UNIVERZITET U BEOGRADU
Matematički fakultet

Miroslav Marić

REŠAVANJE NEKIH NP – TEŠKIH HIJERARHIJSKO-LOKACIJSKIH PROBLEMA PRIMENOM GENETSKIH ALGORITAMA

Doktorska disertacija

B e o g r a d

2008.

Mentor:	**Prof. dr Dušan Tošić** Matematički fakultet u Beogradu
Komentor:	**dr Jozef Kratica** Viši naučni saradnik Matematički institut SANU
Članovi komisije:	**Prof. dr Žarko Mijajlović** Matematički fakultet u Beogradu
	Prof. dr Milan Tuba Matematički fakultet u Beogradu
	Doc. dr Vladimir Filpović Matematički fakultet u Beogradu

Datum odbrane: _____

Rešavanje nekih *NP*–teških hijerarhijsko–lokacijskih problema primenom genetskih algoritama

Rezime

Tema ovog rada je primena genetskih algoritama za rešavanje nekih hijerarhijsko-lokacijskih problema. Lokacijski problemi su veoma zastupljeni u literaturi, dok su njihove hijerarhijske varijante tek u ekspanziji.

U ovom radu opisani su različiti genetski algoritmi za rešavanje tri NP-teška hijerarhijsko-lokacijska problema: problem hijerarhijskog raspoređivanja radnika (Hierarchical Workforce Scheduling Problem – HWSP), problem hijerarhijskog pokrivanja korisnika (Hierarchical Covering Location Problem - HCLP) i lokacijski problem snabdevača neograničenog kapaciteta u više nivoa (Multi-Level Uncapacitated Facility Location Problem - MLUFLP). Navedeni hijerarhijsko-lokacijski problemi imaju široku primenu u praksi – mogu se koristiti za određivanje: lokacija snabdevača, lokacija bolnica i kliničkih centara, lokacija fakulteta i tehničkih škola, lokacija vatrogasnih i policijskih stanica, rasporeda radnika u firmama itd.

Pri rešavanju problema hijerarhijskog raspoređivanja radnika korišćeno je kodiranje realnim brojevima iz kojeg se primenom D'Ont-ovog sistema dobijaju odgovarajući rasporedi radnika. U skladu sa ovakvim načinom kodiranja razvijeni su i odgovarajući genetski operatori.

Za potrebe rešavanja problema hijerarhijskog pokrivanja korisnika razvijeni su posebno modifikovani operatori mutacije i ukrštanja, koji omogućavaju da se očuva specifičnost jedinki koje predstavljaju potencijalno rešenje. Naime, pri definisanju problema tačno je određen broj lokacija koje treba da budu uspostavljene na različitim nivoima, tako da operatori selekcije, ukrštanja i mutacije moraju da očuvaju taj broj.

Koncept dinamičkog programiranja je iskorišćen kao tehnika koja je bitno ubrzala izvršavanje genetskog algoritma za rešavanje problema lokacija snabdevača bez ograničenja u više nivoa.

Kod svih implementacija pažljivo su birani parametri i vršena testiranja radi njihovog poboljšanja. Takođe, analizirana je i složenost izračunavanja predstavljenih algoritama.

Svi implementirani algoritmi su testirani na javno dostupnim instancama, pri čemu su genetski algoritmi dostigli sva poznata rešenja, u zadovoljavajućem vremenu. Genetski algoritmi su testirani i na instancama čije optimalno rešenje do danas nije poznato i dali su rezultate za koje se može pretpostaviti da su kvalitetni. Dobijeni rezultati genetskih algoritama jasno ukazuju na značaj i potencijal ovakog pristupa pri rešavanju hijerarhijsko-lokacijskih problema.

Ključne reči: *Evolutivne metode, Hijerarhijsko-lokacijski problemi, Genetski algoritmi, Kombinatorna optimizacija.*

Solving some NP-hard hierarchical location problems by using genetic algorithms

Abstract

The subject of this paper is the application of genetic algorithms (GA) for solving some hierarchical-location problems. Location problem are well-studied in the literature, but the hierarchical variants of these problems are still in expansion.

In this paper, different genetic algorithms for solving three NP-hard hierarchical-location problems are proposed: the Hierarchical Workforce Scheduling Problem – HWSP, Hierarchical Covering Location Problem - HCLP and Multi-Level Uncapacitated Facility Location Problem - MLUFLP. Those problems can be used to model various real life problems such as: the design of a supply chain systems, the layout of hospitals and clinics, faculties and technical schools, fire, police and other emergency stations, different scheduling problems , etc.

For solving the Hierarchical Workforce Scheduling Problem, the encoding with real numbers is used. The corresponding work shedules are obtained from the genetic code by applying the well known D'Ont system. Regarding the applied encoding scheme, the adequate genetic operators are developped and implemented.

In the GA implementation for solving the Hierarchical Covering Location Problem, the modified genetic operators crossover and mutation are used in order to preserve the specific characteristics of the individuals representing potential solutions in the search space. Since the problem definition specifies the number of locations to be established on each level, the selection, crossover and mutation operators must preserve this number.

The implemented dynamic programming approach signifincatly improves GA performance for solving Multi-Level Uncapacitated Facility Location Problem.

In all three GA implementations, the GA parameters are carefullly determined by series of GA runs. In order to improve the GA performance, exaustive numerical experiments are carried out to find the best set of parameter values. The time-complexity of the proposed GA approaches is also analyzed.

The numerical experiments are carried out on the publicly-available data sets. For all three hierarchical-location problems, proposed genetic algorithms reach all previously known optimal solutions on tested instances. Genetic algorithms are also benchmarked on instances unsolved in the literature so far, and it may be assumed that the obtained results are of high quality. Computational results clearly indicate the significant role and potential of the genetic algorithm approach for solving hierarchical-location problems.

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

PREDGOVOR

Rad se sastoji od pet poglavlja. U uvodnom poglavlju su prezentovane osnovne informacije o lokacijskim, odnosno hijererhijsko-lokacijskim problemima, *NP*–teškim problemima, heuristikama, metaheuristikama i karakteristikama genetskih algoritama. Primena genetskih algoritama za rešavanje problema hijerarhijskog raspoređivanja radnika je opisana u drugom poglavlju. Treće poglavlje je posvećeno rešavanju problema hijerarhijskog pokrivanja korisnika. U četvrtom poglavlju je prikazana implementacija genetskog algoritma za rešavanje lokacijskog problema snabdevača neograničenih kapaciteta u više nivoa. Pregled naučnog doprinosa rada i dobijenih rezultata, sa zaključkom, je prikazan u poslednjem - šestom poglavlju.

Želeo bih da se zahvalim mentoru prof. dr Dušanu Tošiću i komentoru dr Jozefu Kratici na rukovođenju pri izradi ove disertacije. Dugujem im i posebnu zahvalnost jer su me upoznali sa detaljima genetskih algoritama i utrošili dosta vremena i energije kako bih postigao rezultate koji su i doveli do ove disertacije. Zahvaljujem se prof. dr Milanu Tubi na dugogodišnjoj saradnji i radu na prikazanim problemima. Takođe, zahvalio bih se članovima komisije prof. dr Žarku Mijailoviću i doc. dr Vladimiru Filipoviću na korisnim sugestijama koje su doprinele kvalitetu ovog rada. Zahvaljujem se koleginci doc. dr Zorici Stanimirović na stručnim savetima i pomoći u toku rada na genetskim algoritmima.

Zahvaljujem na pomoći i kolegama sa Projekta 144007 - "Matematički modeli i metode optimizacije sa primenama", Ministarstva za nauku i tehnološki razvoj Republike Srbije. Takođe, zahvaljujem se i kompaniji AMD koja mi je obezbedila opremu za istraživanje.

Naročito se zahvaljujem svojim roditeljima i priljateljima na razumevanju i ogromnoj podršci koja mi je puno značila u toku rada na ovoj disertaciji.

Kandidat

Beograd, 2008. mr Miroslav Marić

1 UVOD

1.1 Lokacijski i hijerarhijsko–lokacijski problemi

Lokacijski problemi predstavljaju posebnu klasu zadataka optimizacije, kod kojih se najčešće zahteva minimizacija rastojanja, ukupnog vremena putovanja ili nekog drugog parametra. Ovi problemi privlače veliko interesovanje i veoma često su predmet istraživanja. Glavni razlog za toliku popularnost su velike mogućnosti praktične primene u raznim oblastima, a posebno u snabdevanju, planiranju zaliha i računarskim mrežama. Lokacijski problemi se odnose na određivanje položaja (lokacija) objekata u prostoru u kojem se već nalaze drugi relevantni objekti. Objekti za čije se lociranje traži mesto obično su neka vrsta centara, koji pružaju usluge, pa se često nazivaju snabdevači, dok se korisnici usluga (objekti koji su već postavljeni) nazivaju klijenti ili korisnici.

Pri klasifikovanju lokacijskih problema relevantne su sledeće karakteristike:

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

Najčešća podela lokacijskih problema je na osnovu načina na koji se predstavljaju promenljive (topografije). Po tom kriterijumu lokacijski problemi se dele na diskretne i kontinualne. Diskretni lokacijski problemi su problemi kod kojih su mesta na koja se može postaviti objekat elementi konačnog skupa.

Nasuprot njima, kontinualni lokacijski problemi dopuštaju postavljanje objekata na bilo koju lokaciju u kontinualnom prostoru (u kojem se nalaze drugi objekti). Interesantni su i mrežni problemi, koji predstavljaju kombinaciju prethodno pomenutih diskretnih i kontinualnih problema. Oni mogu imati diskretna rešenja koja se nalaze u temenima grafa, ili kontinualna rešenja koja se nalaze na lukovima.

Kada se posmatra oblik funkcije cilja, najčešće se razmatraju dva tipa problema: *Min-Sum* i *Min-Max*. Kada su u pitanju *Min-Sum* problemi, funkcijom cilja se minimizuje težinski zbir svih rastojanja novih i postojećih objekata. Time se favorizuju "prosečni" klijenti, a zanemaruju udaljeni ili izolovani. Kod *Min – Max* problema, nove lokacije se pronalaze tako da minimizuju maksimalno rastojanje između postojećih i potrebnih objekata, čime se ravnopravno tretiraju svi klijenti. U poslednje vreme, česta primena diskretnih lokacijskih problema u praksi je dovela do razvijanja novih - fleksibilnijih lokacijskih modela koji dobro opisuju širu klasu lokacijskih problema. Posebno interesantna je generalizacija klasičnih diskretnih lokacijskih problema: p-medijane, p-centra i centdian lokacijskog problema i naziva se diskretno uređeni problem medijane (Discrete Ordered Median Problem – DOMP [Nic01, Sta04, Sta07a]). Problemi obuhvaćeni DOMP-om imaju različite funkcije cilja. Kod problema p-medijane potrebno je minimizovati sumu transportnih troškova između svih klijenata i odgovarajućih snabdevača, dok problem p-centra ima za cilj smanjenje maksimalne cene transporta između svih parova klijent-snabdevač. Funkcija cilja centdian lokacijskog problema je konveksna kombinacija funkcija cilja p-medijane i p-centra, koja treba istovremeno da minimizuje ukupne i maksimalne troškove transporta od svakog snabdevača do njemu pridruženog klijenta. DOMP se bitno razlikuje od ostalih lokacijskih problema jer nema tačno definisanu funkciju cilja, već se ona može podesiti pomoću vektora koeficijenata troškova. Za različite izbore vektora koeficijenata troškova dobijaju se različiti tipovi funkcije cilja koji odgovaraju određenim diskretnim lokacijskim problemima iz grupe problema koje DOMP uopštava. Više o formulaciji i rešavanju DOMP problema se može videti u [Sta04].

Lokacijski modeli se mogu klasifikovati i na osnovu broja objekata koje treba locirati. Kod nekih problema ovaj broj je unapred zadat (u Veberovom i lokacijsko-alokacijskom kontinualnom problemu, u diskretnim i mrežnim zadacima p-medijane i p-centra). Ovakvi problemi se u literaturi nazivaju endogeni. Sa druge strane, kod egzogenih problema je broj novih objekata nepoznata veličina i njena vrednost se dobija kao rezultat optimizacije. Tipični predstavnici egzogenih modela su prost lokacijski problem [Kra96, Kra98, Kra00, Kra01a, Han07] (simple-plant location ili uncapacitated facility location) i problem prekrivanja skupa [Bof97, Chr74, Gal00, Sch93].

Kapacitet lokacija može biti ograničen ili neograničen, pa se na osnovu toga i lokacijski problemi mogu podeliti na one sa ograničenim, odnosno neograničenim kapacitetom.

Lokacijski problemi su svaki za sebe specifični, te njihova forma – funkcija cilja, uslovi i promenljive zavise od problema koji se rešava. Samim tim ne postoji univerzalni lokacijski problem koji bi se mogao primeniti za rešavanje svih diskretnih lokacijskih problema. Detaljnije informacije o ovim problemima, njihovoj klasifikaciji i metodama njihovog rešavanja se mogu naći u preglednim radovima [Aik85, Dea85, Dre02, Fra83, Fra92, Klo04, Lov88, Mir90, ReV05], a neke specifične metode se mogu naći u [Abr95, Jai02, Shm97, Sny06].

Za razliku od lokacijskih problema, hijerarhijska verzija ovakvih problema nije toliko istraživana, pa samim tim i nije odgovarajuće zastupljena u literaturi. Hijerarhijsko – lokacijski problemi predstavljaju proširenje lokacijskih problema, koje obično podrazumeva da se objekti (snabdevači) postavljaju tako da se poštuje hijerarhija među njima. Detaljan pregled ovakvih problema se može naći u [Nar84, Sah07], a neki od rešavanih problema u [Edw01, Esp03, Gal96, God05, Pla06].

Hijerarhijsko – lokacijski problemi su, kao i originalni lokacijski problemi od kojih su nastali, obično NP – teški, pa se za njihovo rešavanje koriste razne heuristike i metaheuristike. Više o NP – kompletnosti, heuristikama i metaheuristikama može se videti u narednim odeljcima.

1.2 *NP*–teški problemi

Problemi za koje postoji mogućnost rešavanja na savremenim računarima se mogu podeliti na dve klase: algoritamski nerešive i algoritamski rešive.

Algoritamski nerešivi zadaci su uopšteni problemi koji se ne mogu rešiti na savremenim računarima koji rade na principima koje je definisao fon Nojman. Zbog te činjenice ni bitno poboljšanje karakteristika računara neće doprineti rešavanju ovakvih problema. U grupu algoritamski nerešivih problema, između ostalih, spadaju: Problem zaustavljanja Tjuringove mašine, Deseti Hilbertov problem, Problem ekvivalentnosti reči u asocijativnom računu itd.

Mnogo interesantniju klasu predstavljaju algoritamski rešivi problemi. Za rešavanje ovih problema postoje odgovarajući algoritmi. Međutim, ovi zadaci mogu biti različite složenosti. Složenost podrazumeva računarske resurse, odnosno procesorsko vreme i količinu memorijskog prostora, koji su potrebni za pronalaženje rešenja. Praktično, procesorsko vreme je kritični resurs, tako da u zavisnosti od potrebnog procesorskog vremena za njihovo rešavanje, algoritamski rešivi problemi se mogu podeliti u potklase.

Svaki problem određen je spiskom parametara koji predstavljaju ulazne podatke i uslovima koje mora da zadovoljava rešenje. Individualni zadatak predstavlja instancu problema sa konkretnim parametrima. Formalno, neki zadatak je algoritamski rešiv ako postoji algoritam koji se može primeniti na svaki njegov individualni zadatak. Medjutim, nije dovoljno da postoji algoritam, već je potrebno da taj algoritam bude efektivan, odnosno pri nalaženju rešenja ne sme zahtevati nerealne računarske resurse.

Svaki zadatak ima svoju dimenziju (veličinu, razmeru, obim), odnosno jedan ceo broj koji se određuje na osnovu vrednosti parametara zadatka. Na primer, za problem pretraživanja ili sortiranja niza to je broj elemenata niza, dok je kod problema trgovačkog putnika to broj gradova, itd. Ako neki problem ima više parametara onda se na pogodan način može izvršiti kodiranje parametara, tako da se za svaki individualni zadatak na jedinstven način odredi dimenzija.

Vreme složenosti nekog algoritma za rešavanje zadatka $t(n)$ je maksimalno vreme koje je potrebno algoritmu za nalaženje rešenja bilo kog individualnog zadatka koji ima dimenziju n. Najčešće je tačan zapis funkcije $t(n)$ komplikovan, pa se ne koristi takav zapis već neka funkcija koja ima jednostavniji oblik, ali pri tome i ograničava funkciju $t(n)$. Praktično, za merenje efektivnosti algoritma najčešće se koristi Landauov simbol O (veliko O).

Problem ima algoritam vremenske složenosti $O(g(n))$ ako postoji konstanta $C > 0$, tako da za zadatak dimenzije n vreme izvršavanja algoritma, u najgorem slučaju, ne prelazi $C \cdot g(n)$. Na primer, vremenska složenost pronalaženja neke vrednosti po neuređenom nizu od n elemenata je $O(n)$, složenost pronalaženja vrednosti po uređenom nizu je $O(\log n)$, složenost sortiranja niza sa n elemenata Quick-sort algoritmom je $O(n \log n)$ itd.

