180	14.184	143567.78336	0.718	21.672	Isti
100T_10	109633.40777	1.665	29.443	109633.40777	1.191	32.119	Isti
100T_15	92947.09085	1.303	44.241	92635.94839	0.596	34.64	GA2
100T_20	82643.69838	1.089	64.463	82425.42707	0.689	42.901	GA2
200L_2	182459.25446	5.063	17.105	182459.25446	0.024	58.915	Isti
200L_3	162887.03135	0.660	30.655	162887.03135	0.002	93.141	Isti
200L_4	147814.83950	0.822	35.712	147814.83950	0.024	107.518	Isti
200L_5	140770.40692	0.641	51.264	140563.76616	0.262	118.703	GA2
200L_10	111323.49637	1.908	111.881	110400.53915	0.861	182.995	GA2
200L_15	95477.43756	1.753	172.681	94525.39494	0.611	223.727	GA2
200L_20	86652.80247	1.112	253.171	85290.57664	0.548	261.82	GA2
200T_4	158157.03172	1.824	40.427	158161.97064	0.368	115.712	GA1
200T_5	146861.18489	1.752	60.706	143422.42519	1.702	139.989	Isti
200T_10	114559.73125	2.090	129.488	112056.60524	2.505	204.915	Isti
200T_15	97145.07242	2.413	185.515	96448.95340	1.378	223.062	Isti
200T_20	87457.05109	1.313	241.654	86496.09061	1.04	260.028	Isti

Poredeći vremena izvršavanja CPLEX, GA1 i GA2 metoda na AP instancama manijh dimenzija, treba imati u vidu da se oba GA izvršavaju u dodatnom t_{tot}*[s]- t[s]* vremenu, iako je algoritam već dostigao optimalno/najbolje rešenje (videti sekciju 2.4). Ako uporedimo kolone *t[s]* za CPLEX, GA1 i GA2 u Tabelama 3.1, 3.2 i 3.3 respektivno, možemo zaključiti da obe predložene GA metode dostižu optimalno rešenje u značajno kraćem CPU vremenu u odnosu na CPLEX. Čak je i srednje ukupno CPU vreme predloženih GA metoda (kolona t_{tot}*[s]*) značajno kraće u odnosu na CPU vreme CPLEX-a na AP instancama sa n=20,25.40,50 čvorova, dok su u slučaju AP instanci sa n=10 čvorova vremena izvršavanja sve tri metode slična. Za AP instance praktičnih dimenzija sa n=100, 200 čvorova, srednje ukupno vreme GA1 i GA2 metoda je t_{tot}*[s]* ≤ 253.171, odnosno t_{tot}*[s]* ≤260.028. Srednje početno vreme je još kraće: *t[s]*≤125.229 za GA1 i *t[s]*≤ 154.482 za GA2, dok CPLEX u realnom CPU vremenu ne može da reši nijednu od ovih instanci (videti t_{tot}*[s]* i *t[s]* kolone u Tabelama 3.4 i 3.5). Na svim testiranim AP instancama problema, srednje ukupno vreme i srednje početno vreme izvršavanja za obe predložene GA metode su slična.

4 PROBLEM P-HAB CENTRA OGRANIČENIH KAPACITETA SA JEDNOSTRUKIM ALOKACIJAMA

U ovom poglavlju rešavamo još jednu varijantu hab lokacijskog problema sa ograničenim kapacitetima: problem p-hab centra sa jednostrukim alokacijama (Capacitated Single Allocation p-Hub Center Problem - CSApHCP). Problem se odnosi na uspostavljanje fiksiranog broja habova *p* u mreži sa ukupno *n* čvorova. Svaki čvor u mreži može biti pridružen samo jednom habu (šema jednostruke alokacije), tako da se transport između svaka dva ne-hab čvora odvija preko jednog ili najviše dva haba. Problem uključuje i ograničenja kapaciteta koja se odnose na količinu protoka koji ulazi u svaki hab čvor. Cilj CSApHCP je minimizacija maksimalnih transportnih troškova u mreži sa tačno *p* lociranih habova.

4.1 Formulacija problema

U literaturi do sada ne nalazimo radove koji se bave rešavanjem CSApHCP. Problem je NP kompletan, jer je njegov potproblem sa neograničenim kapacitetima USApHCP takođe NP-kompletan, što je dokazano u [Kara99b] i [Ern04b]. Kako CSApHCP do sada nije razmatran u literaturi, u ovoj sekciji predlažemo jednu formulaciju problema koja koristi sledeće oznake:

C_{ij} = rastojanje između čvorova *i* i *j* (u smislu metrike),

W_{ij} = količina protoka (broj jedinica količine robe) od čvora-snabdevača *i* do čvora-korisnika *j,*

Γ_k = kapacitet haba *k,*

O_i = količina protoka koji polazi iz čvora *i*, tj. $O_i = \sum_{j=1}^{n} W_{ij}$,

Promenljive CSApHCP modela su Z_{ij} = 1 ako je čvor *I* pridružen habu *k*, 0 inače.

Koristeći gornju notaciju, CSApHCP se može matematički zapisati na sledeći način:

min z (4.1)

uz uslove:

$$\sum_{k=1}^{n} Z_{kk} = p \quad (4.2)$$

$$\sum_{k=1}^{n} Z_{ik} = 1, \quad \text{za svako i=1,…,n} \quad (4.3)$$

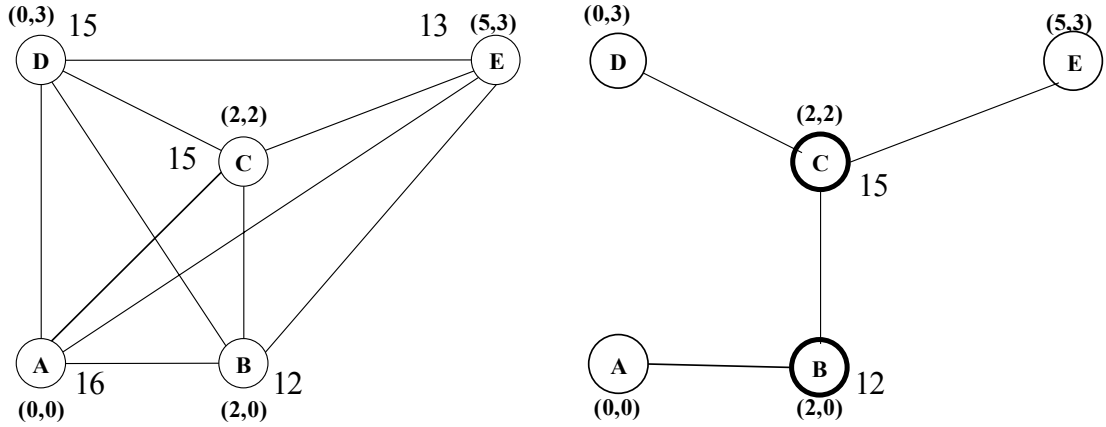
$$Z_{ik} \le Z_{kk}, \quad \text{za svako i,k=1,…,n} \quad (4.4)$$

$$\sum_{i=1}^{n} O_i Z_{ik} \le \Gamma_k Z_{kk}, \quad \text{za svako k=1,…,n} \quad (4.5)$$

$$z \ge \sum_{k=1}^{n} (\chi C_{ik} + \alpha C_{kl}) Z_{ik} + \delta C_{lj} Z_{jl} \quad \text{za svako i,j,l=1,…,n} \quad (4.6)$$

$$Z_{ik} \in \{0,1\}, \quad \text{za svako i,k=1,….n} \quad (4.7)$$

Funkcija cilja CSApHCP minimizuje maksimalne troškove transporta između bilo koja dva čvora u mreži (4.1). Uslov (4.2) fiksira broj habova koje treba uspostaviti na *p.* Ograničenja (4.3) i (4.4) obezbeđuju da svaki korisnik/snabdevač bude pridružen tačno jednom, prethodno lociranom habu. Uslovom (4.5) se ograničava količina protoka koji dolazi u svaki hab. Donja granica za promenljivu z funkcije cilja je data uslovom (4.6). Promenljive Z_{ik} imaju binarnu reprezentaciju, što je određeno uslovom (4.7). Koeficijenti χ, α i δ imaju isto značenje kao u prethodnim poglavljima.



Slika 4.1 Hab mreža sa n=5, p=2 i $\chi=\delta=1$, α=0.25

Posmatrajmo još jednom primer mreže (n=5, p=2), iz prethodih poglavlja, sa istim ograničenjima kapaciteta čvorova (slika 4.1). U optimalnom rešenju CSApHCP uspostavljeni su habovi **B** i **C**, ne-hab čvorovi D i E pridruženi su habu **A**, a ne-hab čvor A habu **B**. Primetimo da se zbog dodatnih ograničenja kapaciteta, optimalno rešenje za CSApHCP razlikuje od optimalnog rešenja za USApHCP na istoj mreži čvorova (videti sliku 2.1). Očigledno je, da se zbog različitih funkcija cilja, optimalna rešenja za CSApHCP i CSApHMP na datoj mreži takođe razlikuju (videti sliku 3.1).

Troškovi transporta između svakog para čvorova u mreži su: “A-**B**-A” 2+2=4, “A-B” 2, “A-**B**-**C**” 2+2*0.25=2.5 , ”A-**B**-**C**-D” 2+2*0.25+$\sqrt{5}$=4.736, ”A-**B**-**C**-E” 2+2*0.25+$\sqrt{10}$=5.662, “**B**-**B**” 0, “**B**-**C**” 2*0.25=0.5, “**B**-**C**-D” 2*0.25+$\sqrt{5}$=2.736, “**B**-**C**-E” 2*0.25+$\sqrt{10}$=3.662, “**C**-**C**” 0, “**C**-D” $\sqrt{5}$, ”**C**-E” $\sqrt{10}$, “D-**C**-D” $\sqrt{5}+\sqrt{5}$=4.472, “D-**C**-E” $\sqrt{5}+\sqrt{10}$=5.398, “E-**C**-E” $\sqrt{10}+\sqrt{10}$=6.324. Vrednost funkcije cilja za CSApHCP (maksimalni transportni troškovi) u ovom slučaju se dobija za transport "E-**C**-E" i iznosi 6.324.

4.2 Predložene GA implementacije, rezultati i poređenja

Imajući u vidu da CSApHCP i CSApHMP imaju različite funkcije cilja, ali slična ograničenja, GA1 i GA2 implementacije koje su opisane u prethodnom poglavlju, se uz male modifikacije programskog koda mogu iskoristiti i za rešavanje CSApHCP.

Predložene GA implementacije koriste iste načine kodiranja, genetske operatore i GA parametre opisane u sekciji 3.3. Takođe je primenjena ista politika zamene generacija, kao i keširanje. Maksimalan broj generacija GA1 i GA2 je N_{gen}=5000. Algoritmi se takođe zaustavlja ako je najbolja jedinka ili najbolja vrednost funkcije cilja ostala nepromenjena tokom N_{rep}=2000 uzastopnih generacija.

U ovoj sekciji prikazani su i rezultati GA1 i GA2 metoda za rešavanje CSApHCP, kao i poređenja sa rezultatima dobijenim korišćenjem solvera CPLEX 8.1.0. GA implementacije su kodirane u programskom jeziku C i testirane na AMD Sempron procesoru na 1597 MHz sa 256 MB RAM memorije pod Linux (Knoppix 3.7) operativnom sistemu. Testiranja sa CPLEX-om su izvršena na Intel procesoru na 1.8 GHz sa 256 MB RAM memorije. Sva tri algoritma su testirana na standardnim AP hab instancam [Bea96] sa n≤200 čvorova i p≤20 habova. Korišćena su dva tipa ograničenja kapaciteta na habovima: jaka (tight -T) i slaba (loose-L). Na svakoj AP instanci problema predloženi GA1 i GA2 izvršeni su po 20 puta.