Vrednost eksponencijalnih funkcija, ali i funkcije $n!$ mnogo brže raste od vrednosti stepenih funkcija (polinoma), pa se zbog toga razlikuju problemi sa eksponencijalnom složenošću (kod kojih se vreme predstavlja eksponencijalnom funkcijom) i problemi sa polinomskom složenošću, kod kojih je vreme složenosti $O(n^k)$, gde je k neka konstanta. Detaljnija klasifikacija složenosti se može pronaći u [Ognj04a].

Problem odlučivosti je problem za koji se rešenje dobija u obliku DA ili NE odgovora. Jasno je da se svaki problem optimizacije može svesti na problem odlučivosti.

Problem odlučivosti pripada klasi P ako se može rešiti determinističkim algoritmom sa polinomskom složenošću. Deterministički algoritam se opisuje klasičnom Tjuringovom mašinom ili nekim njenim analogonom poput Postove mašine, Markovljevih algoritama itd. Ključna karakteristika determinističkih algoritama je da ista kombinacija ulaznih parametara uvek dovodi do istih rezultata.

Ispitivanjem, odnosno "pretragom" svih mogućih rešenja može se rešiti veoma puno zadataka diskretne matematike. Praktično, do rešenja se dolazi ispitivanjem da li svako potencijalno rešenje predstavlja i stvarno rešenje. Obično se ispitivanje jednog potencijalnog rešenja može obaviti polinomskim algoritmom. U skladu sa navedenim, može se definisati pojam *NP* problema, odnosno problema koji se mogu rešavati nedetirminističkim polinomskim algoritmima.

Problem pripada *NP* klasi ako se za neko izabrano potencijalno rešenje može determinističkim polinomskim algoritmom proveriti da li je i stvarno rešenje.

Ključna razlika između klasa *P* i *NP* je u tome što se problemi iz klase P mogu efektivno rešiti, dok se u slučaju problema iz klase NP može samo verifikovati rešenje u polinomskom vremenu izvršavanja. Takođe, jasno je da svaki problem iz klase *P* pripada i klasi *NP,* odnosno da je $P \subset NP$, ali pravo pitanje je da li je *P* prava potklasa klase *NP*. Ovo je jedno od ključnih pitanja savremene matematike na koje još uvek nema odgovora. Praktično, pitanje je da li postoji neki problem koji pripada klasi *NP,* a nije u klasi *P.* Postoji dosta rezultata koji su vezani za *NP* probleme, a koji pomažu da se razgraniči odnos klasa *P* i *NP.* Najvažniji se odnose na NP kompletnost i dokazuju slabiji rezultat, koji je najčešće oblika: "Ako su *P* i *NP* različite klase onda posmatrani problem *I* pripada *NP* \ *P*". Većina tih rezultata je dobijena svođenjem jednih problema na druge.

Problem *B* je svodljiv na problem *C* ako i samo ako postoji algoritam *A* kojim se svaki individualni problem *IB* (problema *B)* u polinomskom vremenu svodi na individualni problem *IC* (problema *C*).

Takođe, u literaturi je prisutna i definicija da je problem B svodljiv na problem C ako postoji algoritam polinomske složenosti za rešavanje problema B, u kome se, jednom ili više puta, izvršava algoritam za problem C, ali se jedno rešavanje problema C tretira kao operacija sa jediničnom cenom [Pap82].

Svodljivost u polinomskom vremenu je predstavljena u Lemi 1.1 koja se naziva lema o svodljivosti (dokaz se može pronaći u [Živ00]) i ona omogućava klasifikaciju problema.

Lema 1.1. Ako problem *B* pripada klasi *P* i problem A može da se u polinomskom vremenu svede na *B*, onda i problem *A* pripada klasi P. Takođe, ako problem *A* pripada klasi *NP* i problem *A* može da se u polinomskom vremenu svede na problem *B*, onda i problem B pripada klasi *NP*.

Lema o svodljivosti omogućava da se u klasi *NP* problema izdvoji jezgro najtežih problema koji se nazivaju *NP*-kompletni problemi. Smatra se da je problem *A, NP*-kompletan ako:

- pripada klasi *NP,*
- svaki drugi problem *B* iz *NP* je polinomski svodljiv na problem *A.*

Dalje, problem je *NP*-težak ako i samo ako postoji ekvivalentan problem odlučivosti koji je *NP*-kompletan.

Iz prethodnog sledi da ako za bar jedan *NP*-kompletan problem postoji deterministički algoritam sa polinomskom složenošću, onda bi on važio i za sve zadatke iz klase *NP*. Takođe, ako je bar jedan *NP*-kompletan problem teško-rešiv, onda to važi i za sve *NP*-kompletne probleme.

Jedan od prvih problema za koji je dokazano da je *NP*-kompletan je "Problem zadovoljivosti" u matematičkoj logici. Kuk je zasnovao teoriju *NP*-kompletnosti ([Coo71]) i pokazao da je problem zadovoljivosti *NP*-kompletan. Kasnije je za veliki broj problema pokazano da predstavljaju *NP*-kompletne probleme. Detaljan pregled ovih problema se može naći u [Cre97, Gar79].

Kada je dimenzija *NP*-teškog problema dovoljno velika, onda se takav problem ne može rešiti pretragom ni na savremenim računarima, sa najboljim karakteristikama. Iz tog razloga su pronađene razne približne metode (algoritmi) za rešavanje *NP*-teških problema. Međutim, ovi algoritmi ne pružaju garancije da će se uvek doći do optimalnog rešenja. Rešenja koja se dobijaju pomoću ovih metoda nazivaju se suboptimalna, a same metode se nazivaju heuristike. Dosta često se u praksi pomoću heuristika može dobiti optimalno rešenje, ali se ne može verifikovati njegova optimalnost. Najčešće je verifikacija optimalnosti rešenja višestruko složenija od samog dobijanja takvog rešenja. Heuristike su veoma popularne i često su izbor za rešavanje *NP* – teških problema, prvenstveno zbog njihove efikasnosti, ali i mogućnosti primene na velike klase problema. Naredni odeljak je posvećen heurističkim metodama.

1.3 Heurističke metode

Metode optimizacije se generalno mogu podiliti na egzaktne i heurističke (približne). Egzaktne metode su najpoželjnije. Međutim, kao što je opisano u prethodnom odeljku, one najčešće daju loše rezultate kada se primene na NP-teške probleme velikih dimenzija. Sa druge strane, rezultati dobijeni primenom heurističkih metoda su nepouzdaniji ali se ovim metodama mogu rešavati i ovakvi problemi. Heurističke metode se koriste i kada nije moguće definisati

matematički model za probleme koji nemaju “lepe” karakteristike, poput konveksnosti, diferencijabilnosti itd.

Formalno, može se reći da je heuristika tehnika kojom se traži dobro rešenje zadatka za relativno kratko vreme, bez mogućnosti garantovanja njegove dopustivosti i optimalnosti, često čak ni njegove bliskosti optimalnom rešenju. Za neke heuristike se može pronaći gornja granica relativne greške njenog rešenja u odnosu na optimalno rešenje. U nekim slučajevima je moguće eksperimentalnim putem zaključiti da je neka takva metoda bolja od svih do tada poznatih metoda.

Kod heuristika poseban akcenat je na relativno kratkom vremenu traženja rešenja, budući da je u većini slučajeva kada je obim *NP*-problema veliki, nemoguće naći egzaktno rešenje u razumnom vremenu.

Klasične heuristike su razvijane postupno. Pedesetih godina dvadesetog veka problemi su rešavani primenom “inteligentnih” i “efektivnih” heuristika. Šezdesetih godina formulisani su približni matematički modeli realnih problema i pokušano rešavanje tih problema egzaktnim metodama. Međutim, ovakav pristup nije davao željene rezultate za veliki broj problema. Sedamdesetih godina se uvodi teorija kompleksnosti, odnosno vremenske složenosti, pa se kombinatorni zadaci dele na klasu relativno lako rešivih i na klasu teško rešivih. Kako je za *NP*-teške probleme mala verovatnoća da postoji egzaktan i brz algoritam za njihovo rešavanje, osamdesetih godina, se ponovo eksploatišu klasične heuristike. Klasične heuristike se mogu podeliti u sedam grupa:

1. Konstruktivne metode. Ovakve metode postepeno grade rešenje maksimalno koristeći specifičnosti zadatka koji se rešava.

2. Iterativno poboljšavanje, odnosno metode lokalnog pretraživanja. Glavna ideja je da se krene od proizvoljnog početnog rešenja, koje se postepeno poboljšava prebrojavanjem njemu susednih. Ovaj proces se ponavlja sve dok u susedstvu postoji bolje rešenje od tekućeg.

3. Matematičko programiranje. Problem se formuliše kao zadatak matematičkog programiranja, a zatim se rešava približnim (ne egzaktnim) metodama.

4. Dekompozicione heuristike. Početni problem se dekomponuje na manje potprobleme, ali se njihovim (čak i egzaktnim) rešavanjem ne može garantovati optimalnost rešenja početnog problema.

5. Podela dopustivog skupa. Glavni princip ovakvih metoda je da se dopustivi skup podeli na više podoblasti pa se delimičnim prebrojavanjem rešenja u svakom od njih pronalazi najbolje. Pretpostavlja se da je dopustivi skup početnog zadatka konačan.

6. Restrikcija dopustivog skupa. Ideja je da se restrikcijom eliminišu određene podoblasti dopustivog skupa tako da se novi zadatak lakše rešava.

7. Relaksacija. Ovaj metod ima suprotan prisup u odnosu na restrikciju, odnosno dopustivi skup se proširuje. Na primer, ako se problem formuliše kao zadatak matematičkog programiranja, tada se brisanjem nekih uslova (jednačina ili nejednačina) iz skupa ograničenja problem "relaksira". Posledica toga je proširenje dopustivog skupa, ali i olakšavanje rešavanja novog problema.

Pri rešavanju nekog problema dozvoljeno je, a često i neophodno, kombinovanje različitih pristupa. Na primer, početno rešenje se može dobiti konstrukcijom a zatim se poboljšavati lokalnim pretraživanjem.

Klasične heuristike razvijane su u cilju rešavanja pojedinačnih problema i veoma su vezane za karakteristike tog problema. Poboljšanje karakteristika računara kao i načina čuvanja i struktuiranja podataka, dovelo je do reprogramiranja klasičnih heuristika. Na taj način nastale su nove klase heuristika - tzv. moderne heuristike, odnosno metaheuristike. Za razliku od klasičnih heuristika, metaheuristike sadrže pravila i načela koja se mogu primeniti pri rešavanju velikog broja praktičnih zadataka iz različitih oblasiti. Devedesetih godina dvadesetog veka metaheuristički pristup je preovladao klasični, tako da se od tada, kada su istraživanja u pitanju, porede različite metaheuristike ili različita metodološka načela u okviru iste metaheuristike. Kao što će detaljnije biti objašnjeno u narednom odeljku, svaka od metaheuristika je uopšten metod i može se uspešno primeniti na široku klasu problema.

1.4 Metaheuristike

Glavna karakteristika metaheurističkih metoda je da se one, uz određene modifikacije, mogu primenjivati na širok spektar problema kombinatorne optimizacije. Ove metode pretražuju skup dopustivih rešenja u cilju nalaženja što boljeg rešenja, pri čemu su dopušteni potezi poput kretanja ka lošijem rešenju od trenutnog, proširivanja skupa na kojem se traži rešenje

nedopustivim elementima, traženje rešenja kombinovanjem postojećih itd. Detaljan opis metaheuristika se može naći u [Glo03, Osm96a, Osm96b, Rib02].

Najpoznatije metaheurističke metode, pored genetskih aloritama, su:

- simulirano kaljenje (Simulated annealing),
- tabu pretraživanje (Tabu search),
- Lagranževa relaksacija (Lagrangian relaxation)
- metoda promenljivih okolina (Variable neighborhood search - *VNS*),
- programiranje sa ograničenjima (Constraint programming)
- i brojni metaheuristički algoritmi inspirisani prirodom.

Simulirano kaljenje je metaheuristika zasnovana na principima statističke termodinamike. Prilikom kaljenja čelika, da bi se postigao što bolji rezultat, primenjuje se postupak postepenog snižavanja temperature. Analogno tome, ova metoda u svakoj iteraciji na slučajan način generiše nekog suseda trenutne tačke pretraživanja. Pronađeni sused se sa određenom verovatnoćom (koja se kontrolisano menja tokom iteracija) prihvata, bez obzira da li se radi o poboljšanju ili pogoršanju vrednosti funkcije cilja. Pri tome, vodi se računa o tome da se u toku izvršavanja postepeno smanjuje prag prihvatanja lošijih rešenja. Ovakvim pristupom se omogućava eventualno izbegavanje konvergencije rešenja ka lokalnom minimumu i povećava mogućnost dobijanja kvalitetnog rešenja. Postupak se ponavlja sve dok se ne ispuni kriterijum zaustavljanja koji je najčešće: unapred definisan maksimalan broj iteracija, broj iteracija u kojem se rešenje nije poboljšalo itd. Više detalja o ovoj metodi može se naći u [Krp83, Sum02, Sum06].

Tabu pretraživanje se zasniva na principu lokalnog pretraživanja, a kao svoju najvažniju komponentu koristi adaptivnu memoriju. Adaptivna memorija sadrži podatke o prethodnim fazama pretraživanja i utiče na sledeće izbore u procesu. Preciznije, u svakoj iteraciji se čuva istorija prethodnog pretraživanja koja sadrži karakteristike nekih od prethodno generisanih rešenja. Istorija se čuva da se algoritam u procesu traženja ne bi vraćao na slučajeve (mesta) koji su već bili pretraženi. Formira se okolina trenutnog rešenja koja može da se proširuje ili sužava, u zavisnosti od podataka sadržanih u istoriji. Kako okolina može biti prevelika, ona se često redukuje na neki svoj podskup na kojem se vrši pretraživanje. Zatim se pronalazi rešenje koje je bolje od trenutnog, ali koje ne mora da bude bolje od ostalih rešenja iz okoline. Očigledno je da tabu pretraživanje dopušta i korake koji ne popravljaju vrednost funkcije cilja, pa se na taj način izbegava konvergencija ka lokalnim ekstremumima. Više o tabu pretraživanju i njegovim primenama se može videti u [Glo77, Glo90, Glo97, Her97, Mis05].

Metoda Lagranževe relaksacije, nekim od uslova zadatka, dodeljuje odgovarajuće faktore koji se nazivaju Lagranževi množioci. Time se oni uključuju u vrednosnu funkciju. Dobijeno optimalno rešenje ovako definisanog problema predstavlja donju granicu za ocenu rešenja polaznog problema. Ukoliko se pogodno izaberu uslovi koji se uključuju u vrednosnu funkciju, može se dobiti jednostavniji problem sa mnogo kraćim vremenom optimalnog rešavanja, koji ima rešenja približna polaznom problemu. Više o ovoj metodi se može naći u [Bea95, Fis04], a neke uspešne primene pri rešavanju NP-teških problema opisane su u [Bea88, Bea90, Bea93b, Gal93, Geo74, Gui88, Ong04, Tan06].

Metoda promenljivih okolina se zasniva na lokalnom pretraživanju, pri čemu se u svakoj iteraciji može vršiti prestruktuiranje okoline trenutnog rešenja. Osnovna ideja ove metode je sledeća: u trenutku kada se dođe do nekog lokalnog optimuma izvrši se slučajno pomeranje u tekućoj okolini do nekog rešenja (koje može biti i veoma loše), pa se iz tog rešenja lokalnim pretraživanjem pokuša pronalaženje nekog drugog lokalnog ekstremuma. Ovim pomeranjem, do rešenja koje se nalazi relativno daleko od trenutnog, postiže se sistematično pretraživanje prostora rešenja i sprečava konvergencija metode ka lošijem lokalnom ekstremumu. Takođe, u slučajevima kada pomeranje nije dovelo do boljeg rešenja, zadržavanje u trenutno najboljem rešenju smanjuje mogućnost nepotrebnog širenja pretraživanja na nove oblasti prostora dopustivih rešenja. Kriterijumi zaustavljanja ove metode su slični kao i kriterijumi zaustavljanja u slučaju tabu pretraživanja. Metoda promenljivih okolina ima brojne primene u rešavanju raznih problema optimizacije [Han99, Han01, Han07, Kov08, Mla95, Mla97].

Mravlji algoritmi su tipični predstavnici algoritama inspirisanih prirodom. Socijalni insekti su u stanju da organizuju proizvodnju, komunikacioni sistem, sistem za uzbujnjivanje, izvrše podelu rada itd. Kolonije ovakvih insekata su vrlo fleksibilne i prilagodljive promenama koje mogu nastati u okruženju. Biolozi su mnogo istraživali interakciju između insekata u koloniji. Dokazano je da lučenjem feromona mravi daju informacije ostalim članovima kolonije. Kod većine mravljih vrsta izvestan broj skauta napušta koloniju u potrazi za hranom. Skauti koji su bili uspešni ostavljaju trag feromona iza sebe. Feromon ukazuje ostalim mravima kojim putem treba da se kreću. Pojava sve većeg broja mrava na novootkrivenom putu znatno uvećava količinu feromona i samim tim šalje jači signal ostalim mravima koji tragaju za hranom. Analogno, pri formiranju heurističkog algoritma, koji je ispirisan ovom idejom, veštački mravi pretražuju dopustivi prostor i formiraju rute, a određeni mehanizmi (poput isparavanja feromona) sprečavaju prebrzu konvergenciju ovakvih procesa. Više o mravljim algoritmima, kao i drugim algoritmima ovog tipa i njihovim primenama može se naći u [Dor91, Dor92, Dor96, Dor04, Ken01].

1.5 Genetski algoritmi

Genetski algoritmi su zasnovani na Darvinovoj teoriji o postanku vrsta i prirodnoj evoluciji [Dar59] i Mendelovim zakonima [Bow89], koji su se pojavili krajem devetnaestog veka. Svoje utemeljenje pronalaze i u biologiji, naime poznato je da se svi živi organizmi sastoje od ćelija, a u svakoj ćeliji se nalazi isti skup hromozoma. Hromozomi sadrže DNK lance koji predstavljaju opis celokupnog organizma. Hromozome čine geni koji predstavljaju blokove DNK, odnosno svaki gen opisuje neku osobinu organizma. U toku reprodukcije odabrani roditelji rekombinuju (ukrštaju) gene i time se formiraju novi hromozomi. Takođe, mutacija dovodi do malih promena DNK elemenata. Kvalitet (prilagođenost) novonastalog organizma se meri njegovim uspehom u životu. Upravo, na ovim principima izgrađena je i teorija genetskih algoritama.

Smatra se da je idejni tvorac genetskih algoritama John Holland koji je knjigom "Adaptation in natural and artificial systems" [Hll75] postavio temelje ove metode. Više o konceptu genetskih algoritama se može videti u [Bäc00a, Bäc00b, Bea93a, Bea93b, Gol89, Mic96, Mit99, Müh97], a veoma su zastupljeni i u domaćoj literaturi [Čan96, Fil98, Kra00, Kra01b, Toš04].

Populacija jedinki je osnovna konstrukcija kod genetskih algoritama. Broj jedinki u praktičnim primenama je najčešće između 5 i 500, a svaka jedinka predstavlja moguće rešenje u pretraživačkom prostoru za dati problem. Jedinka se predstavlja genetskim kodom nad određenom konačnom azbukom. U praksi se najčešće koristi binarno kodiranje kod kojeg se genetski kod sastoji od niza bitova. Međutim, u nekim slučajevima je pogodno koristiti i azbuke veće kardinalnosti. Način kodiranja rešenja je veoma bitan za rešavanje problema, a neadekvatno kodiranje može dovesti do loših rezultata, bez obzira na ostalu strukturu algoritma.

Početna populacija se najčešće generiše na slučajan način, čime se doprinosi raznovrsnosti genetskog materijala. Ipak, u nekim slučajevima je do boljih rezultata dovelo generisanje cele (ili dela) početne populacije nekom pogodno izabranom heuristikom. Preduslov za korišćenje ovakve metode je da vreme izvršavanja date heuristike bude relativno kratko, kao i da što manje smanjuje raznovrsnost genetskog materijala.

Svakoj jedinki se pridružuje funkcija prilagođenosti koja ima zadatak da ocenjuje kvalitet jedinke. Genetski algoritam, uzastopnom primenom operatora selekcije, ukrštanja i mutacije, obezbeđuje da se iz generacije u generaciju poboljšava apsolutna prilagođenost svake jedinke u populaciji, a time i srednja prilagođenost celokupne populacije. Ovim mehanizmom se dobijaju sve bolja rešenja datog konkretnog problema.

Selekcija nagrađuje natprosečno prilagođene jedinke tako što one dobijaju veću šansu za reprodukciju pri formiranju nove generacije. Sa druge

strane, slabije prilagođenim jedinkama se smanjuju šanse za reprodukciju pa one postepeno "izumiru".

Operator ukrštanja vrši razmenu gena jedinki, čime doprinosi raznovrsnosti genetskog materijala. Ovim operatorom se daje mogućnost da, razmenom genetskog materijala, dobro prilagođene jedinke generišu još bolje prilagođene jedinke. Takođe, relativno slabije prilagođene jedinke sa nekim dobro prilagođenim genima dobijaju svoju šansu da rekombinacijom dobrih gena proizvedu dobro prilagođene jedinke. Ukrštanje se vrši sa unapred zadatom vrednošću verovatnoće ukrštanja. Ova vrednost određuje koliko jedinki učestvuje u ukrštanju proizvodeći nove jedinke, ali i koliko se jedinki prenosi u sledeću generaciju bez modifikacija.

Primena isključivo operatora selekcije i ukrštanja po pravilu dovodi do preuranjene konvergencije ka lošem lokalnom ekstremumu. Do toga dolazi drastičnim i uzastopnim gubljenjem genetskog materijala, zbog čega sve veći regioni pretraživačkog prostora postaju nedostupni. Mutacijom se vrši slučajna promena određenog gena, sa datom verovatnoćom p_{mut}, čime je moguće vraćanje izgubljenog genetskog materijala u populaciju. To znači da se primenom ovog operatora postepeno vraćaju izgubljeni regioni pretraživačkog prostora. Zbog toga operator mutacije predstavlja osnovni mehanizam za sprečavanje preuranjene konvergencije genetskog algoritma ka lokalnom ekstremumu.