Tabela 4.1. Rezultati CPLEX-a na instancama manjih dimenzija

Inst.	CPLEX			
	Opt.reš.	t[s] Intel 1.8 GHz	N_{iter}	$N_{čvorova}$
10L_2	40382.6537	1.490	529	10
10L_3	34772.3751	1.360	244	0
10L_4	32574.2031	1.520	589	11
10L_5	32531.213	1.000	489	0
10T_2	45169.777	1.800	516	0
10T_3	32191.4032	1.820	423	0
10T_4	32574.2031	1.020	471	7
10T_5	32531.213	0.450	241	0
20L_2	45954.1514	63.580	2802	0
20L_3	43400.4457	91.410	13045	100
20L_4	38607.295	45.920	5749	10
20L_5	37868.148	130.000	14102	119
20L_10	37868.148	40.360	6004	13
20T_2	56078.3218	46.040	2343	0
20T_3	43643.316	77.400	11813	200
20T_4	38607.295	50.350	6166	10
20T_5	37868.148	44.200	6334	21
20T_10	37868.148	1.070	257	0
25L_2	51202.3203	520.650	26683	200
25L_3	46608.3144	352.290	5250	0
25L_4	45552.4969	834.880	40610	30
25L_5	45552.4969	252.200	5238	50
25L_10	45552.4969	110.590	3323	10
25T_3	56589.8643	1148.370	124753	1400
25T_4	51205.7005	620.840	46760	300
25T_5	45552.4969	283.570	11809	245
25T_10	45552.4969	3.480	361	0
40L_2	54368.094	7208.260	39500	30
40L_3	50971.9338	7214.030	18909	0
40L_4	49741.2007	7205.730	49235	60
40L_5	49741.2007	3798.990	32321	280
40L_10	49741.2007	1036.150	6226	220
40T_3	55384.2781	7208.050	61102	110
40T_4	52799.3265	7210.070	35416	0
40T_5	49741.2007	204.970	58647	120
40T_10	49741.2007	7044.370	91463	240
50L_2	-----	>2sata	34500	0
50L_3	51737.6297	7231.030	24962	0
50L_4	-----	>2sata	35600	0
50L_5	50707.8663	7223.630	37304	0
50L_10	50707.8663	7225.400	6713	0
50T_3	-----	>2sata	35000	0
50T_4	-----	>2sata	40800	0
50T_5	50707.8663	7241.750	11047	0
50T_10	50707.8663	7227.360	15931	0

Optimalna rešenja dobijena korišćenjem CPLEX-a na AP instancama manjih dimenzija ($n \leq 50$, $p \leq 10$) prikazana su u Tabeli 4.1, zajedno sa vremenom izvršavanja, brojem iteracija i brojem čvorova BnB drveta za svaku od testiranih instanci. Instance koje nedostaju u tabeli (25T_2, 40T_2 i 50T_2) su nedopustive jer je zbir kapaciteta habova manji od količine protoka koja polazi iz čvorova snabdevača. Na instancama 50L_2, 50_4, 50T_3 i 50T_4 CPLEX algoritam nije dao rešenje ni posle dva sata izvršavanja (oznaka ---).

U Tabelama 4.2 i 4.3 dati su rezultati predloženih GA1 i GA2 implementacija respektivno na istim AP instancama manjih dimenzija ($n \leq 50$, $p \leq 10$). Rezultati su prikazani na sličan način kao u prethodnim poglavljima (ime instance, optimalno

rešenje, najbolje rešenje GA, srednje početno i srednje ukupno vreme izvršavanja,...).

Tabela 4.2 Rezultati GA1 na instancama manjih dimenzija

Inst.	Opt.reš. CPLEX	GA1						
		Najb.reš.GA1	t[s]	t_{tot}[s]	gen	agap[%]	eval	kеš[%]
10L_2	40382.6537	opt	0.001	0.401	2001	0.000	7073.4	92.9
10L_3	34772.3751	opt	0.002	0.764	2001	0.000	33718.3	66.3
10L_4	32574.2031	opt	0.002	0.837	2001	0.000	28404.0	71.7
10L_5	32531.213	opt	0.004	1.142	2001	0.000	39296.9	60.8
10T_2	45169.777	46491.9768	0	0.374	2001	2.927	7247.4	92.8
10T_3	32191.4032	opt	0.002	0.601	2004	8.018	24217.2	75.9
10T_4	32574.2031	opt	0.001	0.779	2001	0.000	27728.8	72.3
10T_5	32531.213	opt	0.003	1.115	2001	0.000	39088.1	61.0
20L_2	45954.1514	opt	0.005	1.131	2004	0.000	25568.3	74.5
20L_3	43400.4457	opt	0.006	1.820	2001	0.000	35077	65.0
20L_4	38607.295	opt	0.092	2.085	2061	0.000	40019.1	61.2
20L_5	37868.148	opt	0.03	2.698	2016	0.000	46301.3	54.1
20L_10	37868.148	opt	0.007	4.773	2001	0.000	66691.6	33.4
20T_2	56078.3218	----	0	0.001	1		150.0	0.0
20T_3	43643.316	45520.3371	0.162	1.354	2214	4.301	37709.1	66.0
20T_4	38607.295	opt	0.023	1.498	2015	0.000	30630.4	69.7
20T_5	37868.148	opt	0.048	2.443	2027	0.000	42286.1	58.4
20T_10	37868.148	opt	0.008	4.954	2001	0.000	66713.5	33.4
25L_2	51202.3203	53207.4592	0.029	1.516	2028	3.916	28029.5	72.4
25L_3	46608.3144	opt	0.039	2.768	2021	0.000	40104.6	60.4
25L_4	45552.4969	opt	0.041	3.213	2021	0.000	47780.0	52.8
25L_5	45552.4969	opt	0.024	3.611	2008	0.000	54734.8	45.6
25L_10	45552.4969	opt	0.009	5.717	2001	0.000	70421.7	29.7
25T_3	56589.8643	opt	0.376	1.801	2417	0.000	41890.9	65.4
25T_4	51205.7005	opt	0.286	2.421	2226	0.000	45658.2	59.1
25T_5	45552.4969	opt	0.038	3.178	2018	0.000	48621.3	51.9
25T_10	45552.4969	opt	0.01	5.774	2001	0.000	69825.9	30.3
40L_2	54368.094	61682.5012	0.461	4.244	2260	13.453	43442.2	61.4
40L_3	50971.9338	58747.9748	1.136	7.328	2359	15.256	54875.8	53.6
40L_4	49741.2007	52265.2745	2.099	9.865	2561	5.074	61421.7	52.1
40L_5	49741.2007	opt	1.018	9.803	2239	0.000	58590.6	47.8
40L_10	49741.2007	opt	0.015	11.378	2001	0.000	68359.4	31.8
40T_3	55384.2781	62972.8933	0.141	2.875	2004	13.702	44644.1	52.8
40T_4	52799.3265	53380.7928	1.231	5.255	2541	1.101	61227.8	51.4
40T_5	49741.2007	opt	3.029	9.834	2782	0.000	71550.4	48.7
40T_10	49741.2007	opt	0.017	13.694	2001	0.000	67948.9	32.2
50L_2	-----	66080.3548	0.127	4.297	2051		42037.8	59.1
50L_3	51737.6297	60550.1271	2.57	12.769	2560	17.033	59594.1	53.6
50L_4	-----	52905.7696	2.553	16.637	2423		61778.6	49.1
50L_5	50707.8663	opt	2.577	17.301	2380	0.000	63898.5	46.5
50L_10	50707.8663	opt	0.094	15.705	2009	0.000	68385.4	32.1
50T_3	-----	63152.8184	0.378	4.564	2193		52814.5	51.9
50T_4	-----	53010.9807	4.322	10.837	3173		78663.1	50.4
50T_5	50707.8663	51215.9916	8.01	22.625	3306	1.002	89270.5	46.0
50T_10	50707.8663	opt	0.201	16.594	2021	0.000	66807.4	34.0

Iz Tabele 4.2 vidimo da GA1 na AP instancama manjih dimenzija dostižu optimalna rešenja dobijena CPLEX-om u 30 od ukupno 41 slučajeva. Na instancama 50L_2, 50L_4, 50T_3 i 50T_4, koje CPLEX nije rešio ni posle dva sata izvršavanja, GA1 je našao rešenje za t_{tot}[s]≤22.625 sekundi. U slučaju instanci 10T_2, 20T_3, 25L_2, 40L_2, 40L_3, 40L_4, 40T_3, 40T_4, 50L_3 i 50T_5 srednje odstupanje najboljeg rešenja GA1 od optimalnog je 1.101% ≤ *agap[%]* ≤17.033%. Na instanci 20T_2 GA1 nije dao rešenje.

Rezultati GA2 prikazani u Tabeli 4.3 pokazuju da je ova GA implementacija dala slične rezultate kao GA1. Najbolje rešenje GA2 se poklapa sa optimalnim u slučaju 31 instance, od ukupno 40 za koje je optimalno rešenje dobijeno

CPLEX-om. GA2 je takođe rešio instance 50L_2, 50L_4, 50T_3 i 50T_4 za koje optimalno rešenje nije poznato za t_{tot}[s]≤24.855 sekundi. Na instancama: 10T_2, 20T_3, 25L_2, 40L_2, 40L_3, 40L_4, 40T_3, 40T_4, 50L_3 i 50T_5 srednje odstupanje najboljeg rešenja GA1 od optimalnog je 0.374% ≤ *agap[%]* ≤22.625%. Za instance 40T_3 i 50T_3 GA2 nije dao rešenje (instancu 50T_3 ni GA2 ni CPLEX algoritam nisu rešili, za razliku od GA1).

Tabela 4.3 Rezultati GA2 na instancama manjih dimenzija

Inst.	Opt.reš.	GA2						
		Najb.reš.GA2	t[s]	t_{tot}[s]	gen	agap[%]	eval	keš[%]
10L_2	40382.6537	opt	0.002	0.514	2001	0.000	28490.0	71.6
10L_3	34772.3751	opt	0.002	1.197	2001	0.000	75142.9	25.0
10L_4	32574.2031	opt	0.001	1.673	2001	0.000	85802.8	14.4
10L_5	32531.213	opt	0.004	1.948	2003	0.000	91716.9	8.6
10T_2	45169.777	46491.9768	0.001	0.428	2001	2.927	33756.7	66.3
10T_3	32191.4032	34772.3751	0.003	0.929	2004	8.018	71017.1	29.3
10T_4	32574.2031	opt	0.002	1.461	2001	0.000	85110.1	15.1
10T_5	32531.213	opt	0.004	1.938	2003	0.000	91071.1	9.2
20L_2	45954.1514	opt	0.085	1.933	2097	0.000	52860.1	49.7
20L_3	43400.4457	opt	0.010	2.763	2005	0.000	67809.3	32.5
20L_4	38607.295	opt	0.130	3.098	2077	0.000	75419.6	27.4
20L_5	37868.148	opt	0.037	3.957	2017	0.000	83151.9	17.7
20L_10	37868.148	opt	0.010	6.471	2001	0.000	97201.8	3.0
20T_2	56078.3218	opt	0.001	0.455	1402	0.000	32816.3	38.8
20T_3	43643.316	opt	0.079	1.951	2066	4.301	65659.7	36.5
20T_4	38607.295	opt	0.350	2.972	2262	0.000	83543.4	26.1
20T_5	37868.148	opt	0.059	3.782	2029	0.000	81513.3	19.8
20T_10	37868.148	opt	0.007	6.678	2001	0.000	97168.1	3.0
25L_2	51202.3203	53207.4592	0.209	2.483	2267	3.916	49672.1	56.2
25L_3	46608.3144	opt	0.022	3.910	2010	0.000	65372.2	35.1
25L_4	45552.4969	opt	0.025	4.255	2010	0.000	73611.4	26.9
25L_5	45552.4969	opt	0.014	4.840	2003	0.000	83461.9	16.8
25L_10	45552.4969	opt	0.010	7.567	2001	0.000	95788.4	4.4
25T_3	56589.8643	opt	0.554	2.802	2398	0.000	69955.8	41.6
25T_4	51205.7005	opt	0.818	3.818	2517	0.000	82689.6	34.2
25T_5	45552.4969	opt	0.029	4.449	2011	0.000	79706.9	20.9
25T_10	45552.4969	opt	0.010	7.589	2001	0.000	95791.3	4.4
40L_2	54368.094	61682.5012	0.316	7.330	2123	13.453	51473.3	51.5
40L_3	50971.9338	58192.7613	2.169	11.304	2459	14.166	77729.0	36.6
40L_4	49741.2007	52265.2745	3.443	15.353	2613	5.074	89438.2	31.8
40L_5	49741.2007	opt	0.140	13.035	2025	0.000	78718.5	22.4
40L_10	49741.2007	opt	0.040	13.498	2005	0.000	92821.4	7.6
40T_3	55384.2781	----------	0.000	0.002	1		150.0	0.0
40T_4	52799.3265	53380.7928	2.083	8.606	2663	1.101	88681.9	32.7
40T_5	49741.2007	opt	2.949	13.819	2556	0.000	95143.1	25.7
40T_10	49741.2007	opt	0.047	14.950	2005	0.000	92106.8	8.3
50L_2	-----	66080.3548	1.041	9.463	2255		51534.7	54.3
50L_3	51737.6297	60747.9082	0.373	11.219	2056	17.415	60836.8	41.0
50L_4	-----	52905.7696	1.841	17.383	2315		74624.7	35.5
50L_5	50707.8663	opt	3.042	22.252	2366	0.000	85475.5	27.9
50L_10	50707.8663	opt	0.023	21.428	2001	0.000	91556.9	8.6
50T_3	-----	----------	0.000	0.001	1		150.0	0.0
50T_4	-----	54433.6309	3.462	13.386	2677		82614.4	38.6
50T_5	50707.8663	51215.9916	4.564	21.823	2593	1.002	91608.6	29.5
50T_10	50707.8663	opt	0.076	24.855	2006	0.000	90476.8	10.0