Osnovni koraci genetskog algoritma su prikazani sledećim pseudokodom (Slika 1.1).

```
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 1.1 Osnovna varijanta genetskog algoritma

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

Prost genetski algoritam je najpribližniji hipotezi o gradivnim blokovima i teoremi o shemi [Hll75, Gol89], međutim, u praktičnoj primeni pokazuje nedostatke. Naime, pri teorijskim razmatranjima, veličina populacije se ne uzima u obzir jer se smatra da populacija ima beskonačno mnogo članova. Zbog toga se primenjuju tehnike iz teorije verovatnoće. Međutim, u realnim situacijama populacija sadrži konačno mnogo jedinki pa primenom genetskih operatora selekcije, ukrštanja i mutacije može doći do greške uzorkovanja. Ova greška može značajno da utiče na izvršavanje genetskog algoritma. U proseku, na velikom obimu uzorka broj potomaka svake jedinke približno je jednak očekivanom broju. Međutim, pojedinačno su moguća velika odstupanja. Ovakva odstupanja u nekoliko početnih generacija mogu presudno da utiču na kasnije performanse genetskog algoritma, prvenstveno na kvalitet dobijenog rešenja i brzinu konvergencije. To se najčešće manifestuje gubitkom genetskog materijala, preuranjenom i sporom konvergencijom.

Do pojave preuranjene konvergencije dolazi ako jedna ili više relativno dobrih (ali ne i optimalnih) jedinki postepeno preovladaju u populaciji pa zbog toga proces konvergira ka lokalnom ekstremumu. Mogućnosti genetskog algoritma za poboljšanje datog rešenja u nastavku izvršavanja su jako male. Selekcija i ukrštanje u populaciji sa istim jedinkama nemaju nikakav efekat pa teorijski jedino mutacija može da doprinese izlazu iz date situacije. Međutim, zbog potpuno uništenog genetskog materijala u praksi je mutacija često bez efekata. Ukoliko je nivo mutacije relativno mali, promene genetskog materijala su neznatne pa dominantne jedinke vrlo brzo eliminišu sve ostale jedinke iz populacije. U suprotnom, ukoliko je nivo mutacije relativno veliki, genetski algoritam se pretvara u slučajnu pretragu.

Do preuranjene konvergencije najčešće dolazi zbog primene proste rulet selekcije. Ako u populaciji postoji jedinka sa relativno velikom funkcijom prilagođenosti, ona će najverovatnije istisnuti sve ostale jedinke iz populacije. Pri tome, ostale jedinke (čak i one sa malom funkcijom prilagođenosti) sadrže raznovrsne gene od kojih su neki možda i bolji u poređenju sa genima favorizovane jedinke. Međutim, u procesu proste selekcije, zbog lošeg rasporeda dobrih gena po lošim jedinkama i konačne veličine populacije, dolazi do eliminacije jedinki, a samim tim i dobrih gena iz populacije. Takođe, dobri geni gube šansu da se, uz pomoć operatora ukrštanja i selekcije, rekombinuju u dobre jedinke i time poboljšaju rešenje.

Spora konvergencija je problem koji je suprotan preuranjenoj konvergenciji. Do nje obično dolazi u kasnijoj fazi izvršavanja prostog genetskog algoritma. Ukoliko nije dostignuto optimalno rešenje, a populacija je postala dovoljno slična, srednja prilagođenost svih jedinki u populaciji je velika, dok su razlike između najbolje jedinke i ostalih jedinki u populaciji male. Iz tog razloga ne postoji dovoljni gradijent u funkciji prilagođenosti, koji bi pomogao genetskom algoritmu da se približi optimalnoj vrednosti u dostižnom broju generacija.

1.5.1 Moderni koncepti genetskih algoritama

Zbog svojih nedostataka prost genetski algoritam se može uspešno primeniti samo na manji broj problema kombinatorne optimizacije. Za rešavanje složenijih problema koriste se specifične vrste kodiranja i odgovarajuće funkcije prilagođenosti, posebno dizajnirane verzije genetskih operatora selekcije, ukrštanja i mutacije, razne politike zamene generacija, kao i fiksna ili adaptivna promena parametara tokom izvršavanja genetskog algoritma.

1.5.1.1 Kodiranje i funkcija prilagođenosti

Kodiranje i funkcija cilja su najznačajniji faktori za uspešnu primenu genetskog algoritma pri rešavanju nekog problema. Oni direktno zavise od prirode problema koji se rešava, tako da se često događa da metode koje daju dobre rezultate u nekim primenama mogu dati vrlo loše rezultate kada se primene na druge probleme. Za kodiranje je idealno da funkcija prilagođenosti bude neprekidna i glatka pa da jedinke sa sličnim genetskim kodom imaju sličnu vrednost prilagođenosti. Takođe, dobar preduslov za dobre rezultate je funkcija prilagođenosti koja nema mnogo lokalnih ekstremuma ili suviše izolovan globalni ekstremum ([BeD93a]). U tako povoljnim slučajevima druge metode često daju približne ili čak i bolje rezultate, pa se genetski algoritmi prvenstveno primenjuju i u nepovoljnijim slučajevima, kada druge metode nije moguće primeniti ili one daju izuzetno loše rezultate. Funkcja cilja preslikava genetski kod jedinke u prostor pretrage na neki "prirodan" način. Međutim, često se dešava da takav način ne postoji ili iz nekog razloga nije pogodan.

Najčešće korišćeni načini računanja funkcije prilagođenosti su: direktno preuzimanje, linearno skaliranje, skaliranje u jedinični interval i sigma odsecanje. Izbor odgovarajućeg načina najviše zavisi od specifičnosti problema, a često je neophodno i kombinovanje različitih principa.

Najjednostavniji način računanja funkcije prilagođenosti je direktno preuzimanje kod kojeg se za vrednost funkcije prilagođenosti neke jedinke

uzima njena vrednost funkcije cilja. Međutim, ovaj pristup u praksi često daje slabe rezultate. Kada je linearno skaliranje u pitanju, prilagođenost neke jedinke se računa kao linearna funkcija vrednosti funkcije cilja te jedinke. Skaliranje u jedinični interval podrazumeva da funkcija prilagođenosti uzima vrednosti iz intervala [0, 1]. Najbolja jedinka ima prilagođenost čija je vrednost gornja granica intervala - tj. jedan, dok najslabije prilagođena jedinka ima prilagođenost čija je vrednost nula. Kada je sigma odsecanje u pitanju, funkcija prilagođenosti se računa po formuli (1.1).

$$f(x) = \begin{cases} 0, x < \bar{x} - C\sigma \\ x - (\bar{x} - C\sigma), x \geq \bar{x} - C\sigma \end{cases} \quad (1.1)$$

U formuli (1.1) oznaka $\bar{x}$ predstavlja srednju vrednost funkcije cilja, dok simbol σ predstavlja standardnu devijaciju u populaciji.

Korišćenje Gray kodova, čija je osobina da se susedni genetski kodovi razlikuju samo u jednom bitu, je jedna od metoda koja doprinosi poboljšanju funkcije prilagođenosti, a time i poboljšanju performansi genetskih algoritama. Više o različitim načinima za računanje funkcije prilagođenosti jedinki se može naći u [Kra00].

Genetski algoritmi su se pokazali kao veoma uspešni za rešavanje problema koji se mogu binarno kodirati na prirodan način. Binarno kodiranje svakom rešenju dodeljuje kod unapred zadate dužine nad binarnom azbukom. Takođe, u nekim slučajevima, kao veoma uspešna su se pokazala i kodiranja nad azbukama veće kardinalnosti [Ant89, BeD93b].

Za genetski algoritam je najbolje da svaki genetski kod odgovara jednom rešenju problema. Međutim, često ova veza nije bijekcija a ponekad čak ni preslikavanje. U takvim situacijama primenom genetskih operatora se mogu dobiti nekorektne jedinke, odnosno jedinke čiji genetski kod ne pripada skupu dopustivih rešenja. Ovaj problem se trivijalno može rešiti dodeljivanjem nule za vrednost funkcije prilagođenosti svakoj nekorektnoj jedinki pa se primenom operatora selekcije eliminišu ovakve jedinke. U praksi se ovaj pristup pokazao dobrim samo u situacijama kada odnos broja nekorektnih i korektnih jedinki u populaciji nije preveliki, što najčešće nije slučaj. Jedno od rešenja ovog problema je i uključivanje nekorektnih jedinki u populaciju. Naime, svakoj nekorektnoj jedinki se dodeljuje vrednost kaznene funkcije, sa ciljem da i nekorektne jedinke dobiju šansu za učestvovanje u ukrštanju, ali i da budu diskriminisane u odnosu na korektne jedinke. Pri tome se mora voditi računa da se vrednosti kaznene funkcije prilagode, jer suviše male vrednosti mogu da dovedu do toga da genetski algoritam neki od nekorektnih kodova proglasi za rešenje. Takođe, prevelike kazne mogu da prouzrokuju gubljenje korisnih informacija iz nekorektnih jedinki pa je ključno pronaći idealan balans kada je

kažnjavanje u pitanju. Problem sa nekorektnim jedinkama se može rešiti popravkom tako da i one postanu korektne, odnosno da se svaka nekorektna jedinka zameni korektnom.

1.5.1.2 Tipovi operatora selekcije

Zadatak operatora selekcije je da izvrši odabir jedinki koje će učestvovati u postupku stvaranja nove generacije. Pri tome se vodi računa o tome da verovatnoća odabira neke jedinke mora da bude u skladu sa vrednošću njene funkcije prilagođenosti. Prosta rulet selekcija je najprostiji oblik operatora selekcije. Ona je definisana tako da verovatnoća odabira neke jedinke bude direktno proporcionalna vrednosti njene funkcije prilagođenosti. Uzorak na kojem se vrši rulet selekcija je često suviše mali da bi se selektovani kodovi pojavljivali u odnosu u kojem su nijhove funkcije prilagođenosti pa se dešava da dobre jedinke budu favorizovane više nego što je to poželjno. Zbog toga primena proste rulet selekcije obično dovodi do brzog gubitka genetskog materijala, a samim tim i preuranjene konvergencije.

Jedan od načina za prevazilaženje prethodno pomenutog nedostatka je korišćenje selekcije zasnovane na rangiranju genetskih kodova prema njihovoj prilagođenosti. Funkcija prilagođenosti jedinke se bira iz unapred zadatog niza rangova, tako da zavisi samo od pozicije jedinke u populaciji. Na taj način, u slučaju velikih razlika među vrednostima funkcije cilja jedinki u populaciji, neke lošije jedinke mogu da dobiju šansu.

Turnirska selekcija je jedan od popularnijih operatora selekcije. Turniri su takmičenja između jedinki populacije koje se nadmeću radi preživljavanja i učešća u sledećoj generaciji. Veličina turnira (N_{tur}) je broj jedinki koje učestvuju na turniru i ona predstavlja parametar turnirske selekcije. Ovaj parametar se najčešće unapred zadaje. Turnirska selekcija funkcioniše tako što se najpre na slučajan način biraju podskupovi od po N_{tur} jedinki, a zatim se u svakom podskupu bira najbolja jedinka koja učestvuje u stvaranju nove generacije. Ovaj postupak se ponavlja onoliko puta koliko je potrebno da se izaberu jedinke koje će učestvovati u ukrštanju. Jedno od mogućih unapređenja turnirske selekcije je Fino gradirana turnirska selekcija o kojoj će biti više reči u odeljku 2.2.2.

Detaljan opis ove i ostalih tipova selekcije može se naći u [Fil98, Fil06]. Poređenje fino gradirane selekcije sa drugim operatorima i ponašanje ove selekcije u praksi prikazano je u [Fil00, Fil01, Fil03, Fil06].

Naročito je interesantan operator uniformne selekcije po prilagođenosti [Hut02], jer on, za razliku od dotad poznatih operatora, omogućava očuvanje genetske raznovrsnosti, a ne povećava, po svaku cenu prosečnu prilagođenost celokupne populacije. Ovakav pristup je u koliziji sa hipotezom o gradivnim

blokovima i teoremom o shemama. Međutim, u praktičnim primenama on je pokazao veoma dobre rezultate, od kojih su neki prezentovani u [Leg04].

1.5.1.3 Operatori ukrštanja

Operatorom ukrštanja izvršava se razmena genetskog materijala jedinki koje su operatorom selekcije odbabrane da budu roditelji novim jedinkama (potomcima). Ukrštanje se najčešće postiže jednopozicionim, dvopozicionim, višepozicionim ili uniformnim ukrštanjem, ali postoje i složenije varijante ovog genetskog operatora [Müh97].

Kod jednopozicionog ukrštanja se na slučajan način biraju parovi roditelja iz populacije i broj koji predstavlja tačku ukrštanja, a koji je manji od dužine genetskog koda. Svi geni, počevši od pozicije tačke ukrštanja do poslednje pozicije u genetskim kodovima roditelja, uzajamno menjaju mesta stvarajući pri tom dva nova potomka. Operator dvopozicionog ukrštanja funkcioniše tako što se slučajno biraju dve tačke ukrštanja, a zatim razmenjuju delovi genetskih kodova roditelja između ovih pozicija.

Kod uniformnog ukrštanja se za svaki roditeljski par na slučajan način generiše "maska", odnosno binarni niz iste dužine kao genetski kod. Roditelji zatim razmenjuju gene na svim pozicijama na kojima maska ima vrednost 0, dok na mestima gde maska uzima vrednost 1 roditelji zadržavaju svoje gene. Detaljan opis operatora uniformnog ukrštanja se može naći u [Spe91].

Izbor operatora ukrštanja mora biti prilagođen prirodi problema koji se rešava. Naime, kada je potrebno delimično sačuvati strukturu genetskog koda koristi se jednopoziciono ukrštanje. U slučajevima kada je potrebno što više razbiti i izmešati blokove u genetskom kodu, koristi se dvopoziciono ili višepoziciono ukrštanje. Ako su geni nezavisni, onda najbolje rezultate daje uniformno ukrštanje.

Za razliku od kodiranja i selekcije, gde korišćenje različitih pristupa može dovesti do bitnih razlika u performansama genetskog algoritma, kod ukrštanja su razlike mnogo manje, ali su ipak primetne.

1.5.1.4 Operatori mutacije

Mutacija je jedna od najznačajnijih operatora kod genetskih algoritama. Ovaj operator može biti realizovan na različite načine, a često presudno utiče na tok izvršavanja algoritma. Najpoznatije varijante operatora mutacije su: prosta mutacija, mutacija pomoću binomne raspodele i mutacija pomoću normalne raspodele.

Kada se jedinke binarno kodiraju, a populacija ne sadrži nekorektne jedinke, najčešće se koristi operator proste mutacije. Ovakav operator, bit po bit, procesira genetski kod, proveravajući pri tome da li je došlo do mutacije. Nivo mutacije predstavlja verovatnoću sa kojom svaki bit mutira i zadat je na početku izvršavanja algoritma. Međutim, mora se voditi računa da se za nivo mutacije odabere neka mala vrednost kako genetski algoritam ne bi prerastao u slučajnu pretragu. Prosta mutacija se, u nekim slučajevima, realizuje i preko binarnog niza koji se naziva maska. Za svaku jedinku se "maska" slučajno generiše i čuva informaciju o tome na kojoj poziciji u genetskom kodu treba da dođe do promene gena. Poboljšanje proste mutacije se može ostvariti i korišćenjem "zaleđenih" bitova, sto će detaljno biti objašnjeno u narednom poglavlju.

Korišćenje binomne ili normalne raspodele može bitno ubrzati realizaciju operatora proste mutacije. Mutacija pomoću binomne raspodele se zasniva na činjenici da slučajna promenljiva, koja predstavlja broj mutiranih gena jedinke, ima binomnu raspodelu $B(N_{bit}, p_{mut})$. Pri tome je N_{bit} dužina genetskog koda, a p_{mut} nivo mutacije. Mutacija se izvodi tako što se prvo bira slučajan broj $s \in [0,1]$, a zatim se pronalazi broj X_{mut} za koji je ispunjeno: $F(X_{mut}) \leq F(s) < F(X_{mut}+1)$, gde je F funkcija raspodele binomne raspodele. Na kraju se na slučajan način bira tačno X_{mut} pozicija u genetskom kodu, na kojim se izvršava mutacija.

Kada je proizvod dužine genetskog koda i nivoa mutacije dovoljno veliki, pogodno je da se pomenuta binomna raspodela aproksimira normalnom raspodelom $N(N_{bit} \cdot p_{mut}; N_{bit} \cdot p_{mut} \cdot (1-p_{mut}))$. Slično kao i kod prethodne realizacije mutacije, uz korišćenje inverzne funkcije, za funkciju normalne raspodele se određuje broj mutiranih gena X_{mut}, a zatim vrši mutacija. Varijanta ovog operatora za primenu na celoj populaciji je takođe razvijena, prvenstveno zato što se pri mutaciji genetski kodovi svih jedinki mogu statistički posmatrati kao jedna celina. Detaljnije informacije o gore navedenim konceptima mutacije mogu se pronaći u [Kra00, Toš04].

Posebno su interesantni slučajevi kada mutacija nije uniformna, odnosno kada geni genetskog koda nisu ravnopravni, već je neke delove koda jedinke potrebno mutirati sa manjom ili većom verovatnoćom. Tada se obično koristi normalna mutacija koja se primenjuje u skladu sa normalnom raspodelom, ili

eksponencijalna mutacija kod koje broj mutiranih gena u kodu eksponencijalno opada.

Za slučajeve kada genetski algoritam koristi kodiranje celim ili realnim brojevima, razvijeni su drugi koncepti mutacije poput: zamene gena slučajno izabranim brojem, dodavanje ili oduzimanje male vrednosti, množenje brojem bliskim jedinici itd. Potrebne vrednosti za pomenute operatore su slučajne i mogu imati uniformnu, eksponencijalnu, binomnu ili Gausovu raspodelu. Detaljnije o ovim operatorima se može naći u [BeD93a,BeD93b].

U nekim situacijama nije moguće direktno primeniti standardne genetske operatore ili njihove modifikacije, ili ih je moguće primeniti, ali daju relativno loše rezultate. Najčešće je to posledica specifičnog načina kodiranja ili postojanja velikog broja ograničenja koja se (sva) ne mogu implementirati u genetski kod. Tada se primenjuju genetski operatori mutacije i ukrštanja koji zavise od prirode problema. Ovakvi operatori uglavnom čuvaju korektnost jedinki na koje se primenjuju. Neki od primera operatora zavisnih od problema su: hipermutacija (Hypermutation) koja je opisana u [Crr01], N4H mutacija ([Alk04]), Modified-Basic Operator (MBO), String-of-Change Operator (SOC), Template Operator (TEMP), Backward Crossover Operator (BACK) prikazani u [Boz02] itd. Pomenuti operatori iskorišćeni su za rešavanje nekih lokacijskih problema.

1.5.1.5 Karakteristike genetskih algoritama i ostali aspekti

Izvršavanje genetskih algoritama, koji su u osnovi stohastičke metode pretrage dopustivog prostora rešenja, može trajati beskonačno dugo ukoliko im se ne nametne kriterijum zaustavljanja. Najprimenjivaniji kriterijumi završetka genetskog algoritma su:

- dostignuti maksimalni broj generacija,
- sličnost jedinki u populaciji,
- ponavljanje najbolje jedinke određeni (maksimalni) broj puta,
- dostizanje optimalnog rešenja ako je ono unapred poznato,
- dokazana optimalnost najbolje jedinke (ukoliko je to moguće),
- ograničeno vreme izvršavanja genetskog algoritma i
- prekid od strane korisnika.

Kriterijum završetka je najosetljivije mesto genetskih algoritama. Navedeni kriterijumi imaju dobre i loše strane, pa se u praktičnim primenama najbolje pokazalo njihovo kombinovanje. Korišćenjem više kriterijuma zaustavljanja smanjuje se mogućnost loše procene prekida genetskog algoritma.

Politika zamene generacija je jedan od bitnih aspekata genetskih algoritama. Najčešće se primenjuju sledeće strategije:

- Generacijska, koja u svakoj generaciji menja sve jedinke u populaciji.
- Stacionarna, koja u svakoj generaciji generiše samo deo populacije, dok se preostale jedinke prenose iz prethodne populacije.
- Elitistička strategija, koja bez primene genetskih operatora i računanja funkcije prilagođenosti, u svaku generaciju propušta određen broj elitnih jedinki, 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 strategijama.

Jedan od nedostataka genetskih algoritma je u tome što ne postoji jedinstvena kombinacija parametara (veličina početne populacije, nivo mutacije, nivo ukrštanja itd) koja bi bila najbolja za sve probleme, već se oni u svakom konkretnom slučaju posebno podešavaju eksperimentalnim putem. Takođe, raznovrsnost genetskog materijala nije jednaka u svim fazama izvršavanja genetskog algoritma pa se često optimalne vrednosti nivoa ukrštanja ili mutacije menjaju tokom izvršavanja genetskog algoritma (videti [Brm91, Bëc92, Bëc93, Sri94]). Načini promene parametara u toku izvršavanja se mogu podeliti u dve kategorije:

- fiksna promena parametara, koja unapred zadaje linearno ili eksponencijalno povećanje ili smanjivanje parametara i
- adaptivna promena parametara, koja može da promeni parametar u zavisnosti od toga kakve je rezultate operator do tada dao, odnosno koliko je bio uspešan.

Često realni problemi nameću da se genetski algoritmi kombinuju sa drugim heuristikama. Heuristike se mogu koristiti, kako za poboljšavanje početne populacije tako i svake naredne generacije. Mogu se primenjivati na celu populaciju, jedan njen deo ili na pojedinačnu jedinku. Paralelizacijom i keširanjem se, takođe, bitno poboljšavaju performanse genetskog algoritma, kao što je prikazano u [Kra00].

Genetski algoritmi su se pokazali vrlo uspešnim pri rešavanju široke klase problema optimizacije. Prikaz svih primena genetskih algoritama daleko prevazilazi obim ovog rada. Veliki broj primena genetskih algoritama se može naći u [Ala08]. Neke od novijih primena genetskih algoritama su:

- diskretni lokacijski problemi [Alp03, Dre03, Dvo99, Fil00, Kra98, Kra99b, Kra00, Kra01a, Kra01b, Mar08a, Mar08b, Mar08c, Sta07a],
- hab lokacijski problemi [Fil06, Kra05, Kra06, Kra07, Sta07b, Sta08],
- minimalna povezana dopuna grafa [Lju00a, Lju00b, Lju01, Lju03, Lju04, Trm00],
- metrička dimenzija grafa [Kra08a, Kra08b, Kra08c],
- raspodela poslova [Sav08a, Sav08b],
- dizajn računarskih mreža [Kra02, Kra08d],
- zadovoljivost formula [Ognj01, Ognj04b],
- višedimenzioni problem ranca [Puc06, Rai05],
- minimalna međuzavisnost binarnih nizova [Kov08],
- maksimalno balansirana povezana particija grafa [Đur08],
- izbor indeksa baze podataka [Kra03],
- rutiranje vozila [Kra08e],
- pokrivanje čvorova grafa [Mil08],
- bojenje grafa [Juh06].

2 PROBLEM HIJERARHIJSKOG RASPOREĐIVANJA RADNIKA

Jedan od osnovnih preduslova za uspešno poslovanje firmi je efikasna raspodela radne snage. Manjak radnika doprinosi smanjenju troškova, ali istovremeno i pogoršava nivo usluge. Sa druge strane, veći broj radnika će poboljšati nivo usluge, ali i dovesti do povećanja izdataka poslodavaca, kao i do problema sa radnim prostorom. Firme se trude da pronađu idealan balans između broja zaposlenih, njihovih kvalifikacija i ukupne cene angažovanja, vodeći računa da se pri tome maksimalno zadovolje postavljeni zadaci. Hijerarhijsko raspoređivanje radnika (Hierarhical workforce sheduling problem - HCLP) u velikim firmama predstavlja ozbiljan problem, posebno poslednjih godina kada sa brojem poslova raste i broj radnika koji te poslove treba da obave. Polazne parametre problema predstavljaju hijerarhija stručne kvalifikacije radnika i različite potrebe poslodavca. Ova tematika je dosta razmatrana u literaturi i postoji mnogo radova u kojima su prezentovani ovakvi problemi: [Bil99, Emm85, Emm91, Ern04, Hun93, Hun94, Hun96, Hun04, Nar97, Nar00, Sec07, Top02]. Glavni princip, pri definisanju problema, je da stručniji radnici mogu da obavljaju manje zahtevne poslove, ali ne i obrnuto. Naravno, stručniji radnici moraju biti i više plaćeni pa se, kako u praksi tako i pri rešavanju ovog problema, teži da se takve situacije izbegnu. Upravo zbog toga pomenuti slučaj je specijalan, pre nego uobičajen. U [Bil99] je predstavljen osnovni model problema raspoređivanja radnika. Takav model uključuje tri parametra: stručnost radnika, težinu poslova i dane u nedelji. Tačnije, model bi trebao da zadovoljava sledeće uslove:

- radna nedelja ima sedam dana, koji su od ponedeljka do nedelje obeleženi brojevima od 1 do 7.
- svi radnici rade puno radno vreme, ali su klasifikovani tako da su najkvalifikovaniji ranga 1 i tako dalje do ranga *m,* pri čemu su različiti

tipovi radnika i različito plaćeni. Poslove ranga *l,* svakog dana, mogu izvršavati radnici tog ili višeg ranga.

- svaki radnik ima *n* slobodnih dana u nedelji (obično je *n* = 2,3,4) .
- zadatak je pronaći najjeftinije rešenje koje omogućava da se svi nedeljni poslovi završe.

Ovde indeks *k* označava tip radnika, *m* - broj različitih tipova radnika (pri čemu radnici sa manjim indeksom *k* imaju viši rang), indeks *l* označava tip posla (poslova ima koliko i vrsta radnika - *m),* dok indeks *j* određuje o kom danu u nedelji se radi. Imajući u vidu prethodno uvedene indekse mogu se definisati promenljive:

- c_k - cena radnika ranga *k*,
- w_k - broj radnika tipa *k*,
- x_{klj} - broj radnika tipa *k*, angažovanih dana *j* na poslu *l*,
- y_{kj} - broj radnika tipa *k* koji ne rade dana *j* i
- d_{lj} - broj poslova ranga *l* koje je potrebno završiti dana *j*.

Sa ovako uvedenim indeksima i promenljivima matematički model celobrojnog linearnog programiranja (preuzet iz [Bil99]) se definiše na sledeći način:

Minimizovati:

$$Z = \sum_{k=1..m} c_k w_k \quad , \tag{2.1}$$

tako da važi:

$$\sum_{l \geq k} x_{klj} + y_{kj} = w_k \quad , \tag{2.2}$$

pri čemu *k = 1, ... , m, j = 1, ... , 7*.

$$\sum_{j} y_{kj} \geq w_k n \quad , \tag{2.3}$$

(k = 1, ... , m, n=2, 3, 4),

$$\sum_{k \le l} x_{klj} = d_{lj}, \qquad (2.4)$$

(l = 1, ... , m, j = 1, ... , 7).

Uslov (2.2) kazuje da svakoga dana određen broj radnika radi dok ostali odmaraju, tačnije da su svi na broju. Uslov (2.3) obezbeđuje da svaki radnik ima odgovarajući broj slobodnih dana, a uslov (2.4) garantuje da će planirani poslovi biti završeni. Važno je napomenuti da se pri formulisanju problema vodi računa da važi nejednakost $c_1 > c_2 \ldots > c_m$.

Primer 1: Za $m = 3$ - odnosno u situaciji kada postoje tri različita tipa radnika, $n = 2$ (radi se 5 dana u nedelji), cene $c_1 = 12$, $c_2 = 8$ i $c_3 = 6$ i neka je u Tabeli 2.1 prikazan broj poslova određenog ranga koje treba završiti nekog dana u nedelji.

Tabela 2.1 Poslovi koje treba završiti u toku nedelje

Rang posla	Pon.	Uto.	Sre.	Čet.	Pet.	Sub.	Ned.
1	2	3	1	4	2	5	5
2	2	1	2	2	2	1	1
3	4	4	4	1	6	4	4

Optimalno rešenje je: $w_1 = 5$, $w_2 = 2$, $w_3 = 5$, što znači da je za obavljanje poslova potrebno angažovati pet najkvalifikovanijih radnika, dva srednje i pet najslabije kvalifikovanih radnika. Pri tome je u Tabeli 2.2 dat broj radnika koji odmaraju određenog dana:

Tabela 2.2 Raspored odmaranja radnika

Rang radnika	Pon.	Uto.	Sre.	Čet.	Pet.	Sub.	Ned.
1	3	1	4	1	1	0	0
2	0	2	0	0	0	1	1
3	1	1	1	4	1	1	1

Tabela 2.3 prikazuje koji posao kog dana radi koji radnik. Na primer, radnik 3 najviše kategorije (1) ne radi ponedeljkom i sredom, utorkom, četvrtkom subotom i nedeljom radi poslove svog (najvišeg) ranga, dok petkom radi posao druge klase itd.

Tabela 2.3 Poslovi koje radnici obavljaju u toku radne nedelje

Rang radnika	Radnik	Pon.	Uto.	Sre.	Čet.	Pet.	Sub.	Ned.
1	1	-	1	-	1	1	1	1
	2	-	1	-	1	1	1	1
	3	-	1	-	1	2	1	1
	4	1	-	1	-	2	1	1
	5	1	2	-	1	-	1	1
2	1	2	-	2	2	3	-	2
	2	2	-	2	2	3	2	-
3	1	-	3	3	-	3	3	3
	2	3	-	3	-	3	3	3
	3	3	3	-	3	-	3	3
	4	3	3	3	-	3	-	3
	5	3	3	3	-	3	3	-

U ovom radu je razmatrano uopštenje prikazanog modela kod kojeg radno vreme nije fiksirano, odnosno radnici mogu da rade i preko 8 sati, ali u takvim situacijama rade manji broj dana u nedelji. Detaljan opis problema sledi u nastavku.

2.1 Matematička formulacija problema

Matematički model linearnog celobrojnog programiranja za rešavanje problema hijerarhijskog raspoređivanja radnika (preuzet iz [Sec07]) definisan je na sledeći način:

Minimizovati:

$$Z = \sum_{b=1..B} \sum_{k=1..m} c_{bk} w_{bk} \ , \tag{2.5}$$

tako da važi:

$$\sum_{l>k} x_{bklj} + y_{bkj} = w_{bk} \ , \tag{2.6}$$

pri čemu b = 1,2,3, k = 1, ... , m, j = 1, ... , 7.

$$\sum_{j} y_{bkj} \geq A_{bn} w_{bk} \ , \tag{2.7}$$

(b = 1,2,3, k = 1, ... , m, n=2,3,4)

$$\sum_{b=1,..,3} \sum_{k \leq l} x_{bklj} = d_{lj} , \tag{2.8}$$

(l = 1, ... , m, j = 1, ... , 7).

Ovde, slično kao kod modela iz [Bil99], indeks k označava tip radnika, broj različitih tipova radnika je m (pri čemu radnici sa manjim indeksom k imaju viši rang), indeks l označava tip posla (poslova ima koliko i vrsta radnika - m), indeks j određuje o kom danu u nedelji se radi, a indeks b označava tip radnog vremena.

Sa ovako uvedenim indeksima, c_{bk} predstavlja cenu radnika ranga k sa radnim vremenom tipa b, w_{bk} - broj radnika tipa k sa radnim vremenom tipa b, x_{bklj} - broj radnika tipa k sa radnim vremenom tipa b angažovanih dana j na poslu l, y_{bkj} - broj radnika tipa k sa radnim vremenom tipa b koji ne rade dana j i d_{lj} - broj poslova ranga l koje je potrebno završiti dana j.

Jasno je da uslovi (2.1) - (2.4), koji opisuju model iz [Bil99], predstavljalju specijalan slučaj modela (2.5) - (2.8) iz [Sec07]. Fiksiranjem *B=1* gubi se raznovrsnost radnih vremena, tako da svi radnici imaju osmočasovno radno vreme, a samim tim i rade pet dana u nedelji.

Primer 2.2: Broj različitih tipova radnika je *m* = 3, broj različitih tipova radnih vremena *B* = *3*, pa je u slučaju *b* = 1 radno vreme 8 sati dnevno, a radnik radi 5 dana u nedelji. Za *b* = 2 radno vreme je 10 sati, a radi se 4 dana u nedelji, dok za *b* = 3 radnik radi 3 dana u nedelji po 12 sati. Cene radnika, u zavisnosti od *k* i *b* (c_{bk}), su predstavljene u Tabeli 2.4.