Na AP instancama sa n=10 čvorova, vremena izvršavanja CPLEX-a i predloženih GA1 i GA2 implementacija su slična, dok su već za instance sa n=20 srednja ukupna vremena izvršavanja GA1 i GA2 dvadesetak puta kraća u odnosu na odgovarajuća vremena CPLEX-a (t[s] ≤130 sekundi za CPLEX, t_{tot}[s]≤ 4.954 za GA1 i t_{tot}[s]≤6.471 za GA2). Za instance sa n=25 i n=40

čvorova, razlika u vremenima izvršavanja je još veća, tj. srednja ukupna vremena GA1 i GA2 su i do nekoliko stotina puta kraća u odnosu na odgovarajuća vremena CPLEX-a (t[s]≤7214.030 za CPLEX, t_{tot}[s]≤ 13.694 za GA1 i t_{tot}[s]≤ 15.35300 za GA2). Za one instance sa n=50 čvorova koje je CPLEX optimalno rešio, bilo je potrebno oko dva sata procesorskog vremena, dok su GA1 i GA2 dali rešenja istog ili nešto lošijeg kvaliteta za manje od 25 sekundi.

Poredeći vremena izvršavanja CPLEX, GA1 i GA2 metoda na AP instancama manijh dimenzija, takođe treba imati u vidu da se oba GA izvršavaju u dodatnom t_{tot}*[s]- t[s]* vremenu, iako je optimalno rešenje već dostignuto. Ako uporedimo kolone *t[s]* u Tabelama 4.1-4.3 vidimo da su srednja početna vremena za koje algoritmi GA1 i GA2 prvi put dostižu optimalno rešenje na istoj instanci su znatno kraća: t[s] ≤ 8.01 sekundi za GA1 i t[s] ≤ 4.564 sekundi za GA2.

U Tabelama 4.4 i 4.5 prikazani su rezultati predloženih GA implementacija na AP instancama praktičnih dimenzija sa n=100,200 čvorova, p≤20 habova, (tabele su organizovane na isti način kao prethodne). Algoritam CPLEX je takođe testiran na ovim AP instancama, ali pokazalo se da je na instancama velikih dimenzija vreme izvršavanja CPLEX izuzetno dugo. CPLEX uglavnom nalazi gornje granice rešenja, ali ne i optimalno rešenje problema, čak ni ni posle nekoliko dana neprekidnog izvršavanja.

Tabela 4.4 Rezultati GA1 na instancama velikih dimenzija

Inst.	GA1							
	Najb.reš.GA1	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	kеš[%]
100L_2	66346.85012	3.758	19.012	2568	0.980	0.821	56099.2	56.5
100L_3	60658.88360	23.904	77.118	2952	1.908	1.049	80964.3	45.3
100L_4	56124.74052	9.282	64.091	2378	0.581	0.976	68887.0	42.1
100L_5	54243.49699	6.620	67.022	2226	0.548	0.538	67961.1	39.0
100L_10	51860.02625	3.074	70.952	2090	0.000	0.000	72358.2	30.9
100L_15	51860.02625	0.757	70.821	2021	0.000	0.000	77854.4	23.1
100L_20	51860.02625	0.093	74.364	2001	0.000	0.000	79322.3	20.8
100T_4	56434.44649	18.913	57.499	3105	2.519	2.994	87608.9	43.6
100T_5	54360.60347	21.294	73.033	2770	1.773	1.911	85232.5	38.5
100T_10	51860.02625	4.625	78.380	2119	0.000	0.000	72894.6	31.3
100T_15	51860.02625	0.982	82.039	2026	0.000	0.000	77849.4	23.3
100T_20	51860.02625	0.094	81.525	2001	0.000	0.000	80453.3	19.7
200L_2	71538.38424	6.248	132.197	2116	0.000	0.000	54003.0	49.0
200L_3	64237.35489	84.770	363.077	2507	0.586	1.055	74639.5	40.5
200L_4	60958.51479	150.760	532.069	2827	0.175	0.217	88493.9	37.5
200L_5	58918.61584	116.096	489.274	2756	0.571	0.866	89401.3	35.2
200L_10	55958.75081	35.741	414.697	2205	0.000	0.000	80039.4	27.5
200L_15	55958.75081	8.220	360.062	2053	0.000	0.000	82795.6	19.5
200L_20	55958.75081	4.650	358.516	2031	0.000	0.000	83730.6	17.7
200T_4	61984.45476	107.346	344.090	3069	1.072	0.895	93439.9	39.1
200T_5	59126.02128	139.263	444.356	2880	1.820	1.839	93682.8	35.0
200T_10	55958.75081	59.035	447.947	2294	0.005	0.021	81663.1	28.9
200T_15	55958.75081	11.319	418.459	2066	0.000	0.000	83570.0	19.2
200T_20	55958.75081	5.929	398.412	2034	0.000	0.000	83846.8	17.7

Tabela 4.5 Rezultati GA2 na instancama velikih dimenzija

Inst.	GA2							
	Najb.reš.GA2	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
100L_2	66346.85012	2.188	25.091	2229	0.857	0.816	52436.2	53.3
100L_3	60658.88360	15.583	72.855	2557	2.931	1.021	73167.1	42.6
100L_4	56434.44649	2.457	59.128	2118	2.479	2.299	67612.3	36.1
100L_5	54243.49699	11.232	88.148	2309	1.158	1.316	80390.4	30.5
100L_10	51860.02625	21.085	114.821	2480	0.000	0.000	104303.6	15.8
100L_15	51860.02625	0.436	81.694	2012	0.000	0.000	92185.4	8.5
100L_20	51860.02625	0.243	80.264	2005	0.000	0.000	95317.1	5.1
100T_4	57423.25844	24.909	72.989	3059	5.362	2.723	90175.6	40.8
100T_5	54360.60347	26.036	94.412	2785	4.789	2.845	91481.5	34.0
100T_10	51860.02625	17.497	116.069	2401	0.077	0.345	101613.8	15.4
100T_15	51860.02625	0.410	95.179	2010	0.000	0.000	92418.9	8.2
100T_20	51860.02625	0.253	82.454	2005	0.000	0.000	95244.8	5.2
200L_2	71538.38424	2.991	185.631	2033	0.108	0.136	45090.3	55.7
200L_3	64753.25146	57.222	465.712	2301	2.235	1.355	62902.9	45.3
200L_4	60972.68290	121.111	457.790	2737	3.571	2.222	79049.1	42.3
200L_5	58918.61584	123.241	543.447	2598	2.807	1.818	83073.8	35.7
200L_10	55958.75081	89.479	498.364	2484	1.617	1.546	96004.9	22.8
200L_15	55958.75081	32.602	407.603	2204	0.162	0.498	96645.4	12.4
200L_20	55958.75081	2.273	340.166	2015	0.000	0.000	92070.9	8.8
200T_4	62813.19218	50.948	449.654	2328	5.014	1.993	66711.1	42.8
200T_5	59806.92995	105.423	480.146	2563	4.368	2.237	81407.4	36.4
200T_10	55958.75081	135.330	619.324	2585	2.383	2.017	100416.2	22.4
200T_15	55958.75081	40.980	493.369	2204	0.081	0.362	96618.9	12.4
200T_20	55958.75081	5.160	406.219	2029	0.000	0.000	93333.9	8.1

Tabela 4.6 Poređenje GA1 i GA2 na AP instancama velikih dimenzija

Inst.	GA1			GA2			bolji
	Najb.reš.GA1	agap[%]	t_{tot}[s]	Najb.reš.GA2	agap[%]	t_{tot}[s]	
100L_2	66346.85012	0.980	19.012	66346.85012	0.857	25.091	isti
100L_3	60658.88360	1.908	77.118	60658.88360	2.931	72.855	isti
100L_4	56124.74052	0.581	64.091	56434.44649	2.479	59.128	HGA1
100L_5	54243.49699	0.548	67.022	54243.49699	1.158	88.148	isti
100L_10	51860.02625	0.000	70.952	51860.02625	0.000	114.821	isti
100L_15	51860.02625	0.000	70.821	51860.02625	0.000	81.694	isti
100L_20	51860.02625	0.000	74.364	51860.02625	0.000	80.264	isti
100T_4	56434.44649	2.519	57.499	57423.25844	5.362	72.989	HGA1
100T_5	54360.60347	1.773	73.033	54360.60347	4.789	94.412	isti
100T_10	51860.02625	0.000	78.380	51860.02625	0.077	116.069	isti
100T_15	51860.02625	0.000	82.039	51860.02625	0.000	95.179	isti
100T_20	51860.02625	0.000	81.525	51860.02625	0.000	82.454	isti
200L_2	71538.38424	0.000	132.197	71538.38424	0.108	185.631	isti
200L_3	64237.35489	0.586	363.077	64753.25146	2.235	465.712	HGA1
200L_4	60958.51479	0.175	532.069	60972.68290	3.571	457.790	HGA1
200L_5	58918.61584	0.571	489.274	58918.61584	2.807	543.447	isti
200L_10	55958.75081	0.000	414.697	55958.75081	1.617	498.364	isti
200L_15	55958.75081	0.000	360.062	55958.75081	0.162	407.603	isti
200L_20	55958.75081	0.000	358.516	55958.75081	0.000	340.166	isti
200T_4	61984.45476	1.072	344.090	62813.19218	5.014	449.654	HGA1
200T_5	59126.02128	1.820	444.356	59806.92995	4.368	480.146	HGA1
200T_10	55958.75081	0.005	447.947	55958.75081	2.383	619.324	isti
200T_15	55958.75081	0.000	418.459	55958.75081	0.081	493.369	isti
200T_20	55958.75081	0.000	398.412	55958.75081	0.000	406.219	isti

S obzirom da optimalna rešenja za AP instance velikih dimenzija nisu poznata, u Tabeli 4.6 dato je poređenje najboljih rešenja GA1 i GA2 metoda i odgovarajućih vremena izvršavanja. Iz poslednje kolone tabele 4.6 možemo videti da su predložene GA implementacije u najvećem broju slučajeva dostigle ista najbolja rešenja, dok je u slučaju šest AP instanci: 100L_4, 100T_4, 200L_3, 200L_4, 200T_4 i 200T_5 predložena GA1metoda je dala bolja rešenja u odnosu na GA2. Srednje procentualno odstupanje od najboljeg dobijenog rešenja agap[%] ≤ 2.519 za GA1 i agap[%] ≤ 5.362 za GA2 implementaciju.

Obe predložene GA implementacije u relativno kratkom CPU vremenu daju rešenja za sve AP instance velikih dimenzija (videti *t[s]* i t_{tot}*[s]* kolone u Tabelama 4.4 i 4.5). Srednje ukupno vreme GA1 i GA2 metoda je t_{tot}*[s]* ≤ 532.069, odnosno t_{tot}*[s]* ≤619.324 sekunde. Srednje početno vreme je još kraće: *t[s]*≤150.760 za GA1 i *t[s]*≤ 135.330 za GA2. Na svim testiranim AP instancama problema, srednja ukupna vremena i srednja početna vremena izvršavanja za predložene GA metode su slična.

5 HAB LOKACIJSKI PROBLEM OGRANIČENIH KAPACITETA SA JEDNOSTRUKIM ALOKACIJAMA

5.1 Formulacija problema

Ovo poglavlje se bavi rešavanjem hab lokacijskog problema ograničenih kapaciteta sa jednostrukim alokacijama (Capacitated Single Allocation Hub Location Problem-CSAHLP). Ovaj model ima šemu jednostruke alokacije, što znači da se ukupna količina protoka iz nekog čvora-snabdevača može sakupljati samo u jednom habu, odnosno distribuirati nekom čvoru-korisniku iz samo jednog hab čvora. Količina protoka koja dolazi u svaki hab čvor je ograničena i broj habova koje je potrebno uspostaviti nije unapred poznat. Uspostavljanje svakog haba donosi izvesne fiksne troškove. Zadatak CSAHLP je da se izabere skup habova i alociraju ne-hab čvorovi elementima tog izabranog skupa, tako da je suma transportnih troškova i fiksnih troškova minimalna

Formulacija problema, preuzeta iz [Ern99], koju koristimo u ovom poglavlju koristi sledeće oznake:

C_{ij} = rastojanje između čvorova *i* i *j* (u smislu metrike),

W_{ij} = količina protoka (broj jedinica količine robe) od čvora-snabdevača *i* do čvora-korisnika *j,*

Γ_k = kapacitet haba *k,*

F_k = troškovi uspostavljanja haba *k,*

O_i = količina protoka koji polazi iz čvora *i*, tj. $O_i = \sum_{j=1}^{n} W_{ij}$,

D_j = količina protoka koji dolazi u čvor *j,* tj. $D_j = \sum_{i=1}^{n} W_{ij}$.

Promenljive CSAHLP modela su:

Z_{ij} = 1 ako je čvor *I* pridružen habu *k*, 0 inače,

Y_{kl}^{i} = količina protoka koja polazi od čvora-snabdevača *I,* sakuplja se u habu *k* I distribuira preko haba *l.*

Koristeći gornju notaciju, CSAHLP se može matematički zapisati na sledeći način:

$$\min \sum_{i=1}^{n}\sum_{k=1}^{n} C_{ik} Z_{ik} (\chi O_i + \delta D_i) + \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} \alpha C_{kl} Y_{kl}^{i} + \sum_{k=1}^{n} F_k Z_{kk} \qquad (5.1)$$

uz uslove:

$$\sum_{k=1}^{n} Z_{ik} = 1, \qquad \text{za svako i=1,…,n} \qquad (5.2)$$

$$Z_{ik} \le Z_{kk}, \qquad \text{za svako i,k=1,…,n} \qquad (5.3)$$

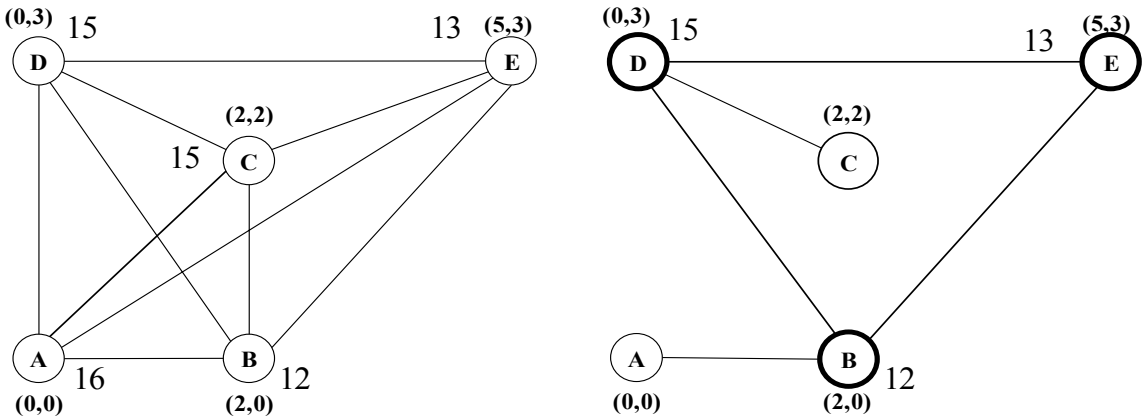
$$\sum_{j=1}^{n} W_{ij} Z_{jk} + \sum_{l=1}^{n} Y_{kl}^{i} = \sum_{l=1}^{n} Y_{lk}^{i} + O_i Z_{ik}, \qquad \text{za svako i,k=1,…,n} \qquad (5.4)$$

$$\sum_{i=1}^{n} O_i Z_{ik} \le \Gamma_k Z_{kk}, \qquad \text{za svako k=1,…,n} \qquad (5.5)$$

$$Y_{kl}^{i} \ge 0, \qquad \text{za svako i,k,l=1,…,n} \qquad (5.6)$$

$$Z_{ik} \in \{0,1\}, \qquad \text{za svako i,k=1,….n} \qquad (5.7)$$

Cilj CSAHLP je minimizacija ukupnih troškova transporta u mreži i fiksnih troškova uspostavljanja habova (5.1). Ograničenja (5.2) i (5.3) obezbeđuju da svaki ne-hab čvor bude pridružen tačno jednom, prethodno uspostavljenom habu, čime se sprečava direktna komunikaciju između ne-hab čvorova. Uslov (5.4) se odnosi na konzervaciju protoka u mreži, dok (5.5) ograničava količinu protoka koji dolazi u svaki hab. Konačno, (5.6) i (5.7) ukazuju na ne-negativnu, odnosno binarnu reprezentaciju promenljivih Y_{kl}^{i} i Z_{ik}. Koeficijenti χ, α i δ imaju isto značenje kao u prethodnom poglavlju.



Slika 5.1 Hab mreža sa n=5 i $\chi=\delta=1$, α=0.25

Na levoj strani slike 5.1 ponovo vidimo mrežu čvorova iz prethodnih poglavlja, sa istim matricama rastojanja, protoka i ograničenjima kapaciteta. Fiksni troškovi uspostavljanja habova na lokacijama A, B, C, D, E su 70, 25, 40, 32 i 22 respektivno. U optimalnom rešenju CSAHLP za datu mrežu, habovi su locirani u čvorovima **B**, **D** i **E** (broj habova nije unapred zadat), ne-hab čvor A je pridružen habu **B**, a ne-hab čvor C habu **D**. Troškovi transporta između svakog para čvorova u mreži su: "A-**B**-A" 2+2=4, "A-**B**" 2, "A-**B**-**D**-C" 2+$\sqrt{13}$ *0.25+$\sqrt{5}$ =5.137," A-**B**-**D**" 2+$\sqrt{13}$ +0.25=2.901, "A-**B**-**E**"

2+3*$\sqrt{2}$*0.25=3.061, "**B**-**B**" 0, "**B**-**D**-C" $\sqrt{13}$*0.25+$\sqrt{5}$=3.137, "**B**-**D**" $\sqrt{13}$*0.25=0.901, "**B**-**E**" 3*$\sqrt{2}$*0.25=1.061, "C-**D**-C" 2*$\sqrt{5}$=4.472, "C-**D**" $\sqrt{5}$, "C-**D**-**E**" $\sqrt{5}$+5*0.25=3.486, "**D**-**D**" 0, "**D**-**E**" 5*0.25=1.25, "**E**-**E**" 0.
Ukupni transportni troškovi i fiksni troškovi uspostavljanja habova su 137.814.

5.2 Postojeći načini rešavanja

U literaturi nalazimo veoma malo radova koji se odnose na CSAHLP. Potproblem ovog problema koji nema ograničenja kapaciteta na habovima (Uncapacitated Single Allocation Hub Location Problem-USAHLP) proučavan je u [OK92] i [Cam94a]. U radu [Kara98] dokazana je NP-kompletnost problema USAHLP, što znači da je CSAHLP kao širi problem takođe NP-kompletan.

Ernst i Krishnamoorthy u [Ern99] predlažu novu mešovitu celobrojnu LP formulaciju za CSAHLP, u kojoj funkcija cilja obuhvata i fiksne troškove uspostavljanja habova. Nova formulacija ima manje promenljivih i uslova u odnosu na prethodne ([OK92] i [Cam94a]), što je čini pogodnijom za rešavanje problema većih dimenzija. Autori takođe opisuju heuristike simuliranog kaljenja (simulated annealing-SA) i promenljivog spusta (randomized descent heuristic RDH) za rešavanje CSAHLP. Nova LP formulacija problema omogućava algoritmu CPLEX 4.0 da optimalno reši AP instance problema sa n≤50 čvorova. Predložene heuristike u relativno kratkom vremenu izvršavanja dostižu optimalna rešenja na AP instancama dimenzija n≤50, sem instance 50TT. Obe heuristike daju rešenja i za AP instance većih dimenzija sa n=100 i n=200 čvorova. Poređenja pokazuju da RDH metoda za nešto kraće vreme rešava probleme manjih dimenzija, dok SA heuristika ima malo bolje performanse pri rešavanju problema realnih dimenzija u pogledu i kvaliteta rešenja i vremena izvršavanja.

U [Lab03] autori proučavaju osobine hab lokacijskih problema ograničenih/neograničenih kapaciteta sa jednostrukim alokacijama i na osnovu dobijenih rezultata predlažu egzaktnu metodu grananja i sečenja (branch-and-cut) za njihovo rešavanje. Algoritam najpre izvršava proceduru preprocesiranja i "ojačavanja" koja proverava dopustivost problema i dodaje nove uslove koristeći dobijene donje granice za količinu protoka i broj habova koje treba locirati. Procedura za svaki par čvorova proverava da li se mogu pridružiti istom habu i uklanja čvorove koji se ne mogu pridružiti nijednom čvoru. Branch-and-cut algoritam za početni čvor uzima rešenje LP relaksacije problema koje je eventualno popravljeno nekom heuristikom zaokruživanja. Algoritam koristi dve strategije grananja, tri strategije izračunavanja i nekoliko nejednakosti za "sečenje" branch-and-cut drveta. Predložena egzaktna metoda je primenjena za rešavanje hab lokacijskog problema sa jednostrukim alokacijama sa ograničenom količinom protoka koji prolazi kroz svaki hab čvor (u radu [Lab03] označen sa Quadratic Capacitated Hub Location Problem-QHL), pošto ovaj problem predstavlja uopštenje problema CSAHLP i USAHLP (ili Linear Capacitated/Uncapacitated Hub Location Problem -LHL/UHL u [Lab03]). Testiranja su izvršena na skupu instanci sa n=17 i n=20 čvorova, koje su formirane na osnovu podataka dobijenih iz francuske telefonske mreže France Telecom i na standardnim AP instancama sa n≤50 čvorova. Na prvom skupu instanci, branch-and-cut metoda daje optimalna rešenja, sem u tri slučaja kada algoritam nije uspeo da se izvrši do kraja zbog vremenskog ili memorijskog

ograničenja. Na skupu AP instanci algoritam je optimalno rešio problem za n=10,20,25 čvorova, kao i za instance sa n=40 i n=50 čvorova koje imaju "lakša" ograničenja kapaciteta. Za ostale AP instance (14 od ukupno 48) branch-and-cut metoda nije dala rešenja zbog nedostatka memorije ili nemogućnosti algoritma da pronađe dopustivo početno rešenje (u 2 od 14 slučajeva). Poređenja na instancama manjih dimenzija pokazuju da je predložena branch-and-cut metoda efikasnija od algoritma CPLEX 7.0.