Tabela 2.4 Cene radnika

b \ k	**1**	**2**	**3**
1	12	8	6
2	10	6	4
3	8	5	3

Poslovi (d_{lj}) različitih rangova, koje treba izvršiti u toku nedelje, dati su u Tabeli 2.5.

Tabela 2.5 Poslovi

Rang posla	**Pon.**	**Uto.**	**Sre.**	**Čet.**	**Pet.**	**Sub.**	**Ned.**
1	2	3	1	4	2	5	5
2	2	1	2	2	2	1	1
3	4	4	4	1	6	4	4

Optimalno rešenje, čija je vrednost *Z=98*, odgovara rasporedu radnika (w_{bk}) koji je prikazan u Tabeli 2.6.

Tabela 2.6 Raspored radnika na različitim radnim mestima

b \ k	**1**	**2**	**3**
1	3	0	0
2	1	3	2
3	1	0	6

U Tabeli 2.7 prikazani su detalji o radu svakog radnika, odnosno koji radnik kog dana radi koji posao, u slučaju da tog dana ne odmara. Ukoliko je radnik nekog dana slobodan u njegovoj koloni za taj dan se nalazi crta (-).

Na primer, radnik 3 koji je ranga 1 sa radnim vremenom tipa 1 radi utorkom, četvrtkom, petkom, subotom i nedeljom poslove svog (1) ranga. Ukupno tri radnika rade pet dana u nedelji po 8 sati, osam radnika rade 10 sati dnevno četiri dana u nedelji dok tri radnika rade tri dana u nedelji po 12 sati.

Tabela 2.7 Poslovi i radnici koji ih izvršavaju u toku nedelje

Vrsta radnog vremena	Tip radnika	Radnik	Pon.	Uto.	Sre.	Čet.	Pet.	Sub.	Ned.
1	1	1	-	1	-	1	1	1	1
		2	-	1	-	1	1	1	1
		3	-	1	-	1	2	1	1
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
2	1	1	1	-	-	1	-	1	1
	2	1	-	-	2	2	-	2	2
		2	2	-	2	2	2	-	-
	3	1	-	-	3	-	3	3	3
		2	-	-	-	3	3	3	3
		3	3	3	-	-	-	3	3
		4	3	3	3	-	3	3	-
		5	3	3	3	-	3	-	-
		6	3	3	3	-	3	-	-
3	1	1	-	-	-	-	1	1	1
	2	1	2	2	-	-	2	-	-
	3	1	-	-	-	-	3	3	3

2.2 Genetski algoritam za rešavanje problema hijerarhijskog raspoređivanja radnika

2.2.1 Kodiranje i računanje funkcije cilja

U ovoj implementaciji je primenjeno kodiranje realnim brojevima za određivanje svih elemenata višedimenzionog niza *X*. Nizom *X* predstavljen je broj radnika tipa *k* sa radnim vremenom *b*, angažovanih dana *j* na poslu *l*. Svakom od članova niza pridružuje se realan broj koji se dobija iz genetskog koda. Pošto je problem numerički stabilan, svaki realan broj može se kodirati samo pomoću 16 bitova, čime se dobija smanjenje dužine genetskog koda. Na osnovu realnih brojeva dobijenih iz genetskog koda, za svako *l* i *j,* računaju se celobrojne vrednosti niza x_{bklj} korišćenjem tzv. D'Ont-ovog sistema [Don08]. Ovaj sistem ima minimalnu grešku uzorkovanja (sampling error), pa se zato koristi i kod raspoređivanja poslaničkih mandata u proporcionalnom izbornom sistemu. Nakon pebrojavanja glasova, za svaku listu se računaju količnici po formuli $\frac{V}{s+1}$, pri čemu je *V* broj glasova koje je lista dobila, a s broj mandata koje je lista dobila do tog trenutka. Sledeći mandat dobija lista koja ima najveći količnik, zatim se količnici ponovo računaju i postupak se ponavlja sve dok se ne raspodele svi mandati.

Korišćenje D'Ont-ovog sistema pri rešavanju problema hijerarhijskog raspoređivanja radnika podrazumeva da se za fiksirano *l* i *j,* u zavisnosti od realnih brojeva koji su pridruženi *X*-ovima, povećava odgovarajući x_{bklj} i sve tako dok se ne zadovolje svi poslovi d_{lj}. Preciznije, za svako *l* i za svako *j* primenjuje se po jedan D'Ont-ov sistem za raspoređivanje d_{lj} u skladu sa formulom (2.8). Kao što je već opisano, u svakom koraku formiraju se količnici u kojima je deljenik realni broj koji se uzorkuje (dobijen iz genetskog koda), a delilac je do tada dobijena celobrojna vrednost niza *X* uvećana za 1. Vrednost niza *X,* koja odgovara najvećem količniku, se povećava za 1, čime se povećava i njegov delilac. Ukupan broj koraka je d_{lj}.

Niz w_{bk} se dobija na osnovu elemenata x_{bklj} za svako *b* i *k*. Izračuna se $\frac{\sum_{k>l}\sum_{j} x_{bklj}}{7-A_{bn}}$ i za svaki dan u nedelji *j* se nalazi $\sum_{k>l} x_{bklj}$. Maksimum izračunatih brojeva (njih 8) postaje vrednost w_{bk}. Vrednosti $y_{bkj} = -w_{bk} + \sum_{l>k} x_{bklj}$ dobijaju

se na osnovu uslova (2.6). Na ovaj način obezbeđuje se dopustivost rešenja u skladu sa uslovima (2.6)-(2.8). Vrednost Z se dobija sumiranjem po formuli (2.5).

Lema 2.1: Složenost izračunavanja funkcije cilja predstavljenim genetskim algoritmom je $O(m^2 \cdot B \cdot MaxD)$, pri čemu je $MaxD$ najveća vrednost u nizu poslova.

Dokaz

Imajući u vidu gore prikazane korake, jasno je da je složenost inicijalizacije niza X $O(7 \cdot B \cdot m^2) = O(B \cdot m^2)$, a određivanja članova niza X $O(7 \cdot m^2 \cdot B \cdot MaxD + 7 \cdot m^2 \cdot B) = O(m^2 \cdot B \cdot MaxD)$. $MaxD$ je najveća vrednost u nizu poslova d_{lj} i predstavlja gornju granicu pri računanju vrednosti D'Ont-ovim sistemom. Složenost izračunavanja članova niza W je $O(7 \cdot B \cdot m) = O(B \cdot m)$, a niza Y $O(7 \cdot B \cdot m^2) = O(B \cdot m^2)$. Vrednost funkcije cilja računa se sa složenošću $O(B \cdot m)$. Na kraju, imajući u vidu složenosti svih koraka izračunavanja funkcije cilja, jasno je da je njena složenost $O(m^2 \cdot B \cdot MaxD)$. □

2.2.2 Selekcija

Operator selekcije je mehanizam kojim se biraju roditelji koji daju potomke za sledeću generaciju. Turniri su takmičenja dve ili više jedinki koje se nadmeću da bi preživele, odnonosno učestvovale u sledećoj generaciji. Praktično, kada je u pitanju turnirska selekcija, jedinka će biti izabrana ukoliko je bolja od nekoliko slučajno izabranih protivnika. Veličina turnira N_{tour} je jedini parametar ove selekcije, a složenost izračunavanja pobednika je $O(N_{tour} \cdot N_{nnel})$, gde je N_{nnel} broj jedinki koje nisu elitne, odnosno broj turnira koje je potrebno organizovati. Kako brojevi N_{tour} i N_{nnel} ne zavise od veličine problema, može se uzeti da je složenost izračunavanja konstantna, odnosno $O(1)$. Osnovni nedostatak klasične turnirske slekcije je to što se veličina turnira N_{tour} bira iz skupa celih brojeva. Za razliku od toga, ostali operatori se podešavaju realnim parametrima, što omogućava veću fleksibilnost. U praksi je izbor parametara za turnirsku selekciju sužen na dve – tri vrednosti, tako da se teško može podesiti odnos istraživanja i eksploatacije. Kada je ovakav operator selekcije u pitanju, često se događa da za jedan parametar konvergencija bude veoma spora, a da povećanje parametra za jedan dovede do preuranjene konvergencije, odnosno vezivanja rešenja za lokalni ekstremum.

Upravo, iz gore pomenutih razloga, pri rešavanju problema hijerarhijskog raspoređivanja radnika korišćeno je jedno unapređenje turnirske selekcije poznato kao Fino gradirana turnirska selekcija [Fil00, Fil03]. Ovaj operator koristi realni (racionalni) parametar F_{tour} koji označava očekivanu prosečnu veličinu turnira. Kao i kod turnirske selekcije, jedinka će biti izabrana ukoliko bude bolja od određenog broja slučajno izabranih protivnika, ali veličina izbornih turnira nije jedinstvena u okviru populacije. Primenjuju se dva tipa turnira: jedan se održava k_1 puta sa turnirima veličine $\lceil F_{tour} \rceil$, a drugi k_2 puta sa turnirima veličine $\lfloor F_{tour} \rfloor$ jedinki. Pri tome je ispunjeno $F_{tour} \approx \frac{k_1 \lfloor F_{tour} \rfloor + k_2 \lceil F_{tour} \rceil}{N_{nnel}}$, $k_1+k_2=N_{nnel}$.

U ovoj implementaciji je F_{tour} = 5.4, sa pridruženim vrednostima $k_1=30$ i $k_2=20$ za N_{nnel} = 50. Kao i kada je u pitanju turnirska selekcija, složenost izračunavanja ovog operatora je konstantna (*O(1)).*

2.2.3 Ukrštanje

Na jedinke odabrane operatorom selekcije se primenjuje operator ukrštanja koji rekombinacijom genetskog materijala od dve odabrane (roditeljske) jedinke stvara dve nove jedinke (potomke). Potomci u novu generaciju prenose dobre osobine svojih roditelja. Pri rešavanju problema korišćeno je jednopoziciono ukrštanje kod kojeg se na slučajan način biraju parovi roditelja iz populacije i broj $k \in \{0,..., d-1\}$ koji predstavlja tačku ukrštanja (gde je d dužina genetskog koda). Svi geni, počevši od pozicije $k+1$ do poslednje pozicije $d-1$ u genetskim kodovima roditelja, uzajamno menjaju mesta stvarajući pri tom dva nova potomka, kao što se može videti na slici 2.1.

Roditelj 1:	XX-XXXXXX	Potomak1:	XX-YYYYYY
Roditelj 2:	YY-YYYYYY	Potomak2:	YY-XXXXXX

Slika 2.1 Šema jednopozicionog ukrštanja

U genetskom algoritmu za rešavanje problema hijerarhijskog raspoređivanja radnika korišćeno je jednopoziciono ukrštanje sa nivoom ukrštanja 0.85, što znači da u ukrštanju učestvuje 85% jedinki.

2.2.4 Mutacija

Za operator mutacije genetskog algoritma izabrana je prosta mutacija sa zaleđenim bitovima.

Tokom izvršavanja genetskog algoritma može se desiti da na istoj bit-poziciji skoro sve jedinke u populaciji imaju istu vrednost. Takve bitove definišemo kao "zaleđene", a njih može biti više u istoj generaciji. Njihova pojava predstavlja određeni nedostatak u izvršavanju genetskog algoritma, jer ako je broj zaleđenih bitova q pretraživački prostor postaje 2^q puta manji i mogućnost preuranjene konvergencije rapidno raste. Lako je uočiti da operatori selekcije i ukrštanja nikako ne mogu promeniti vrednost zaleđenog bita. Jedino operator mutacije može promeniti vrednost zaleđenog bita, ali je nivo mutacije uglavnom vrlo mali, pa često nije dovoljan da bi povratio izgubljene regione prostora pretrage. Nivo mutacije ne sme da bude veliki jer onda genetski algoritam gubi svoj smisao i postaje slučajna pretraga. Upravo zbog toga, pogodno je nivo mutacije povećavati samo na zaleđenim bitovima, jer je tu situacija najproblematičnija.

Pri rešavanju zadatog matematičkog modela, za osnovni nivo mutacije uzeta je vrednost koja je dovodila do najboljih rezultata prilikom testiranja. Ona se dobija iz formule (2.9).

$$p_{mut} = \frac{10.8}{\mathrm{b} \cdot 7 \cdot \mathrm{m} \cdot (\mathrm{m}+1)} \tag{2.9}$$

Na zaleđenim bitovima nivo mutacije je bio 2.5 puta veći od osnovnog nivoa p_{mut} , što predstavlja dobar kompromis za vraćanje izgubljenog pretraživačkog prostora pri eventualnoj pojavi zaleđenih bitova.

2.2.5 Politika zamene generacija

Pri rešavanju problema upotrebljeni su parametri koji su davali najbolje rezultate pri testiranju. Korišćena je veličina populacije od 150 jedinki. Primenjena je elitistička strategija koja favorizuje dobro prilagođene jedinke. Praktično, u svakoj generaciji 2/3 populacije (100 jedinki sa najboljom funkcijom prilagođenosti) direktno prelazi u sledeću generaciju, dok se ostalih 50 jedinki dobija primenom genetskih operatora na svih 150 jedinki.

Da bi se izbegla mogućnost preuranjene konvergencije uklanjaju se sve višestruke pojave iste jedinke u populaciji. Uklanjanje se vrši implicitno, postavljanjem funkcije prilagođenosti date jedinke na nulu, čime ona gubi šansu da se pojavi u narednoj generaciji. Takođe se ograničava broj pojava jedinki sa istom vrednošću, a različitim genetskim kodovima na 40.

2.3 Keširanje

Performanse genetskog algoritma su poboljšane primenom tehnike keširanja, koja je prvi put primenjena u [Kra99a]. Najzahtevniji deo, kada je u pitanju vreme izvršavanja, je izračunavanje funkcije cilja. Zbog toga se izračunate vrednosti funkcije cilja, zajedno sa odgovarajućim genetskim kodom, smeštaju u memoriju. Pri izvršavanju genetskog algoritma proverava se da li tekuća jedinka u populaciji postoji u keš memoriji, odnosno da li se pojavljivala u prethodnim generacijama. Ako je to tačno, njena funkcija cilja se ne izračunava, već se uzima vrednost iz keš memorije. U suprotnom, računa se vrednost funkcije cilja i zajedno sa genetskim kodom smešta u keš memoriju.

Za keširanje se koristi slobodna RAM memorija koja se alocira dinamički. Keš memorija se deli u blokove, koji su kod genetskih algoritama jednake veličine. Kao što je ranije pomenuto, svaki blok keš memorije čuva vrednosti koje se odnose na jednu jedinku populacije u nekoj generaciji, odnosno čuva sledeće podatke: genetski kod, vrednost njegove funkcije cilja i indikator validnosti jedinke (koji je binarna promenljiva).

U ovoj implementaciji je korišćena hijerarhija keširanja sa jednim nivoom. Ona je jednostavna za implementaciju i time su izbegnuti problemi konzistentnosti keša, koji se javljaju samo kod hijerarhija keširanja sa više nivoa.

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

- Pretraživanje keš memorije korišćenjem heš tabele za pronalaženje tražene jedinke (ako takav blok postoji, odnosno ako se jedinka nalazi u keš memoriji).
- Izbacivanje najstarijeg nekorišćenog bloka iz keš memorije (korišćenjem reda) ako je keš memorija potpuno popunjena.

- Ubacivanje aktuelne jedinke u keš memoriju ako se ta jedinka već ne nalazi u keš memoriji.

Praktično, proverava se da li se tražena jedinka nalazi u keš memoriji. Vraća se pokazivač na blok memorije gde se tražena jedinka nalazi, ili *NULL* pointer ukoliko se jedinka ne nalazi u keš memoriji.

Ako je jedinka već smeštena u keš memoriju, očitava se vrednost funkcije cilja odgovarajuće jedinke iz memorije, umesto da se ponovo izračunava. Posle toga se odgovarajući blok memorije označava kao poslednji korišćeni i time penje na vrh reda.

Ako ne postoji blok memorije koji sadrži traženu jedinku računa se funkcija cilja za odgovarajuću jedinku. Zatim se proverava da li je keš memorija puna pa se, u tom slučaju, izbacuje najstariji blok memorije, odnosno blok kojem se najranije u prošlosti pristupalo. Nakon toga se podaci o odgovarajućoj jedinki (genetski kod, vrednost funkcije cilja i indikator validnosti) smeštaju u slobodni blok memorije umesto prethodno izbačenih podataka, ili na prvo slobodno mesto ukoliko keš memorija nije bila puna pre ove operacije. Na kraju se ovaj blok označava kao poslednje posećeni.

Red pokazivača na keš memoriju je organizovan u skladu sa jedinkama populacija tokom generacija genetskog algoritma i njihovim nalaženjem /nenalaženjem u keš memoriji. Na vrhu reda se nalaze blokovi kojima se najskorije pristupalo (ili su dodavani), dok se na kraju reda nalaze blokovi kojima se pristupalo najdalje u prošlosti. U konkretnom slučaju, broj keširanih vrednosti funkcije cilja u heš-red tabeli je ograničen na $N_{cache}=5000$.

2.4 Algoritam

Uzimajući u obzir sve prethodno opisane detalje, na slici 2.2 predstavljen je šematski prikaz implementiranog genetskog algoritma.