U [Yam03] razmatra se varijanta QHL problema pod nazivom Quadratic Hub Location Problem With Integer Links (QHLI) koji ima primenu u telekomunikacijskim sistemima i takođe je NP-kompletan [Yam02]. Problem se odnosi na mreže sa specifičnom arhitekturom (tributary/backbone networks), u kojima je mreža habova (backbone network) potpuno povezana, dok mreža ne-hab čvorova (tributary network) ima zvezdastu strukturu, tj. svaki ne-hab čvor je direktno povezan sa tačno jednim habom [Kli98]. QHLI model takođe podrazumeva šemu jednostruke alokacije ograničenja količine protoka koji prolazi kroz hab čvorove, ali razlika u odnosu na QHL model je u tome što cena transporta duž svakog luka u mreži deo po deo linearna funkcija. Pored fiksnih troškova uspostavljanja habova, funkcija cilja QHLI problema uključuje i fiksne troškove izgradnje lukova odgovarajućih kapaciteta. Autori predlažu dvofaznu metaheuristiku za rešavanje QHLI problema, koja se sastoji od heuristike tabu pretraživanja (lokacija habova) i gramzivog algoritma (alokacija ne-hab čvorova lociranim habovima). Metaheuristika pored dobijenog najboljeg rešenja daje i skup "kandidata", odnosno lokacija habova koji se najčešće javljaju u nekoliko najboljih rešenja problema. Najbolje rešenje dobijeno metaheuristikom koristi se kao početna gornja granica branch-and-cut metode koja bira lokacije habova samo iz skupa "kandidata".Opisana hibridna metoda je testirana na instancama dobijenih iz prakse sa n=12,17,49 čvorova i modifikovanim AP instacama sa $n \leq 50$ čvorova. Rezultati pokazuju da su na instancama manjih dimenzija uglavnom dostignuta optimalna rešenja, osim na instancama sa "teškim" ograničenjima kapaciteta. U slučaju problema većih dimenzija, odstupanje (agap) od optimalnog rešenja je značajno. Sličan pristup rešavanju modifikacije QHLI problema koja uključuje promenljive kapacitete grana u backbone/tributary mreži opisan je u [Yam04].

Dvofazna heuristika lokalnog pretraživanja za rešavanje CSAHLP u tributary/backbone mreži opisana je u radu [Car04]. U prvoj fazi primenjuje se jedna od tri predložene heuristike tabu pretraživanje (tabu search), slučajni ponovljeni start (random multistart), iterativno lokano pretraživanje (iterated local search) za nalaženje što boljeg skupa habova. Alokacije korisnika/snabdevača uspostavljenim habovima određuju se u drugoj fazi korišćenjem gramzive heuristike lokalnog pretraživanja (greedy local search). U post-optimizacijskoj fazi primenjuje se lokalno pretraživanje u cilju dodatnog popravljanja rešenja. Testiranja i poređenja na 19 instanci problema sa $n \leq 49$ čvorova pokazuju da je kombinacija tabu i gramzive heuristike pretraživanja najefikasnija, ali da vreme izvršavanja značajno raste sa povećanjem dimenzije problema.

5.3 Predložena GA implementacija

5.1.1 Reprezentacija jedinki i računanje funkcije cilja

Genetski kod jedinke se sastoji od *n* gena, gde svaki od gena ukazuje na jedan čvor mreže. Prvi bit u svakom genu uzima vrednost 1 ako taj čvor jeste hab, inače uzima vrednost 0. Razmotrivši vrednosti ovih bitova, lako se formira niz uspostavljenih habova. Ostali bitovi u genu referišu na hab koji je dodeljen tekućem čvoru i pri tom je svaki hab čvor je alociran samom sebi. Kako kod ovakve reprezentacije ne može doći do dupliranja indeksa habova, a takođe broj uspostavljenih habova nije fiksiran, očigledno da sa tih aspekata nemamo problem pojavljivanja nekorektnih jedinki kao u prethodnim slučajevima.

Za svaki konkretni ne-hab čvor se kreira niz uspostavljenih habova sortiran u neopadajućem poretku po rastojanjima do datog ne-hab čvora. U slučajevima kada je $\alpha < \chi$ i $\alpha < \delta$ optimalno rešenje obično ne uključuje alokaciju svakog ne-hab čvora svom najbližem habu. Indeksi habova koji su bliži ne-hab čvorovima se često pojavljuju u optimalnom rešenju, a indeksi udaljenih habova se retko javljaju. Zbog toga se pretraga usmerava prema „bližim“ habovima, dok se „udaljeni“ habovi razmatraju sa malom verovatnoćom. Sortiranje niza habova u neopadajući poredak po rastojanju od ne-hab čvora obezbeđuje se da bliži habovi imaju viši prioritet u dodeli (uređenje "najbližeg suseda").

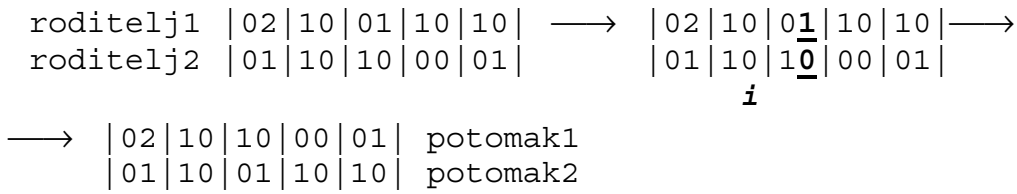
Vrednost funkcije cilja se računa na sledeći način: kada se fiksira skup habova Z_k, odrede se najkraći putevi korićenjem modifikacije dobro poznatog Flojd-Varšalovog algoritma. Potom se vrednost funkcije cilja dobija tako što se sumiraju najkraća rastojanja pomnožena sa koeficijentima toka χ, α i $\delta\,\alpha$ i na tu sumu se dodaju fiksni troškovi F_k za svaki uspostavljeni hab (tamo gde je $Z_{kk} = 1$).

U slučaju da je neki ne-hab čvor pridružen habu čiji (preostali) kapacitet nije dovoljan da bi zadovoljio potrebe datog čvora, iz sortiranog niza uspostavljenih habova za dati ne-hab čvor uzimamo prvi sledeći hab, koji ima dovoljno kapaciteta da opsluži dati ne-hab čvor. Ako takav hab ne postoji, datu jedinku smatramo nekorektnom i uklanjamo je iz populacije postavljajući njenu funkciju prilagođenosti na nulu.

5.1.2 Genetski operatori

Što se genetskih operatora tiče, za selekciju se koristi FGTS sa vrednošću parametra selekcionog metoda 5.4 (videti sekciju 2.1.1.1).

Po izvršenju selekcije, primenjuje se modifikovani operator ukrštanja, koji proizvodi dva potomka. Operator na slučajan način bira poziciju bita *i* u genetskom kodu koja predstavlja tačku ukrštanja, a zatim se zamenjuju čitavi geni roditelja počevši od gena koji sadrži tačku ukrštanja (slika 5.2). Verovatnoća ukrštanja je 0.85, što znači da će oko 85% jedinki učestvovati u proizvodnji potomaka, dok u oko 15% slučajeva neće biti ukrštanja i potomci će biti identični roditeljima.



Slika 5.2 Modifikovani operator ukrštanja za n=5

Operator mutacije je dizajniran kao dvonivoovski operator sa zamrznutim genima. U svakom od gena, osnovni nivo mutacije je:

- 0.4/*n* za bit na prvoj poziciji;
- 0.1/*n* za bit na drugoj poziciji. Sledeći bitovi u genu imaju svaki put dvostruko smanjen nivo mutaije (0.05/*n* , 0.025/*n* , 0.0125/*n* ,...)

U poređenju sa normalnim nivoom mutacije, zamrzuti bitovi mutiraju sa većim nivoom:

- 2.5 puta višim nivoom (1.0/*n* umesto 0.4/*n*) ako se radi o bitu na prvoj poziciji u genu;
- 1.5 puta višim nivoom, inače.

5.1.3 Ostali aspekti GA implementacije

Svaka jedinka inicijalne populacije se generiše korišćenjem sledeće strategije:

- Prvi bit u svakom genu se postavlja pseudoslučajno i dobija vrednost 1 sa verovatnoćom *5*/*n* ;
- S obzirom da, pri alociranju habova ne-nab čvorovima, treba favorizovati „bliže“ habove, poželjno je da preostali deo gena sadrži mnogo nula. Zato se verovatnoća generisanja bitova sa vrednošću 1 postavlja na 2.5/*n* za drugi bit u svakom genu, dok sledeći bitovi uzimaju vrednost 1 svaki put sa dvostruko manjom verovatnoćom u odnosu na svog neposrednog prethodnika (1.25/n, 0.625/n,...)

Veličina populacije je takođe 150 jedinki. Pri izvršavanju se koristi stacionarna verzija algoritma, kao i elitna strategija na isti način kao kod GA implementacija opisanih u prethodnim poglavljima. U svakoj generaciji predloženog GA, duplikati se uklanjaju iz populacije postavljanjem njihove funkcije prilagođenosti na nulu. Jedinke koje imaju istu prilagođenost, ali različite genetske kodove u nekim slučajevima mogu dominirati populacijom. I ovde se, radi izbegavanja takvih situacija, ograničava (na N_{rv}=40) broj jedinki koje imaju istu prilagođenost, a različite genetske kodove.

Kodove GA implementacije, izvršavanje staje posle 5000 generacija (ako se rešavaju veće instance) ili posle 500 generacija (ako se rešavaju male instance), odnosno, ako se prilagođenost najbolje jednike nije popravila tokom 2000 generacija (kod većih instanci) ili posle 200 generacija (kod manjih instanci). Performanse GA metode se i ovde poboljšane keširanjem sa veličinom keš memorije N_{cache}= 5000.

5.2 Eksperimentalni rezultati

Predložena GA metoda je testirana na skupu AP instanci ([Bea96]) sa n≤200 čvorova koje uključuju dva tipa fiksnih troškova i ograničenja kapaciteta na habovima: teške (T) i lakše (L). Za svaku dimenziju problema n, izvedena su četiri tipa AP instanci problema: LL, LT, TL i TT. Parametri koji se odnose na transportne troškove po jedinici količine robe imaju vrednosti $\chi=3$, $\alpha=0.25$ i $\delta=2$.

Rezultati GA koji se prezentuju u ovom poglavlju dobijeni su u eksperimentima na AMD Athlon K7/1.33GHz računaru, sa 256 MB unutrašnje memorije. Dobijeni rezultati GA poređeni su sa rezultatima egzaktne BnB metode iz [Ern99] (testirana na DEC 3000/700, 200MHz) i heuristika promenljivog spusta (randomized descent heuristic RDH) i simuliranog kaljenja (simulated annealing-SA) predloženih u [Ern99] (takođe testirane na DEC 3000/700, 200MHz).

Kolone u Tabelama 5.1 i 5.2 sadrže sledeće podatke (po redosledu pojave sleva udesno):

- Dimenziju AP instance zajedno sa "težinama" ograničenja kapaciteta i fiksnih troškova respektivno. Prvi sufiks se odnosi na fiksne troškove uspostavljanja habova (L označava lake a T teške fiksne troškove), a sledeći sufiks ukazuje na ograničenja kapaciteta (L označava lakša, a T teža ograničenja kapaciteta habova). Na primer, instanca 40LT ima n=40 čvorova, lake fiksne troškove i teža ograničenja kapaciteta habova;
- Optimalno rešenje za tekuću instancu (Opt. reš) dobijeno BnB metodom iz [Ern99]. Izuzetak je instanca 50TT za koju je dato samo najbolje BnB rešenje, jer BnB metoda nije pronašla optimalno rešenje u ovom slučaju;
- Najbolje GA rešenje, pri čemu „opt“ označava da je GA dostigao optimum.
- Prosečno vreme t (u sekundama) potrebno da GA dostigne rešenje;
- Prosečno ukupno vreme t_{tot} (u sekundama) potrebno za završetak GA.
- Prosečan ukupan broj generacija (*gen*);
- Prosečno odstupanje rešenja dobijenog preko GA od optimalnog *agap* (u procentima);
- Standardna devijacija odstupanja od optimuma σ;
- Prosečan broj izračunavanja funkcije cilja *eval*;
- Kolona *keš* prikazuje uštedu vremena (u procentima) ostvarenu primenom tehnike keširanja

Kao što se može videti u tabelama 5.1 i 5.2, predloženi GA brzo dostiže sva poznata optimalna rešenja na instancama sa n≤50 čvorova za t ≤1.875 sekundi. Izuzetak je instanca 40TT na kojoj rešenje GA prosečno odstupa od optimalnog za 4.249%. U slučaju instance 50TT, za koju optimalno rešenje nije do sada poznato, odstupanje rešenja GA od najboljeg rešenja dobijenog BnB metodom iznosi 2.793% u proseku. Za instance velikih dimenzija n≤200, za koje optimum nije poznat, GA dovodi do rešenja za t ≤195.174 sekundi. Kako se pomoću GA ne može dokazati optimalnost dobijenog rešenja, to ne postoji adekvatan kriterijum završetka, pa GA i po dobijanju rešenja nastavlja da se izvršava dodatnih $\mathbf{t_{tot}-t}$ sekundi, sve dok ne bude ispunjen kriterijum završetka

Tabela 5.1. Rezultati GA na instancama manjih dimenzija

Inst.	Opt.reš. BnB	GA							
		Najb.reš.GA	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
10LL	224250.055	opt	0.019	0.940	2032	0.128	0.574	35547.6	65.1
10LT	250992.262	opt	0.064	1.105	2125	0.000	0.000	49641.4	53.3
10TL	263399.943	opt	0.127	1.067	2255	0.120	0.446	47350.6	58.3
10TT	263399.943	opt	0.028	0.967	2051	0.046	0.137	45390.3	55.8
20LL	234690.963	opt	0.043	2.169	2035	0.000	0.000	66000.7	35.3
20LT	253517.395	opt	0.456	2.406	2461	0.074	0.227	75465.8	37.9
20TL	271128.176	opt	0.451	2.565	2438	0.000	0.000	78903.6	35.6
20TT	296035.402	opt	0.211	2.160	2205	0.297	0.415	53469.3	51.5
25LL	238977.95	opt	0.402	3.066	2288	0.779	1.221	76972.7	32.9
25LT	276372.5	opt	0.127	2.969	2077	0.980	0.562	67922.3	34.8
25TL	310317.64	opt	0.251	3.077	2174	2.247	2.300	72906	33
25TT	348369.15	opt	0.548	3.003	2425	0.845	0.526	64927.9	46.5
40LL	241955.71	opt	0.226	5.266	2082	0.082	0.214	73973.1	29.1
40LT	272218.32	opt	1.15	6.463	2394	3.538	2.686	76834.2	35.9
40TL	298919.01	opt	0.399	5.619	2146	0.000	0.000	78009	27.5
40TT	354874.10	356509.86	0.842	5.750	2321	4.249	3.113	71647.6	38.3
50LL	238520.59	opt	0.521	7.448	2139	0.267	0.652	76537.9	28.6
50LT	272897.49	opt	1.066	8.534	2269	0.518	1.009	80235.6	29.5
50TL	319015.77	opt	1.875	8.866	2526	0.720	0.934	92312.4	26.8
50TT	417440.99*	422794.56	4.348	10.981	3100	2.793	1.591	110612.1	28.7

Tabela 5.2 Rezultati GA na instancama velikih dimenzija

Inst.	GA							
	Najb.reš.GA	t[s]	t_{tot}[s]	gen	agap[%]	σ[%]	eval	keš[%]
100LL	246713.97	5.300	56.052	5520	1.071	0.914	193743.9	29.3
100LT	256207.52	29.020	81.965	7814	3.194	1.791	279284.7	27.5
100TL	364515.24	15.184	66.293	6484	0.773	2.158	246441.4	23.8
100TT	475156.75	24.164	75.963	7298	4.391	4.442	270017.9	25.8
200LL	241992.97	168.966	424.517	8295	0.698	1.323	340043	18
200LT	270202.25	142.640	410.405	7588	1.864	2.227	302818.1	20.2
200TL	273443.81	80.872	325.878	6621	3.63	2.96	246438	25.2
200TT	291830.66	195.174	427.568	8968	0.81	0.8	349444.5	22.1

U Tabeli 5.3 prikazano je poređenje rešenja dobijenih predloženom GA metodom i heuristikama RDH i SA (iz [Ern99]) na osam AP instanci velikih dimenzija. Druga kolona Tabele 5.3 sadrži bolje od rešenja RDH i SA heuristika za tekuću AP istancu. U trećoj koloni dato je odgovarajuće DEC 3000/700 (200MHz) vreme izvršavanja za koje je RDH/SA heuristika dostigla dato rešenje. Naredne dve kolone sadrže najbolje rešenje GA i odgovarajuće AMD (1.33GHz) vreme izvršavanja za tekuću AP instancu. U poslednjoj koloni je naznačeno koja od tri heuristike je dostigla najbolje rešenje za svaku od osam AP instanci. Kao što se može videti iz Tabele 5.3, predložena GA metoda u slučaju dve instance 100LT i 200TT daje bolja rešenja u odnosu na RDH i SA heuristiku. Na instancama 100TL i 100TT SA heuristika dostiže najbolja rešenja, a RDH heuristika je najbolja na instanci 200LT. Na preostale tri instance 100LL, 200LL i 200TL sve tri heuristike daju ista rešenja. Međutim, vreme izvršavanja GA je osetno duže. I pored toga, ovakav pristup ima potencijala pri rešavanju CSAHLP, i da bi se uz izvesne optimizacije programskog koda implementacije mogla dobiti uporediva vremena izvršavanja.

To se može videti iz činjenice da vreme izvršavanja GA umereno raste sa povećanjem dimenzije problema.

Tabela 5.3.Poređenje na instancama velikih dimenzija n=100 i n=200

Inst	Ernst RDH i SA heur		GA		Bolji
	Najb.reš.	Alpha 200 MHz (sec)	Najb.reš.	AMD 1.33 GHz (sec)	
100LL	246713.97	18.55	246713.966	56.052	isti
100LT	256638.38	24.71	256207.520	81.965	GA
100TL	362950.09	30.16	364515.243	66.293	SA
100TT	474680.32	34.83	475156.751	75.963	SA
200LL	241992.97	136.01	241992.973	424.517	isti
200LT	268894.41	437.21	270202.252	410.405	RDH
200TL	273443.81	195.28	273443.813	325.878	isti
200TT	292754.97	190.12	291830.665	427.568	GA

6 ZAKLJUČAK

U ovom radu opisane su GA implementacije koje posebno dizajnirane za rešavanje hab lokacijskih problema. Predloženi pristup je primenjen na četiri NP-teška problema iz ove oblasti koji nalaze veliku primenu u praksi: USApHMP, CSApHMP, CSApHCP i CSAHLP. Za svaki od rešavanih problema detaljno su opisani teoretski i praktični aspekti predložene implementacije, kao i pregled svih ostalih metoda za rešavanje datog problema. Takođe je izvršeno direktno poređenje dobijenih rezultata GA sa najboljim postojećim rezultatima iz literature.

Razmatrani hab lokacijski problemi su teški za rešavanje, što se posebno vidi iz činjenice da su čak i njihovi potproblemi sa fiksiranim habovima takođe NP-teški. To je jedan od razloga što postojeće egzaktne metode mogu rešiti samo instance problema relativno malih dimenzija. Iz istog razloga, postojeće heuristike, zasnovane na metodi lokalnog pretraživanja, uglavnom ne daju rešenja zadovoljavajućeg kvaliteta na instancama problema velikih dimenzija. Zbog toga je predloženi evolutivni pristup vrlo značajan, jer robustnost i adaptivnost predloženih genetskih operatora omogućuju vrlo uspešno rešavanje datih problema, čak i na instancama vrlo velikih dimenzija.

Opisane GA implementacije 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 hab lokacijski problem. Koristeći ovakvu strukturu implementacije, moguće je relativno lako rekonfigurisati i dopuniti sistem, tako da se može primeniti i na slične hab i druge lokacijske probleme.

Predloženi GA koncepti koriste dva nova načina kodiranja jedinki i odgovarajuće funkcije cilja koje popravljaju nekorektne jedinke u populaciji do korektnih. Pri računanju funkcije cilja primenjena je strategija "uređenja najbližeg suseda" kojom se postižu značajna poboljšanja kvaliteta GA rešenja. U skladu sa primenjenim načinom kodiranja i prirodom problema, dizajnirani su i implementirani modifikovani genetski operatori koji čuvaju korektnost jedinki, čime se isključuje pojava nekorektnih jedinki u svakoj generaciji. Primenjeni genetski operatori su pokazali najbolje rezultate pri rešavanju razmatranih hab problema: fino gradirana turnirska selekcija, modifikovani opreratori ukrštanja i dvo-nivoski operatori mutacije sa zaleđenim bitovima. Keširanje GA je u velikoj meri poboljšava performanse predloženih implementacija, ali ne utiče na ostale aspekte GA. Primenjene su i razne metode za sprečavanje preuranjene konvergencije, gubljenja raznovrsnosti genetskog materijala i spore

konvergencije genetskog algoritma. Predložene GA implementacije koriste elitističku strategiju koja omogućava direktno prenošenje vrednosti elitnih jedinki u narednu generaciju, bez ponovnog računanja njihovih funkcija cilja. U poglavlju 2. opisana je hibridizacija GA sa jednom modifikacijom heuristike lokalnog preetraživanja u cilju poboljšanja kvaliteta rešenja.

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

- paralelizacija opisanih genetskih algoritma i izvršavanje na paralelnim računarima sa većim brojem procesora,
- hibridizacija sa nekim drugim heuristikama i/ili egzaktnim metodama,
- modifikacija opisanog GA za rešavanje sličnih problema kombinatorne optimizacije.

6.1 Naučni doprinos rada

Najznačajniji novi rezultati u ovom radu su:

- linearna formulacija CSApHCP kao problema mešovitog celobrojnog programiranja koja do sada nije postojala u literaturi,
- dobijanje rešenja za CSApHCP koji do sada nije rešavan,
- primena novih načina kodiranja i funkcija cilja bez pojave nekorektnih jedinki, korišćenjem strategije "uređenje najbližeg suseda",
- dizajniranje novih genetskih operatora koji odgovaraju prirodi problema, primenjenim načinima kodiranja i čuvaju korektnost jedinki,
- implementacija keširanja za ubrzavanje heuristika lokalnog pretraživanja,
- korišćenje vrednosti potencijalnih rešenja dobijenih heuristikom u glavnom delu GA,
- implementacija keširanja za ubrzavanje heuristike zamene,
- dobijanje rešenja na instancama problema velikih dimenzija koje do sada nisu rešavane.

Dobijeni rezultati jasno pokazuju da su predložene evolutivne metode izuzetno uspešne pri rešavanju hab lokacijskih problema. Zbog svega gore navedenog, naučno istraživanje opisano u ovom radu daje doprinos oblastima kombinatorne optimizacije, lokacijskih problema i genetskih algoritama. Deo dobijenih rezultata je već objavljen u inostranim i domaćim časopisima, dok su ostali delovi u pripremi za objavljivanje.

LITERATURA

[AbH93] **Abdinnour-Helm S., Venkataramanan M.A.,** "Using Simulated Annealing to Solve the p-Hub Location Problem", *IRMIS Working paper 9304,* School of Business, Indiana University, Bloomington, Indiana (1993).

[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., Venkataramanan M.A.,** "Solution Approaches to Hub Location Problems", *Annals Of Operations Research,* Vol. 78, pp. 31-50 (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, California, pp. 86-91 (1989).

[Ayk94] **Aykin T.,** "Lagrangean Relaxation Based Approaches to Capacitated Hub-and-Spoke Network Design Problem", *European Journal of Operational Research,* Vol. 79, pp. 501-523 (1994).

[Ayk95] **Aykin T.,**"Network Policies for Hub-and-Spoke Systems With Application to the Air Transportation System", *Transportation Science,* Vol. 29(3), pp.201-221 (1995).