```
Učitavanje_ulaznih_podataka();
Generisanje_Početne_Populacije();
while (not Kriterijum_Zaustavljanja_GA()) do
    for (i = (N_elit +1)  to  N_pop ) do
        if (Postoji_u_kešu(i)) then
            obj[i] = Pronađi_u_kešu(i);
        else
            obj[i] = Funkcija_Cilja(i);
            Stavi_u_keš(i, obj[i]);
            if (Pun_keš()) then
                Izbaci_LRU_blok_keša()
            endif
        endif
    endfor
    Računanje_funkcije_prilagođenosti();
    Selekcija();
    Ukrštanje();
    Mutacija();
endwhile
Štampanje_Izlaznih_Podataka();
```

Slika 2.2 Genetski algoritam

2.5 Rezultati

U primerima 2.3 i 2.4 su prikazani rezultati dobijeni primenom genetskog algoritma na jedine dve instance koje su poznate iz literature [Bil99, Sec07]. Kasnije su generisane i testirane razne instance, a u primerima 2.5 i 2.6 su prikazani rezultati genetskog algoritma na instancama kod kojih je redom $m = 4$ i $m = 5$. Kod generisanja novih instanci problem su predstavljala brojna ograničenja. Naime, broj dana u nedelji je fiksiran na 7, a dnevno radno vreme je u praksi uglavnom ograničeno na 8 do 12 časova. Takođe, broj mogućih kategorija stručnosti je relativno mali. Eksperimentalni rezultati pokazuju da genetski algoritam za date instance dostiže optimalno rešenje u relativno kratkom vremenu izvršavanja. Pošto vreme izvršavanja genetskog algoritma raste polinomski sa dimenzijom problema, mogu se dobiti rešenja i na instancama veće dimenzije za koje nije poznato optimalno rešenje.

Primer 2.3 U Tabeli 2.8 prikazano je rešenje primera 2.1 dobijeno primenom genetskog algoritma.

Tabela 2.8 Raspored radnika

k / b	**1**	**2**	**3**
1	2	0	0
2	3	2	3
3	0	1	5

Genetski algoritam je dostigao optimalno rešenje navedenog problema, čija je vrednost funkcije cilja 98. Međutim raspored radnika je bio različit u odnosu na Tabelu 2.6. Iz toga se može zaključiti da postoji više različitih optimalnih rešenja datog primera.

Primer 2.4: Broj različitih tipova radnika je *m* = 3, broj različitih radnih vremena *B* = *3*, pa u slučaju *b* = 1 radno vreme je 8 sati dnevno a radnik radi 5 dana u nedelji, za *b* = 2 – radno vreme je 10 sati 4 dana u nedelji i za *b* = 3 radnik radi 3 dana u nedelji po 12 sati. Cene radnika u zavisnosti od *k* i *b* (c_{bk}) su predstavljene u Tabeli 2.9.

Tabela 2.9 Cene radnika

k / b	**1**	**2**	**3**
1	12	8	6
2	10	6	4
3	8	5	3

Poslovi (d_{lj}) različitih rangova, koje treba izvršiti u toku nedelje, prikazani su u Tabeli 2.10.

Tabela 2.10 Poslovi

Rang posla	Pon.	Uto.	Sre.	Čet.	Pet.	Sub.	Ned.
1	2	3	1	4	2	5	5
2	2	1	2	2	2	1	1
3	4	4	4	1	6	4	4

Optimalno rešenje, čija je vrednost funkcije cilja Z=98, je predstavljeno u Tabeli 2.11, za raspored radnika (w_{bk}).

Tabela 2.11 Raspored radnika

K / b	**1**	**2**	**3**
1	3	0	0
2	1	3	2
3	1	0	6

Primer 2.5. Broj različitih tipova radnika je *m* = 4, broj različitih radnih vremena *B* = *3*, cene radnika u zavisnosti od *k* i *b* (c_{bk}) su predstavljene u tabeli 2.12.

Tabela 2.12 Cene radnika

k / b	**1**	**2**	**3**	**4**
1	150	125	100	75
2	120	100	80	60
3	90	75	60	45

Poslovi (d_{lj}) različitih rangova, koje treba izvršiti u toku nedelje, dati su u Tabeli 2.13.

Tabela 2.13 Poslovi

Rang posla	Pon.	Uto.	Sre.	Čet.	Pet.	Sub.	Ned.
1	25	13	22	17	63	41	31
2	42	51	76	58	35	43	53
3	75	45	33	21	43	83	51
4	58	53	79	11	24	32	52

Optimalno rešenje, čija je vrednost funkcije cilja 26965, je prikazano u Tabeli 2.14.

Tabela 2.14 Raspored radnika

k / b	1	2	3	4
1	0	65	11	3
2	2	0	74	30
3	68	11	0	58

Primer 2.6. Broj različitih tipova radnika je *m* = 5, broj različitih radnih vremena *B* = *3*, Cene radnika u zavisnosti od *k* i *b* (c_{bk}) su predstavljene u Tabeli 2.15.

Tabela 2.15 Cene radnika

k / b	1	2	3	4	5
1	190	160	140	115	70
2	155	135	115	95	60
3	120	105	90	75	45

Poslovi (d_{lj}) različitih rangova, koje treba izvršiti u toku nedelje, prikazani su u Tabeli 2.16.

Tabela 2.16 Poslovi

Rang posla	**Pon.**	**Uto.**	**Sre.**	**Čet.**	**Pet.**	**Sub.**	**Ned.**
1	2	1	4	2	7	8	6
2	3	3	2	8	9	5	4
3	2	1	4	3	7	4	6
4	2	1	2	2	2	1	1
5	2	1	4	2	7	8	6

Vrednost fukcije cilja rešenja je 3715, a u Tabeli 2.17 su prikazane vrste radnika koje su potrebne da bi se optimalno obavili zadaci.

Tabela 2.17. Raspored radnika

k / **b**	**1**	**2**	**3**	**4**	**5**
1	2	5	3	2	4
2	3	0	0	0	1
3	3	3	4	0	1

3 POBLEM HIJERARHIJSKOG POKRIVANJA KORISNIKA

3.1 Problemi pokrivanja korisnika

Problem pokrivanja korisnika je jedan od specifičnih lokacijskih problema. Često se koristi pri određivanju lokacija za hitne službe poput policijskih i vatrogasnih stanica, hitne pomoći itd. One moraju da pokriju što veći broj stanovništva, a da pri tome troškovi opsluživanja budu što niži. U praksi su troškovi opsluživanja klijenta, uglavnom direktno proporcionalni rastojanju između klijenta i objekata. Smatra se da je određeno područje "pokriveno" uz pomoć datog objekta, ukoliko se nalazi na rastojanju manjem od nekog unapred definisanog "kritičnog" rastojanja. Primarni cilj je odabrati lokacije za uslužne objekte tako da što više potencijalnih klijenata bude pokriveno. Mnogi modeli ovog problema su definisani i rešavani, ali jedan od najvažnijih je problem maksimalnog pokrivanja korisnika (The maximal covering location problem - MCLP). Zadatak je da se od skupa ponuđenih lokacija snabdevača odabere fiksirani broj tako da se pokrije što više korisnika. Detaljan opis ovog problema se može videti u [Gal96].

Modeli maksimalnog pokrivanja se veoma često koriste pri rešavanju problema koji podrazumevaju postavljanje željenih objekata na ponuđene lokacije. U nekim situacijama ovi objekti su nepoželjni, kao na primer kada su u pitanju deponije smeća, nuklearni reaktori ili zatvori. Ovakvi objekti, neophodni društvu, obično imaju nepovoljan efekat na stanovništvo koje se nalazi u okolini. Naravno, postoje objekti koji su stanovništvu veoma poželjni u okolini poput domova zdravlja, vatrogasnih stanica, fakulteta, škola, itd. Iz prethodno navedenih razloga ovaj problem se može definisati kao problem maksimizacije ili minimizacije funkcije cilja.

Generalno, kao što je već pomenuto u uvodnom poglavlju, kod ovakvih problema postoje dve vrste funkcija cilja: *Min-Max* i *Max-Sum (Min-Max)*. Na primer, kada je u pitanju *Min-Max* funkcija cilja i kada je zadatak postaviti samo jednu "nepoželjnu" lokaciju, potrebno je pronaći lokaciju takvu da je najkraće rastojanje do klijenata maksimalno. Kada je u pitanju *Max-Sum* funkcija cilja zadatak je pronaći lokaciju objekta tako da se maksimizuje suma rastojanja od objekta do svih klijenata.

Za razliku od prethodnog primera, koji je polinomske složenosti izračunavanja, u praksi su jedino interesantni slučajevi kada je potrebno postaviti više objekata. U opštem slučaju ti problemi su *NP*-teški, što znači da se ne mogu rešiti uzastopnom primenom algoritama za rešavanje gore pomenutog problema pronalaženja lokacije pojedinačnog objekta. U zavisnosti od funkcije cilja, formulisano je dosta različitih problema. Na primer, jedan od *Min-Max* problema je i problem p-disperzije (p-dispersion), kod kojeg je potrebno pronaći lokacije za *p* objekata tako da minimalno rastojanje između svaka dva od ovih objekata bude što veće. Kod *Max-Sum* verzije problema disperzije cilj je maksimizovati sumu rastojanja između lociranih objekata. Kao što je već rečeno, ova dva problema su *NP*-teški. Više o problemima ovakve vrste može se naći u [Gal96].

Problem lokacije nepoželjnih objekata, čija funkcija cilja treba da pokrije što manje klijenata, je dosta čest u praksi, a samim tim se često razmatra i u literaturi. U radovima se takav problem naziva problemom minimalnog pokrivanja korisnika (Minimum covering location problem with distance constrains - MCLPDC). Lociranjem fiksiranog broja objekata cilj je u što većoj meri smanjiti broj pokrivenih klijenata (smatra se da je klijent pokriven ukoliko se nalazi na rastojanju manjem od definisanog za bar jednu od uspostavljenih lokacija).

Kada je broj lokacija koje treba uspostaviti veći od jedne kod MCLPDC problema, uslovi o minimalnosti rastojanja su obično definisani tako da spreče postavljanje svih objekata na istu lokaciju. Jasno je da bi takvo rešenje bilo optimalno, ali često i nerealno. Motivacija za definisanje i rešavanje MCLPDC problema, između ostalih sličnih problema, pronađena je i u problemu postavljanja nuklearnih reaktora, koji mogu imati opasne posledice za stanovništvo koje se nalazi u njihovoj okolini. Iz tog razloga poželjno je da što manje ljudi živi u njihovoj okolini, odnosno da bude “pokriveno”. Sa druge strane, iz bezbedonosnih razloga – pre svega zbog sve češćih terorističkih napada, potrebno je reaktore razdvojiti, jer ako se svi nalaze grupisani na jednom mestu, onda su pogodna meta za teroriste. Raspodela reaktora na različite lokacije može se obaviti korišćenjem rešenja MCLPDC problema uz dodatne uslove kojima je određena minimalna dopustiva distanca dva reaktora.

Jedan od srodnih problema je i problem eksproprijacije lokacija (Expropriation location problem - ELP), čija se formulacija može videti u

[Ber03]. Svakoj lokaciji je pridružena vrednost eksproprijacije. Lokacije koje se nalaze na rastojanju manjem od predefinisanog prečnika treba da budu ekspoprisane. U stvari, cilj ELP-a je da pronađe najbolje lokacije za fiksirani broj nepoželjnih objekata tako da ukupna vrednost eksproprijacije bude minimalna. U radu [Ber03] istraživan je i problem minimalnog pokrivanja lokacija u ravni (Minimal covering location problem on the plane) i prikazan algoritam koji ga rešava. Pri tome je testirana i osetljivost poluprečnika na promene vrednosti njegove dužine. Takođe, istraživan je i problem minimalnog pokrivanja na mreži (Minimal covering location problem on a network), ali pod imenom Problem 2 ELP-a na mreži.

Jedan od problema korisnih u praksi je i problem pokrivanja lokacijama iz skupa (Location set covering problem). Kada je ovaj problem u pitanju, cilj je pronaći minimalan broj objekata iz ponuđenog skupa, koji pokrivaju sve zahteve klijenata. Pri tome, ako je svakoj lokaciji dodeljena cena uspostavljanja, LSCP ima zadatak da pronađe objekte koji zadovoljavaju sve zahteve klijenata, a da pri tome cena uspostavljanja tih objekata bude minimalna. U realnim situacijama često se događa da ovakvi objekti nisu poželjni u okolini klijenata koje treba da ”pokriju”, tako da se u literaturi sreću i modeli kod kojih je, kao dodatni uslov, definisano minimalno rastojanje od najbližeg klijenta. Na kraju se ipak pokazalo da ovakvi - prošireni modeli LSCP-a imaju istu strukturu kao i originalni LSCP problem.

3.2 Hijerarhijski model

Hijerarhijska varijanta problema pokrivanja korisnika (HCLP) je veoma česta u realnim situacijama. Tipičan primer hijerarhijskog problema predstavljaju školski i zdravstveni slučaj. Na primer, objekti na nižem nivou, poput domova zdravlja, mogu pružiti samo osnovne zdravstvene usluge prvog nivoa. Bolnice, osim osnovnih usluga, mogu pružiti i specijalističke usluge višeg (drugog) nivoa. Ova hijerarhija je takva da objekti višeg nivoa mogu pružati usluge svog i nižih nivoa, dok obrnuto ne važi. Drugi primer HCLP-a je u visokom školstvu po Bolonjskom procesu, gde imamo strukovne škole i univerzitete. Strukovne škole pružaju praktičnu edukaciju, dok univerziteti mogu pružati i praktičnu i akademsku edukaciju. Treći primer se može uočiti u proizvodnji i prodaji. Objekte čine fabrike i prodavnice, a proizvod do korisnika može stići kupovinom u prodavnici ili kupovinom direktno iz fabrike.

Kao što se iz prethodnog vidi, dva nivoa hijerarhije su najčešće dovoljna za opisivanje realnih problema, a veoma su retki slučajevi kada je potrebno više nivoa hijerarhije.

U ovom poglavlju je formulisan ovaj problem kao i genetski algoritam za njegovo rešavanje. Rešavani problem je NP – težak, jer je generalizacija veoma poznatog problema p-medijane.

3.3 Matematička formulacija problema

U cilju preciznog definisanja problema korišćena je formulacija celobrojnog linearnog programiranja iz [Gal96]. Neka je R_1 poluprečnik pokrivanja za usluge nižeg (prvog) nivoa koje pružaju objekti prvog nivoa, a T_1 poluprečnik pokrivanja kojim objekti drugog nivoa pružaju uslugu prvog nivoa. U principu, ovi poluprečnici mogu biti jednaki, ali u praksi klijenti su spremni da pređu i malo veće rastojanje da bi dobili kompletniju uslugu. Zbog toga je u ovom modelu ispoštovan uslov da je $T_1 > R_1$. Na kraju, neka je R_2 poluprečnik kojim objekti višeg nivoa pokrivaju klijente uslugom višeg (drugog) nivoa. Pošto jedino objekti drugog nivoa pružaju usluge višeg nivoa, realno je očekivati da su klijenti spremni da pređu i veća rastojanja da bi dobili uslugu višeg nivoa. Iz gore navedenih razloga u HCLP modelu važi nejednakost: $R_2 > T_1 > R_1$.

Model celobrojnog linearnog programiranja za HCLP je definisan na sledeći način:

Maksimizovati sumu:

$$\max\left\{\sum_{j\in J} f_j x_j\right\}, \tag{3.1}$$

tako da važi:

$$\sum_{i\in I} a_{ij} y_i + \sum_{i\in I} b_{ij} z_i - x_j \geq 0,\ j \in J, \tag{3.2}$$

$$\sum_{i\in I} c_{ij} z_i - x_j \geq 0,\ j \in J, \tag{3.3}$$

$$\sum_{i\in I} y_i = p, \tag{3.4}$$

$$\sum_{i \in I} z_i = q, \tag{3.5}$$

$$x_j \in \{0,1\}, j \in J, \tag{3.6}$$

$$y_i, z_i \in \{0,1\}, i \in I. \tag{3.7}$$

Pri čemu je:

- $J = \{1, 2, \ldots, m\}$ - skup područja za pokrivanje,
- $I = \{1, 2, \ldots, n\}$ - skup potencijalnih objekata,
- f_j populacija područja j,
- $a_{ij} = 1$ ako područje j može biti pokriveno uslugom prvog nivoa (na distanci R_1) od strane objekta nižeg nivoa lociranog u $i \in I$ ($a_{ij} = 0$ u suprotnom),
- $b_{ij} = 1$ ako područje j može biti pokriveno uslugom prvog nivoa (na distanci T_1) od strane objekta višeg nivoa lociranog u $i \in I$ ($b_{ij} = 0$ u suprotnom),
- $c_{ij} = 1$ ako područje j može biti pokriveno uslugom drugog nivoa (na distanci R_2) od strane objekta višeg nivoa lociranog u $i \in I$ ($c_{ij} = 0$ u suprotnom),
- p – broj objekata nižeg nivoa koje je potrebno locirati,
- q - broj objekata višeg nivoa koje je potrebno locirati.

Promenjive x_j , y_i i z_i su binarne i imaju vrednosti:

- $x_j = 1$ ako je područje j pokriveno ($x_j = 0$ inače),
- $y_i = 1$ označava da je objekat nižeg nivoa postavljen na lokaciju $i \in I$ ($y_i = 0$ u suprotnom),
- $z_i = 1$ ako je objekat višeg nivoa postavljen na lokaciju $i \in I$ ($z_i = 0$ u suprotnom).

Formula (3.1) predstavlja funkciju cilja, odnosno zadatak je maksimizovati pokrivenost populacije uslugama prvog i drugog nivoa. Uslov (3.2) obezbeđuje da područje $j \in J$ bude pokriveno bar jednim uspostavljenim objektom koji

pruža uslugu prvog nivoa. U tom slučaju je objekat nižeg nivoa na rastojanju manjem od R_1, ili objekat višeg nivoa na rastojanju manjem od T_1. Uslov (3.3) osigurava da područje $j \in J$ bude pokriveno uslugom drugog nivoa, odnosno da se na rastojanju manjem od R_2 nalazi bar jedan uspostavljeni objekat višeg nivoa. Uslovi (3.4) i (3.5) garantuju da će biti uspostavljeno tačno p, odnosno q objekata prvog i drugog nivoa, respektivno. Na kraju, uslovi (3.6) i (3.7) definišu binarnu prirodu promenljivih x_j , y_i i z_i.

Primer 3.1: Neka je broj objekata $n = 5$, pri čemu se na prvom i drugom nivou može uspotaviti po jedna lokacija, odnosno $p = 1, q = 1$. Neka su poluprečnici $R_1 = 15, T_1 = 20, R_2 = 30.$ Za cene uspostavljanja objekata $f = [8\ 7\ 6\ 2\ 9]$ i matricu rastojanja između objekata d:

$$d = \begin{bmatrix} 0 & 15 & 22 & 17 & 37 \\ 15 & 0 & 66 & 43 & 29 \\ 22 & 66 & 0 & 79 & 12 \\ 17 & 43 & 79 & 0 & 42 \\ 37 & 29 & 12 & 42 & 0 \end{bmatrix},$$

maksimalna vrednost funkcije cilja je 24. Na nižem nivou uspostavljena je treća potencijalna lokacija ($y_3 = 1$), dok je na drugom nivou uspostavljen drugi potencijalni objekat ($z_2 = 1$). Pri tome, izabrani objekti pokrivaju područja 1, 2, 5 ($x_1 = x_2 = x_5 = 1$) čija je ukupna populacija 8 + 7 + 9 = 24. Optimalnost rešenja je verifikovana uz pomoć programa za programski paket CPLEX ([Cpl08]), koji je implementiran na osnovu modela (3.1) - (3.7).

3.4 Detalji genetskog algoritma

Pri rešavanju problema hijerarhijskog pokrivanja korisnika korišćeno je binarno kodiranje. Svako potencijalno rešenje predstavljeno je binarnim vektorom dimenzije $2n$. Binarne vrednosti na neparnim pozicijama odnose se na potencijalne lokacije na nižem nivou, dok se vrednosti na parnim pozicijama odnose na potencijalne lokacije na višem nivou. Ukoliko je neka od potencijalnih lokacija uspostavljena, tada odgovarajuća vrednost u binarnom vektoru, koji predstavlja rešenje, treba da bude 1, odnosno 0 ukoliko lokacija nije uspostavljena.

Primer 3.2: Binarni kod koji predstavlja optimalno rešenje primera 3.1 je: 00 01 10 00 00. Pošto je $n = 5$, binarni kod je dužine 10 i uspostavljene su treća lokacija sa prvog i druga lokacija sa drugog nivoa.

Sa ovakvim načinom kodiranja promenljive y_i i z_i se dobijaju direktno iz genetskog koda. Računaju se sume iz uslova (3.2) i (3.3) i ako su obe veće ili jednake 1, promenljiva x_j dobija vrednost 1, a u suprotnom 0. Na taj način su po definiciji zadovoljeni svi uslovi, osim uslova (3.4) i (3.5). Da bi i ovi uslovi bili ispunjeni, potrebno je raditi sa jedinkama koje imaju p uspostavljenih potencijalnih lokacija na nižem i q uspostavljenih lokacija na višem nivou.

U želji da se na početku dobije što više korektnih jedinki, verovatnoća generisanja 1, na neparnim pozicijama u binarnom vektoru je p/n, a na parnim pozicijama q/n. Jedinke koje nemaju odgovarajući broj (p) jedinica na neparnim pozicijama (k_p - broj jedinica jedinke na neparnim pozicijama) modifikuju se tako što se sa kraja jedinke oduzme, odnosno doda ako je potrebno $| p - k_p |$ jedinica na neparnim pozicijama. Isto se postupa ukoliko na parnim pozicijama nema tačno q jedinica.