[Ayk96] **Aykin T.,** "On Modelling Scale Economies in Hub-and-Spoke Network Design", *Presented at the Fall Conference of INFORMS,* Atlanta (1996).

[Bea96] **Beasley J.E.**, "Obtaining Test Problems via Internet", *Journal of Global Optimization*, Vol. 8, pp. 429-433 (1996).

[Bä00a] **Bäck T., Fogel D. B., Michalewicz Z.**: "Basic Algorithms and Operators", in: *Evolutionary Computation 1*, Institute of Physics Publishing, Bristol-Philadelphia, (2000).

[Bä00b] **Bäck T., Fogel D. B., Michalewicz Z.**: "Advanced Algorithms and Operators", In: *Evolutionary Computation 2*, Institute of Physics Publishing, Bristol-Philadelphia, (2000).
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, California, pp. 2-8 (1993).

[BeJ95] **Beasley J.E.**, "Lagrangean Relaxation", In: *Modern Heuristic Techniques for Combinatorial Problems,* Reeves C.R. (ed), McGraw-Hill, pp. 243-303 (1995).

[Ber97] **Bertsimas D., Tsitsiklis, J. T.,** "Introduction to Linear Optimization", *Athena Scientific,* Belmont, USA, (1997).

[Bo04] **Bollapragada R., Camm J., Rao U.S., Wu J.,** "A Two-Phase Greedy Algorithm to Locate and Allocate Hubs for Fixed Wireless Broadband Acess", *Operations Reserach Letters,* (in press 2004).

[Bol04] **Boland N., Krishnamoorthy M., Ernst A., Ebery J.,** "Preprocessing and Cutting for Multiple Allocation Hub Location Problems", *European Journal of Operational Research,* Vol. 155, pp. 638-653 (2004). ftp://lumpi.informatik.uni-dortmund.de/pub/GA/papers/icga93.ps.gz

[Boz02] **Bozkaya B., Zhang J., Erkut E.,** "An Efficient Genetic Algorithm for the p-median problem", in Drezner Z. and Hamacher H., 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).

[Bry98] **Bryan, D. L.** "Extensions to the Hub Location Problems: Formulations and Numerical Examples", *Geographical Analysis,* Vol. 34 (4), pp.315-330 (1998).

[Bry99] **Bryan, D. L., O'Kelly M. E.** "Hub-and-Spoke Networks in Air Transportation: An Analytical Review", *Journal of Regional Science,* Vol. 39 (2), pp. 275-295 (1999).

[Ca05] **Campbell A.M., Lowe T.J., Zhang L.,** "The p-Hub Center Allocation Problem", *European Journal of Operational Research*, (accepted 2005).

[Cam94a] **Campbell, J. F.** "A Survey of Network Hub Location", *Studies in Locational Analysis,* Vol. 6, pp. 31-49 (1994).

[Cam94b] **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).

[Cam00a] **Campbell J.F., Ernst A. and Krishnamoorthy M.,** " Hub Arc Location Problems: Part I-Introduction and Results", *Working Paper* (2000).

[Cam00b] **Campbell J.F., Ernst A. and Krishnamoorthy M.,** " Hub Arc Location Problems: Part II-Formulations and Optimal Algorithms", *Working Paper* (2000).

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

[Cam03] **Campbell J.F., Stiehr G., Ernst A.T., Krishnamoorthy M.,** "Solving Hub Arc Location Problems on a Cluster of Workstations", *Parallel Computing,* Vol. 29, pp. 555-574 (2003).

[Cam05a] **Campbell J.F., Ernst A.T., Krishnamoorthy M.**, "Hub Arc Location Problems: Part I - Introduction and Results", *Management Science,* Vol. 51, pp. 1540-1555 (2005).

[Cam05b] **Campbell J.F., Ernst A.T., Krishnamoorthy M.**, "Hub Arc Location Problems: Part II - Formulations and Optimal Algorithms", *Management Science*, Vol. 51, pp.1556-1571 (2005).

[Car04] **Carello G., Della Croce F., Ghirardi M., Tadei R.,** "Solving the Hub Location Problem in Telecommunication Network Design: A Local Search Approach", *Networks,* Vol. 44(2), pp. 94-105 (2004).

[Col91] **Colorni A., Dorigo M., Maniezzo V.,** "Distributed Optimization by Ant Colonies", *Proceedings of the 1991 European Conference on Artifical Life,* Elsevier, pp. 134-142, (1991).

[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).

[Čer85] **Černý V.,** "Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm", *Journal of Optimization Theory and Applications,* Vol. 45, pp. 41-51, (1985).

[Dan54] **Dantzig G.B., Fulkerson D. R., Johnson S. M.,** "Solution of a large scale traveling salesman problem. *Operations Research*, Vol. 2, pp. 393–410 (1954).

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

[Dom03] **Domínguez E., Muñoz J., Mérida E.,** "A Recurent neural Network Model for the p-Hub Problem", *Proceedings of IWANN 2003., Lecture Notes in Computer Science,* Vol. 2687, pp. 734-741 (2003).

[Dor91] **Dorigo M., Colorni A., Maniezzo V.,** "Positive Feedback as a Search Strategy", *Technical Report, TR91-016,* Politecnico di Milano, (1991).

[Dor92] **Dorigo M.,** "Optimization, Learning and Natural Algorithms", *Phd Thesis*, Politecnico di Milano, (1991).

[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., Hamacher H.,** "Facility Theory: Applications and Theory", *Springer-Verlag, Berlin-Heidelberg* (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).

[DJo75] **De Jong K.E.**, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", *PhD thesis*, University of Michigan (1975).

[Dre02] **Drezner Z., Klamroth K., Schöbel A., Weslowsky G.O.,** "The Weber Problem" in Drezner Z. and Hamacher H., 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).

[Đu07] **Đurić B., Kratica J., Tošić D., Filipović V.,** "Solving the maximally balanced connected partition problem in graphs by using genetic algorithm", submitted to *Computing and Informatics* (2007).

[Eb00] **Ebery J.E., Krishnamoorthy M., Ernst A.T., Boland N.,** "The Capacitated Multiple Allocation Hub Location Problem: Formulations and Algorithms", *European Journal of Operational Research,* Vol. 120 (3), pp. 614-631 (2000).

[Ern96] **Ernst, A.T., Krishnamoorthy M.,** "Efficient Algorithms for the Uncapacitated Single Allocation p-Hub Median Problem", *Location Science,* Vol. 4(3), pp. 139-154 (1996).

[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 Operations 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).

[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).

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

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

[Gav92] **Gavish B.,** "Topological Design of Computer Communication Networks-the Overall Design Problem", *European Journal of Operational Research,* Vol. 58(2), pp. 148-172 (1992).

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

[Glo77] **Glover, F.,** "Heuristics for Intege Programming Using Surogate Constraints", *Decision Sciences,* Vol. 8 (1), pp. 156-166 (1977).

[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).

[Glo98] **Glover, F.,** " A Template for Scatter Search and Path Relinking", in: J.K. Hao, E. Lutton, E. Ronald, M. Shoenauer and D. Snyers eds.: "Artificial Evolution"*, Springer Lecture Notes in Computer Science,* Vol. 1363, pp. 3-51 (1998).

[Glo00] **Glover F., Laguna M., Marti R.,** "Fundamentals of Scatter Search and Path Relinking", *Control and Cybernetics,* Vol. 29(3), pp. 653-684 (2000).

[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)

[Han99] **Hansen P., Mladenović N.**, "An Introduction to Variable Neighborhood Search", In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Voss S., Martello S., Osman I.H., Roucairol C. (eds.), Kluwer Academic Publishers, pp. 433-458 (1999).

[Her97] **Hertz A., Taillard E., de Werra D.**, "Tabu search", In: *Local Search in Combinatorial* Optimization, Aarts E.H.L. and Lenstra J.K. (eds.), John Wiley & Sons Ltd., pp. 121-136 (1997).

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

[Hop82] **Hopfield J.J.,** "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences,* Vol. 79, pp. 2554-2558 (1982).

[Hop82] **Hopfield J.J.,** "Neurons With Graded Response Have Collective Computational Properties Like Those of Two State Neurons", *Proceedings of the National Academy of Sciences,* Vol. 81, pp. 3088-3092, (1984).

[Hop85] **Hopfield, J.J., Tank D.W.,** "Neural Computation of Decisions in Optimization Problems, *Biological Cybernetics,* Vol. 52, pp.141-152 (1985).

[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).

[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).

[Jün94] **Jünger M., Reinelt G. ,Thienel S.,** "Provably good solutions for the traveling salesman problem", *Zeitschrift für Operations Research*, Vol. 40, pp. 183–217 (1994).

[Jün96] **Jünger M., Mutzel P.,** " Maximum planar subgraphs and nice embeddings: Practical layout tools" *Algorithmica*, Vol. 16(1), pp. 33–59 (1996).

[Jün97] **Jünger M., Mutzel P.,** "2-Layer straightline crossing minimization: Performance of exact and heuristic algorithms", *Journal of Graph Algorithms and Applications (JGAA)*, Vol.1(1), pp.1–25 (1997).

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

[Kar69] **Kariv O., Hakimi S.L.,** "An Algorithmic Approach to Network Location Problems; Part 2. The p-Medians", *SIAM Journal on Applied Mathematics,* Vol. 37, pp. 539-560 (1969).

[Kara98] **Kara B.Y., Tansel B.C.,** "On the Allocation Phase of the p-Hub Location Problem", *Technical Report, Department of Industrial Engeneering, Bilkent University,* Bilkent 06533, Ankara, Turkey (1998).

[Kara99a] **Kara B.Y., Tansel B.C.,** "The Latest Arrival Hub Location Problem", *Technical Report, Department of Industrial Engeneering, Bilkent University,* Bilkent 06533, Ankara, Turkey (1999).

[Kara99b] **Kara B.Y., Tansel B.C.**, "On the Single-Assignment p-Hub Covering Problem", *Technical report, Department of Industrial Engineering, Bilkent University*, Bilkent 06533, Ankara, Turkey (1999).

[Kara00] **Kara B.Y., Tansel B.C.,**"On the Single Assigment p-Hub center Problem", *European Journal of Operational Research,* Vol.125 (3), pp. 648-655 (2000).

[Kid93] **Kido T., Kitano H., Nakanishi M.**, "A hybrid search for genetic algorithms: Combining genetic algorithms, tabu search, and simulated annealing", In: *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, Calif., pp. 614 (1993).

[Kim92] **Kim H.-J., Tcha D.-W.,** "Optimal Design of a Two-Level Hierarchical Network With Dual Homing Local Connections", *Computers and Industrial Engeneering ,* Vol. 22 (3), pp. 273-281 (1992).

[Kim95] **Kim H.-J., Tcha D.-W.,** "Optimal Design of a Two-Level Distributed Network With Tree-Star Configuration", *IIE Transactions,* Vol. 27, pp. 555-563 (1995).

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

[Kli91] **Klincewicz J.G.,** "Heuristics for the p-Hub Location Problem", *European Journal of Operational Research,* Vol. 53(1), pp. 25-37 (1991).

[Kli92] **Klincewicz J.G.,** "Avoiding Local Optima in the p-Hub Location Problem Using tabu Search And GRASP", *Annals of Operations Research,* Vol. 40, pp. 283-302 (1992).

[Kli96] **Klincewicz J.G.,** "A Dual Algorithm for the Uncapacitated Hub Location Problems", *Location Science,* Vol. 4, pp. 173-184 (1996).

[Kli98] **Klinccewicz J.G.,** "Hub Location in Backbone Tributary Network Design. A Review", *Location Science,* Vol. 6, pp. 307-335 (1998).

[Kli00] **Klincewicz J.G.,** "Enumeration and Search Procedures for a Hub Location Problem With Economies of Scale", *Technical Report* (2000).

[Kra96] **Kratica J., Radojević S., Filipović V., Šćepanović A.**, "Primena epsilon-transformacije u problemu pretrage drveta", Međunarodni naučno-razvojni simpozijum: Stvaralaštvo kao uslov privrednog razvoja - Nove tehnologije i tehnike u službi čoveka, Beograd (1996). http://alas.matf.bg.ac.yu/~kratica/ntt96.pdf

[Kra96a] **Kratica J., Filipović V., Šešum V., Tošić D.,** "Solving of the uncapacitated warehouse location problem using a simple genetic algorithm", *Proceedings of the XIV International Conference on Material handling and warehousing*, pp. 3.33-3.37, Belgrade (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).

[Kra01a] **Kratica J., Tošić D.**, "Introduction to Genetic Algorithms and Some Applications", *Proceedings of a Workshop on Computational Intelligence - Theory and Applications*, pp. 57-68, Niš, Yugoslavia, February 27, (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", *Computers and Informatics,* Vol. 22, pp. 1001-1012 (2005).

[Kra06a] **Kratica J., Stanimirović Z.**, "Solving the Uncapacitated Multiple Allocation p-Hub Center Problem by Genetic Algorithm", *Asia-Pacific Journal of Operational Research*, Vol 23. No. 4, pp 425- 437 (2006).

[Kra06b] **Kratica J., Stanimirović Z., Tošić D., Filipović V.,** "Two Genetic Algorithms for Solving the Uncapacitated Single Allocation p-Hub Median Problem", *European Journal of Operational Research,* Vol 182, pp 15-28 (2006).

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

[La60] **Land A. H., Doig A.G.,** "An automatic method of solving discrete programming problems", *Econometrica,* Vol. 28, pp. 497-520 (1960).

[Lab03] **Labeé M., Yaman H., Gourdin E.,** "A Branch and Cut Algorithm for Hub Location Problems With Single Assignment", *ISRO/OR Preprint 2003/05,* Université Libre de Bruxelles (2003). ftp://smg.ulb.ac.be/

[Le96] **Lee Y., Lim B.H., Park J.S.,** "A Hub Location Problem in Designing Digital Data Service Networks: Lagrangean Relaxation Approach", *Location Science,* Vol. 4(3), pp. 185-196 (1996).

[Lee93] **Lee C.-H., Ro H.-B., Tcha D.-W.,** "Topological Design of a Two-Level Network With Ring-Star Configuration", *Computers and Operations Research,* Vol. 20 (6), pp. 625-637 (1993).

[Lo98] **Love R.F., Moris J.G., Wesolowsky G.**, "Facility location: Models and methods", *Publication in Operations Research*, North-Holland, New York, Vol. 7, (1988).

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

[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).

[Lju01] **Ljubić I., Raidl, G.R.,** "An Evolutionary Algorithm with Stochastic Hill-Climbing for the Edge-Biconnectivity Augmentation Problem", *EvoWorkshops 2001*, pp. 20-29 (2001).

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

[Ma04] **Marin A.,** "Formulating and Solving Splittable Capacitated Multiple Allocation Problems", *Computers and Operations Research,* (2004. in press).

[Mar99] **Marianov V., Serra D., ReVelle C.**, "Location of Hubs in a Competitive Environment", *European Journal of Operational Research*, Vol. 114, pp. 363-371 (1999).

[May02] **Mayer G., Wagner B.,** "Hublocator: An Exact Solution Method for the Multiple Allocation Hub Location Problem*", Computers and Operations Research,* Vol.29 (6), pp. 717-739 (2002).

[Met53] **Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E.**, "Equation of state calculation by fast computing machines", Journal of Chemical Physics, Vol. 21, pp. 1087-1091 (1953).

[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).

[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).

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

[M**ü**h97] **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).

[Mut01] **Mutzel P.,** "An alternative method to crossing minimization on hierarchical graphs", *SIAM Journal on Optimization*, Vol. 11(4), pp.1065–1080 (2001).

[Nagy98] **Nagy G., Salhi S.,** "The Many-Too-Many Location Routing Problem", in: *TOP, Socied de Estadistica & Investigacion Operativa,* Vol 6(2), pp. 261-265 (1998).

[OK86] **O'Kelly, M. E.,** "The location of interacting hub facilities", *Transportation Science,* Vol. 20, pp. 92-106 (1986).

[OK87] **O'Kelly, M. E.,** " A quadratic integer program for the location of interacting hub facilities", *European Journal of Operational Research,* Vol. 32, pp. 393-404 (1987).

[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).

[OK92] **O'Kelly, M. E.,** "Hub Facility Location with Fixed Costs", *Papers in Regional Science. The Journal of the RSAI,* Vol. 77 (1), pp. 293-306 (1992).

[OK94] **O'Kelly M. E., Miller H.J.,** "The Hub Network Design Problem", *Journal of Transport Geography,*Vol. 2 (1), pp. 31-40 (1994).

[OK96] **O'Kelly M. E., 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).

[OK98a] **O'Kelly M. E.** "On the Allocation of a Subset of Nodes to a Mini-Hub in a Package Delivery Network", *Papers in Regional Science.The Papers of RSAI,* Vol. 77(1), pp. 77-99 (1998).

[OK98b] **O'Kelly M. E., Bryan D.,** "Hub Location With Flow Economies of Scale", *Transportation Research B,* Vol. 32(8), pp. 605-616 (1998).

[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).

[Osm96a] **Osman I.H., Kelly J.P.**, "Metaheuristics: Theory and Applications", Kluwer Academic Publisher, Norwell (1996).

[Osm96b] **Osman I.H., Laporte G.**, "Metaheuristic: A bibliography", *Annals of Operations Research,* Vol. 63, pp. 513-623 (1996).

[Pad90] **Padberg M., Rinaldi G.,** " An effcient algorithm for the minimum capacity cut problem", *Mathematical Programming*, Vol. 47, pp.19–36, (1990).

[Pad91] **Padberg M., Rinaldi G.,** "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems", *Siam Review*, Vol. 33, pp. 60–100 (1991).

[Pam01] **Pamuk P., Sepil C.,** "A Solution to the Hub Center Problem via a Single-Relocation Algorithm With Tabu Search", *IEE Transactions,* Vol. 23, pp. 399-411 (2001).

[Pem96] **Pemberton J.C., Zhang W.,** "Epsilon-transformation: exploiting phase transitions to solve combinatorial optimization problems", *Artificial Intelligence*, Vol. 81, pp. 297-325 (1996).

[Pér98] **Pérez- Pérez M., Almeida Rodriguez F., Moreno Vega J.M.,** "Fast Heuristics for the p-Hub Location Problem", *Presented in EWGLAX,* Murcia, Spain (1998).

[Pér00] **Pérez- Pérez M., Almeida Rodriguez F., Moreno Vega J.M.,** "Genetic Algorithm With Multistart for the p-Hub Median Problem", *Proceedings of EUROMICRO 98, IEEE Computer Society,* pp.702-707 (2000).

[Pér04] **Pérez- Pérez M., Almeida Rodriguez F., Moreno Vega J.M.,** "On the Use of the Path Relinking for the p-Hub Median Problem", *Lecture Notes in Computer Science,* Vol. 3004, pp. 155-164 (2004).

[Pér05] **Pérez- Pérez M., Almeida Rodriguez F., Moreno Vega J.M.,** "A Hybrid GRASP-Path Relinking Algorithm for the Capacitated p-Hub Median Problem", *Lecture Notes in Computer Science,* Vol. 3636, pp. 142-153 (2005).

[Pir98] **Pirkul H., Schilling D.A.,** "An Efficient Procedure for Designing Single Allocation Hub and Spoke Systems", *Management Science,* Vol. 44(12), pp. 235-242 (1998).

[Pod99] **Podnar, H., Skorin Kapov J.,** "Genetic Algorithm for Cost Minimization Apllied to Networks With Treshold Based Discounting", *Working Paper* (1999).

[Pod02] **Podnar H., Skorin Kapov J., Skorin Kapov D.**, "Network Cost Minimization Using Threshold-Based Discounting", *European Journal of Operational Research,* Vol. 137, pp. 371-386 (2002).

[Pod03] **Podnar H., Skorin-Kapov J.**, "Genetic Algorithm for Network Cost Minimization Using Threshold Based Discounting, *Journal of Applied Mathematics and Decision Sciences,* Vol. 7 pp. 207-228 (2003).

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

[Rib02] **Ribeiro C.C, Hansen P.,**"essays and Surveys in Metaheuristics", *Operations Research/Computer Science Interfaces Series;ORCS 15,* Kluwer Academic Publishers, Boston/Dordrecht/London, (2002).

[Sas97] **Sasaki M., Suzuki A., Drezner Z.,** "On the Selection of Relay Points in a Logistic System", *Asia-Pacific Journal of Operational Research,* Vol 14, pp.39-54 (1997).

[Sas99] **Sasaki M., Suzuki A., Drezner Z.,** "On the Selection of Hub Airports for an Airline Hub-and-Spoke", *Computers & Operations Research,* Vol 26, pp.1411-1422 (1999).

[Sch86] **Schrivjer A.,** "Theory of Linear and Integer Programming", John Wiley & Sons, Chichester, England (1986).

[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).

[Sko01] **Skorin-Kapov D.**, "On Cost Allocation in Hub-Like Networks", *Annals of Operations Research,* Vol.106, pp. 63–78 (2001).

[Sko05] **Skorin-Kapov D., Skorin-Kapov J.**, "Threshold Based Discounting Networks: The Cost Allocation Provided by the Nucleolus", *European Journal of Operational Research*, Vol. 166, pp. 154-159 (2005).

[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).

[Sm95] **Smith K., Krishnamoorthy M., Palaniswami M.,** "On the Location of Interacting Hub Facilities: Neural Versus Traditional Approaches", *procedings of the 13th National Australian Society of Operations Research Conference,* pp. 423-432 (1995).

[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).

[Sol05] **Solyali O.,** "Solving the Single Allocation p-Hub Center Problem Using Genetic Alogorithms", *35th International Conference on Computers and Industrial Engeneering",* (2005).

[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 problema diskretnih lokacijskih problema primenom genetskih algoritama", Magistarska teza, Matematički fakultet, Beograd (2004).

[Sta06a] **Stanimirović Z., Kratica J., Dugošija Đ.,** "Genetic Algorithms for Solving the Discrete Ordered Median Problem", rad je prihvaćen za štampu u *European Journal of Operational Research,* Vol 182, pp. 983-1001 (2006).

[Sta07] **Stanimirović Z.,** "An Efficient Genetic Algorithm for the Uncapacitated Multiple Allocation p-Hub Median problem", rad je poslat u *Control &Cybernetics* (2007).

[Sto96] **Storer R., Flanders S., Wu S.,** "problem Space Local Search forNumber Partitioning", *Annals of Operations Research,* Vol. 63, pp. 465-487 (1996).

[Su01] **Sung C.S., Jin H.W.,** "Dual Based Approach for a Hub Network Design Problem Under Non-Restictive Policy", *European Journal of Operational Research*, Vol. 132, pp.88-105 (2001).

[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).

[Tošć04] **Tošić D., Mladenović N., Kratica J., Filpović V.,** "Genetski algoritmi*",* Matematički institut SANU, Beograd (2004).

[Vss99] **Voss S., Martello S., Osman I.H., Roucairol C.**, "Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization", Kluwer Academic Publishers, Boston, MA (1999).

[Yam02] **Yaman H.,** "Concentrator Locator in Telecommunication Networks", Phd Thesis, Université Libre de Bruxelles, (2002).
ftp://smg.ulb.ac.be/

[Yam04] **Yaman H, Carello G.,** "Solving the Hub Location Problem With Modular Link Capacities", *Computers & Operations Research,* (in press 2004).

[Yam03] **Yaman H, Carello G.,** " Solving the Hub Location Problem with Integer Links", *Activity Report*, IEOR-2003-08*,* Department of Industrial Engeneering, Bilkent University, Turkey (2003).

[Yo98] **Yoon M.G., Current J.,** "A Dual Based Heuristic for Hub Network Design", *Working paper,* (1998)

[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

[Zha96] **Zhang W., Korf R.E.**, "A study of complexity transitions on the asymmetric traveling salesman problem", *Artificial Intelligence*, Vol. 81, pp. 223-239 (1996).

SADRŽAJ