Ovakvom modifikacijom zadovoljeni su uslovi (3.4) i (3.5) kojima se zahteva da na prvom nivou bude uspostavljeno tačno p, a na drugom nivou tačno q lokacija. Opisani postupak primenjuje se samo na jedinke u prvoj - slučajno generisanoj populaciji, jer su genetski operatori, koji će kasnije biti opisani, napravljeni tako da čuvaju korektnost jedinki.

Pojavljivanje nekorektnih jedinki je bio ograničavajući faktor pri korišćenju binarnog kodiranja kod uobičajenih genetskih algoritama. U ovoj implementaciji ovaj problem je uspešno prevaziđen opisanim metodama, što se može videti i iz eksperimentalno dobijenih rezultata koji će kasnije biti prikazani.

Lema 3.1 Složenost izračunavanja funkcije cilja predloženog genetskog algoritma za rešavanje problema hijerarhijskog pokrivanja korisnika je $O(n \cdot m)$.

Dokaz

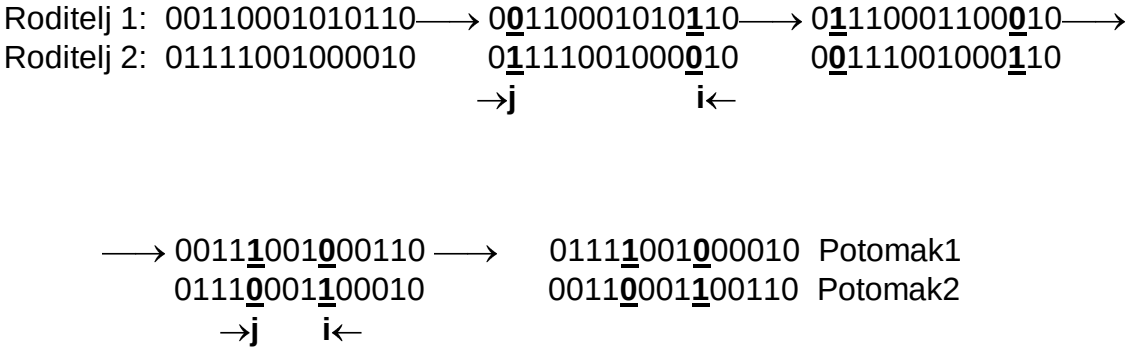
Složenost pretvaranja genetskog koda u parametre problema je $O(n)$, zato što toliko ima članova niza y_i i z_i. To je složenost u svim generacijama osim u prvoj kada je složenost dodatno uvećana za $O(p + q)$, koliko je u najgorem slučaju potrebno za popravke nekorektnih jedinki. Naime, ukoliko jedinka nema tačno p jedinica na parnim, odnosno q na neparnim pozicijama potrebno je slučajno odabrati pozicije koje će biti pretvorene u jedinice. Jasno je da je za ovakvu modifikaciju u najgorem slučaju (kada jedinka nema ni jednu jedinicu), potrebno $p + q$ koraka.

Složenost izračunavanja vrednosti funkcije cilja iz parametara problema je *O(n·m),* jer je potrebno dvostrukim for ciklusom proveriti pokrivenost područja uslugama prvog i drugog nivoa. Tačnije, za svako područje (kojih ima *n*) je potrebno proveriti da li postoje objekti (kojih ima n) koji mu pružaju usluge prvog i drugog nivoa, odnosno da li je: $(a_{i,j} \cdot y_i + a_{i,j} \cdot z_i) \cdot c_{i,j} \cdot z_i > 0$.

Iz prethodnog sledi da je ukupna složenost izračunavanja funkcije cilja *O(n·m).* □

U implementiranom algoritmu korišćena je Fino gradirana turnirska selekcija, opisana u 2.2.2 Realni parametar F_{tour} , koji predstavlja željenu srednju veličina turnira, ima vrednost 5.4.

Standardni operator ukrštanja na slučajan način bira pozicije u genetskom kodu dva roditelja i vrši razmenu genetskog materijala sa fiksiranih pozicija proizvodeći na taj način dva potomka. Ovakav operator ukrštanja nije pogodan za način kodiranja primenjen pri rešavanju HCLP-a, jer može proizvesti nekorektne jedinke (broj uspostavljenih lokacija na nižem nivou različit od *p* ili na višem nivou od *q*), čak i u slučaju da oba roditelja imaju tačno *p* ondnosno *q* jedinica na neparnim, odnosno parnim bitovima genetskog koda. Iz tog razloga predloženi genetski algoritam koristi modifikovani operator ukrštanja, koji čuva broj uspostavljenih lokacija na oba nivoa. Ukrštanje je realizovano tako da se paralelno sa oba kraja genetskog koda dva roditelja pronalaze neparne/parne pozicije *i,j* sa komplementarnim vrednostima bitova, kao što se može videti na slici 3.1. Zatim se vrši zamena (invertovanje) bitova na pronađenim pozicijama. Dati postupak se ponavlja sve dok je $j \geq i$.



Slika 3.1 Modifikovani operator ukrštanja

Operator mutacije sa zaleđenim bitovima, koji je primenjen u prethodm primeru, je modifikovan jer je jasno da mutacija mora da bude kontrolisana kako bi se očuvala validnost jedinke. Posle svake mutacije proverava se broj jedinica na parnim i neparnim pozicijama pa se u slučaju da ih nema p odnosno q vrše prinudne mutacije, odnosno promene slučajno izabranih bitova tako da jedinka postane validna.

Početna populacija je slučajno generisana, a veličina populacije je $N_{pop} = 150$ jedinki. Pri rešavanju HCLP je primenjena elitistička strategija tako da $N_{elit} = 100$ jedinki sa najboljom funkcijom prilagođenosti direktno prelazi u sledeću generaciju, dok se ostalih 50 jedinki dobija primeniom genetskih operatora.

3.5 Eksperimentalni rezultati

Implementirani gentski algoritam je testiran na slučajno generisanim instancama različitih dimenzija. Pri tome, pretpostavljeno je da se potencijalne lokacije nalaze u područjima koja treba da budu pokrivena. To je realno stanje koje odgovara situaciji za smeštanje bolnica ili vatrogasnih stanica. Praktično, uzeto je da je $J = I = \{1, 2, \ldots, n\}$.

U želji da se provere rezultati implementiranog genetskog algoritma, na osnovu matematičkog modela (3.1) – (3.7) je implementiran program za programiski paket CPLEX. Genetski algoritam je na testiranim instancama dostigao iste vrednosti kao i CPLEX program, a kako je vreme izvršavanja bilo veoma kratko u oba slučaja, podaci o vremenima izvršavanja nisu prezentovani.

Ponašanje genetskog algoritma testirano je na dve specijalne instance dimenzije 20. Prva instanca sadrži 20 lokacija koje se nalaze u kvadratu stranice 70, a rezultati testiranja ove instance su prikazani u primeru 3.3. Kod druge instance vodilo se računa da lokacije budu grupisane u tri različita područja. Rezultati testiranja ovakve instance su prikazani u primeru 3.4. Primer 3.5 prikazuje rezultate testiranja genetskog algoritma sa različitim parametrima. Na kraju, u primeru 3.6 predstavljeni su rezultati implementiranog algoritma na instancama različitih dimenzija.

Primer 3.3.: Za $n = 20$, $R_1 = 10$, $T_1 = 12$, $R_2 = 22$, fiksne troškove uspostavaljnja lokacija $f = [32\ 36\ 17\ 11\ 15\ 38\ 35\ 36\ 17\ 21\ 13\ 39\ 23\ 33\ 34\ 36\ 16\ 20\ 21\ 15]$ i matricu rastojanja d:

$$d = \begin{bmatrix}
0 & 6 & 51 & 66 & 14 & 30 & 41 & 58 & 23 & 42 & 71 & 65 & 43 & 46 & 62 & 51 & 67 & 84 & 58 & 71 \\
6 & 0 & 45 & 60 & 10 & 24 & 35 & 52 & 18 & 36 & 64 & 59 & 40 & 41 & 56 & 47 & 62 & 78 & 56 & 66 \\
51 & 45 & 0 & 15 & 45 & 24 & 16 & 14 & 41 & 27 & 27 & 30 & 62 & 47 & 42 & 58 & 52 & 55 & 78 & 62 \\
66 & 60 & 15 & 0 & 59 & 38 & 28 & 13 & 54 & 36 & 18 & 27 & 72 & 54 & 43 & 65 & 53 & 50 & 86 & 64 \\
14 & 10 & 45 & 59 & 0 & 21 & 32 & 49 & 9 & 29 & 60 & 53 & 30 & 32 & 48 & 37 & 53 & 71 & 47 & 56 \\
30 & 24 & 24 & 38 & 21 & 0 & 11 & 28 & 17 & 15 & 41 & 36 & 40 & 30 & 37 & 40 & 45 & 57 & 57 & 52 \\
41 & 35 & 16 & 28 & 32 & 11 & 0 & 17 & 26 & 12 & 29 & 25 & 46 & 31 & 31 & 42 & 40 & 49 & 67 & 49 \\
58 & 52 & 14 & 13 & 49 & 28 & 17 & 0 & 43 & 23 & 14 & 16 & 59 & 41 & 31 & 52 & 41 & 41 & 73 & 51 \\
23 & 18 & 41 & 54 & 9 & 17 & 26 & 43 & 0 & 21 & 53 & 45 & 25 & 23 & 39 & 29 & 44 & 62 & 42 & 48 \\
42 & 36 & 27 & 36 & 29 & 15 & 12 & 23 & 21 & 0 & 32 & 24 & 36 & 20 & 22 & 31 & 30 & 43 & 51 & 38 \\
71 & 64 & 27 & 18 & 60 & 41 & 29 & 14 & 53 & 32 & 0 & 12 & 65 & 46 & 30 & 56 & 39 & 32 & 78 & 49 \\
65 & 59 & 30 & 27 & 53 & 36 & 25 & 16 & 45 & 24 & 12 & 0 & 54 & 35 & 18 & 44 & 27 & 25 & 66 & 38 \\
43 & 40 & 62 & 72 & 30 & 40 & 46 & 59 & 25 & 36 & 65 & 54 & 0 & 19 & 40 & 14 & 39 & 61 & 17 & 36 \\
46 & 41 & 47 & 54 & 32 & 30 & 31 & 41 & 23 & 20 & 46 & 35 & 19 & 0 & 21 & 11 & 22 & 43 & 32 & 25 \\
62 & 56 & 42 & 43 & 48 & 37 & 31 & 31 & 39 & 22 & 30 & 18 & 40 & 21 & 0 & 28 & 10 & 23 & 49 & 21 \\
51 & 47 & 58 & 65 & 37 & 40 & 42 & 52 & 29 & 31 & 56 & 44 & 14 & 11 & 28 & 0 & 25 & 47 & 22 & 22 \\
67 & 62 & 52 & 53 & 53 & 45 & 40 & 41 & 44 & 30 & 39 & 27 & 39 & 22 & 10 & 25 & 0 & 22 & 45 & 11 \\
84 & 78 & 55 & 50 & 71 & 57 & 49 & 41 & 62 & 43 & 32 & 25 & 61 & 43 & 23 & 47 & 22 & 0 & 66 & 29 \\
58 & 56 & 78 & 86 & 47 & 57 & 67 & 73 & 42 & 51 & 78 & 66 & 17 & 32 & 49 & 22 & 45 & 66 & 0 & 38 \\
71 & 66 & 62 & 64 & 56 & 52 & 49 & 51 & 48 & 38 & 49 & 38 & 36 & 25 & 21 & 22 & 11 & 29 & 38 & 0
\end{bmatrix}$$

Genetski algoritam je za parametre *p = 3, q = 3,* dobio maksimalnu vrednost funkcije cilja čija je vrednost 355. Na nižem (prvom) nivou uspostavljene su potencijalne lokacije 8,13 i 15, a na drugom nivou su uspostavljene potencijalne lokacija 2,7 i 14. Tim izborom su pokrivena područja 1, 2, 5, 6, 7, 8, 10, 13, 14, 15, 16 i 17.

Uključivanjem više potenciljanih lokacija pokriva se i više područja, tako da je za parametre p = 6, q = 5 maksimalna vrednost funkcije cilja 497. Na nižem nivou uspostavljene su potencijalne lokacije 2, 3, 8, 13, 18 i 19, a na višem 5, 7, 11, 16 i 17. Pri tome su pokrivena sva područja osim područja 4. Dopuštanjem da se uspostavi još jedna lokacija na nižem nivou, odnosno za parametre p = 7, q = 5, maksimalna vrednost funkcije cilja je 508 i time se pokrivaju sva područja. Na prvom nivou uspostavljene su potencijalne lokacije 1, 3, 4, 8, 13, 18 i 19, a na višem nivou 5, 7, 11, 16 i 17. Iz ovih primera se vidi da se genetski algoritam dobro ponaša, kako u graničnim tako i u normalnim situacijama.

Primer 3.4: Za *n = 20, R_1 = 10, T_1 = 12, R_2 = 22, f = [32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15]* i matricu rastojanja *d*:

$$d = \begin{bmatrix}
0 & 12 & 13 & 19 & 23 & 26 & 63 & 56 & 62 & 55 & 60 & 47 & 52 & 51 & 62 & 69 & 60 & 63 & 65 & 75 \\
12 & 0 & 10 & 15 & 12 & 19 & 60 & 50 & 56 & 45 & 52 & 35 & 40 & 39 & 52 & 60 & 48 & 52 & 54 & 64 \\
13 & 10 & 0 & 7 & 14 & 13 & 51 & 44 & 49 & 42 & 47 & 41 & 44 & 40 & 50 & 56 & 52 & 53 & 54 & 64 \\
19 & 15 & 7 & 0 & 14 & 8 & 45 & 37 & 43 & 36 & 40 & 40 & 42 & 37 & 44 & 50 & 50 & 50 & 49 & 59 \\
23 & 12 & 14 & 14 & 0 & 12 & 52 & 40 & 47 & 34 & 41 & 27 & 30 & 27 & 40 & 49 & 38 & 40 & 42 & 51 \\
26 & 19 & 13 & 8 & 12 & 0 & 41 & 31 & 37 & 29 & 33 & 35 & 36 & 30 & 37 & 42 & 44 & 43 & 41 & 52 \\
63 & 60 & 51 & 45 & 52 & 41 & 0 & 17 & 9 & 32 & 24 & 67 & 62 & 52 & 38 & 29 & 68 & 59 & 48 & 58 \\
56 & 50 & 44 & 37 & 40 & 31 & 17 & 0 & 9 & 15 & 8 & 51 & 45 & 35 & 21 & 16 & 51 & 42 & 32 & 42 \\
62 & 56 & 49 & 43 & 47 & 37 & 9 & 9 & 0 & 23 & 15 & 59 & 54 & 44 & 28 & 20 & 59 & 50 & 39 & 49 \\
55 & 45 & 42 & 36 & 34 & 29 & 32 & 15 & 23 & 0 & 9 & 37 & 31 & 21 & 9 & 15 & 36 & 28 & 18 & 29 \\
60 & 52 & 47 & 40 & 41 & 33 & 24 & 8 & 15 & 9 & 0 & 47 & 40 & 30 & 14 & 9 & 45 & 36 & 24 & 34 \\
47 & 35 & 41 & 40 & 27 & 35 & 67 & 51 & 59 & 37 & 47 & 0 & 9 & 17 & 38 & 51 & 14 & 23 & 33 & 37 \\
52 & 40 & 44 & 42 & 30 & 36 & 62 & 45 & 54 & 31 & 40 & 9 & 0 & 10 & 30 & 43 & 8 & 13 & 24 & 28 \\
51 & 39 & 40 & 37 & 27 & 30 & 52 & 35 & 44 & 21 & 30 & 17 & 10 & 0 & 21 & 34 & 16 & 13 & 17 & 25 \\
62 & 52 & 50 & 44 & 40 & 37 & 38 & 21 & 28 & 9 & 14 & 38 & 30 & 21 & 0 & 13 & 34 & 23 & 11 & 21 \\
69 & 60 & 56 & 50 & 49 & 42 & 29 & 16 & 20 & 15 & 9 & 51 & 43 & 34 & 13 & 0 & 47 & 36 & 23 & 30 \\
60 & 48 & 52 & 50 & 38 & 44 & 68 & 51 & 59 & 36 & 45 & 14 & 8 & 16 & 34 & 47 & 0 & 12 & 25 & 26 \\
63 & 52 & 53 & 50 & 40 & 43 & 59 & 42 & 50 & 28 & 36 & 23 & 13 & 13 & 23 & 36 & 12 & 0 & 13 & 14 \\
65 & 54 & 54 & 49 & 42 & 41 & 48 & 32 & 39 & 18 & 24 & 33 & 24 & 17 & 11 & 23 & 25 & 13 & 0 & 11 \\
75 & 64 & 64 & 59 & 51 & 52 & 58 & 42 & 49 & 29 & 34 & 37 & 28 & 25 & 21 & 30 & 26 & 14 & 11 & 0
\end{bmatrix}$$

Implementirani algoritam je za *p = 3, q = 3,* dobio maksimalnu vrednost funkcije cilja 441. Na nižem (prvom) nivou uspostavljene su potencijalne lokacije 4,11 i 13, a na drugom nivou su uspostavljene potencijalne lokacije 2,15 i 17. Tim izborom su pokrivena sva područja osim 7, 9 i 20. Poboljšanjem usluge prvog nivoa, odnosno za p = 6, q = 3, dobija se maksimalna vrednost funkcije cilja čija je vrednost 493. Uspostavljene su lokacije 6, 13, 15, 16, 18 i 19 na prvom i 2, 9 i 14 na drugom nivou. Ovim poboljšanjem pokrivena su sva područja osim područja 20. Sva područja su grupisana u tri okoline tako da je područje 20 izolovano, ondnosno nalazi se na rastojanju većem od 22 od ostalih regiona. Da bi i ovo područje bilo pokriveno potrebna je još jedna lokacija drugog nivoa koja bi ga pokrila. Dakle, iz prethodnog je jasno da su za pokrivanje svih područja potrebni parametri: *p = 3, q = 4.* U tom slučaju maksimalna vrednost funkcije cilja je 508, uspostavljene su lokacije 4, 11 i 13 na prvom nivou i 2, 9, 17 i 19 na drugom nivou. Pri tome su pokrivena sva područja.

Primer 3.5: Korišćenjem instance iz primera 3.4 testirano je i ponašanje genetskog algoritma za različite veličine početne populacije i procenat elitnih jedinki. Rezultati ovih testiranja su prikazani u Tabelama 3.1 i 3.2.

Tabela 3.1 Broj generacija u zavisnoti od veličine početne populacije

Inicijalna populacija	**20**	**50**	**150**	**400**	**p**	**q**
Broj generacija	414	58	8	6	3	3
	259	91	54	45	6	5
	1558	745	840	363	7	5

Kao što rezultati iz Tabele 3.1 pokazuju, populacija od 150 jedinki ima najbolji odnos broja generacija i veličine inicijalnih populacija u sva tri slučaja, tako da je opravdano izabrana za testiranje genetskog algoritma.

Tabela 3.2 Broj generacija u zavisnosti od procenta elitnih jedinki

Procenat elitnih jedinki	**0.66%**	**10%**	**30%**	**70%**	**p**	**q**
Broj generacija	8	7	4	12	3	3
	54	48	55	37	6	5
	840	23	65	60	7	5

Imajući u vidu rezultate iz Tabele 3.2 jasno je da je najprihvatljivija vrednost za procenat elitnih jedinki koje prelaze u sledeću generaciju 70%. Ta vrednost, u kombinaciji sa najboljom veličinom inicijalne populacije (150), povlači činjenicu da su pri testiranju implementiranog genetskog algoritma korišćeni najbolji parametri – populacija od 150 jedinki, od kojih je 100 jedinki elitno (približno 70%).

Primer 3.6: U ovom primeru su prikazani rezultati genetskog algoritma na slučajno generisanim instancama različitih dimenzija. Instance zbog svoje veličine neće biti prikazane u ovom radu, ali se mogu naći na interntet adresi: http://www.matf.bg.ac.yu/~maricm/instances/hclp/.

Tabela 3.3 prikazuje dimenzije i vrednosti fiksnih troškova za svaku od testiranih instaci koje su bile iste u toku testiranja. Pri svim testiranjima, poluprečnici su imali sledeće veličine: $R_1 = 15$, $T_1 = 20$, $R_2 = 30$.

Tabela 3.3 Dimenzije instanci i vrednosti fiksnih troškova

Ime instance	N	Fiksni troškovi
Hclp10.txt	10	[12 17 13 9 11 8 19 14 12 10]
Hclp15.txt	15	[22 13 21 17 14 15 11 9 31 22 25 16 19 12 18]
Hclp20.txt	20	[32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15]
Hclp25.txt	25	[32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15 13 25 24 33 27]
Hclp30.txt	30	[32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15 13 25 24 33 27]
Hclp40.txt	40	[32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15 13 25 24 33 27 33 31 35 17 13 38 23 15 24 28 38 12 15 18 24]
Hclp45.txt	45	[32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15 13 25 24 33 27 33 31 35 17 13 38 23 15 24 28 38 12 15 18 24 16 31 39 39 30]
Hclp50.txt	50	[20 35 28 25 17 15 29 18 15 20 15 20 36 17 16 11 35 11 18 15 18 15 21 18 10 40 14 20 30 40 37 10 34 35 10 31 22 29 18 12 17 20 28 16 26 20 24 30 33 12]

Tabele 3.4 i 3.5 prikazuju rezultate dobijene pimenom genetskog algoritma. Za svaku instancu prvo su prikazane vrednosti parametara *p* i *q,* zatim maksimalna vrednost funkcije cilja (koja je proverena i CPLEX programskim paketom), sledeće dve kolone prikazuju uspostavljene lokacije na prvom i drugom nivou, a u poslednjoj koloni se nalaze informacije o pokrivenim područjima ovakvim izborom lokacija.

Tabela 3.4 Rezultati

Ime instance	p	q	Max	Lokacije na I nivou	Lokacije na II nivou	Pokrivena područja
Hclp10	1	1	85	1	7	1, 2, 4, 8, 9
Hclp10	2	2	117	3, 10	5, 7	sva osim 6
Hclp10	2	3	265	5, 9	1, 3, 13	Sva
Hclp15	1	1	141	9	7	1, 2, 4, 7, 9, 10, 11
Hclp15	2	2	224	1, 3	10, 13	Sva osim 5,8 i 15
Hclp15	2	3	265	5,9	1, 3, 13	Sva
Hclp20	1	1	210	14	19	8, 9, 11, 14, 15, 16, 18, 19
Hclp20	2	2	224	7, 14	2, 19	1, 2, 7, 8, 9, 11, 13, 14, 15, 16, 18, 19, 20
Hclp20	2	3	427	10, 14	2, 7, 19	1, 2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
Hclp25	1	1	276	20	19	2, 9, 11, 12, 13, 14, 18, 21, 22, 23 i 24.
Hclp25	2	2	466	17, 20	10, 11	2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 21, 22, 23, 24 i 25
Hclp25	2	3	556	2, 18	6, 11, 15	2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23, 24 i 25
Hclp30	1	1	336	16	14	2, 6, 9, 14, 15, 16, 22, 23, 25, 27, 28.
Hclp30	2	2	575	5, 9	14, 26	2, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 18, 19, 21, 22, 24, 25, 26, 27, 28, 30
Hclp30	2	3	684	2, 18	10, 12, 14	2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27, 28, 29, 30

Tabela 3.5 Rezultati

Ime instance	p	q	Max	Lokacije na I nivou	Lokacije na II nivou	Pokrivena područja
Hclp40	1	1	452	9	32	1 9, 12, 14, 15, 16, 19, 23, 24, 25, 26, 31, 32, 34, 36
Hclp40	2	2	769	10, 32	5, 23	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 19, 22, 23, 24, 25, 26, 28, 29, 31, 32, 35, 36, 37, 38, 39
Hclp40	2	3	886	10, 27	5, 23, 32	Sva osim: 2, 18, 20, 30 i 40
Hclp45	1	1	521	45	7	2, 4, 7, 8, 10, 12, 13, 14, 15, 18, 20, 23, 24, 30, 34, 38, 42, 43, 44
Hclp45	2	2	830	27, 45	7, 22	2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 22, 23, 24, 27, 28, 30, 32, 34, 38, 39, 40, 42, 43, 44, 45
Hclp45	2	3	1009	16, 44	14, 30, 42	Sva osim: 5, 17, 24, 27, 37, 38, 39
Hclp50	1	1	409	30	44	6, 8, 10, 11, 17, 22, 27, 29, 31, 33, 34, 36, 44, 47, 48, 50
Hclp50	2	2	730	3, 19	44, 48	3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 22, 24, 26, 27, 29, 30, 31, 33, 34, 36, 39, 40, 43, 44, 47, 48, 49, 50
Hclp50	5	5	1106	37, 38, 39, 45, 46	9, 20, 30, 32,44	Sva

4 LOKACIJSKI PROBLEM SNABDEVAČA NEOGRANIČENOG KAPACITETA U VIŠE NIVOA

4.1 Pregled problema lokacije snabdevača

Lokacijski problem snabdevača neograničenog kapaciteta (The uncapacitated facility location problem - UFLP, u literaturi poznat i kao Uncapacitated warehouse location problem ili The simple plant location problem) je jedan od osnovnih i najviše rešavanih problema u teoriji lokacijskih problema. Zadatak je minimizovati sumu fiksnih troškova i troškova transporta, a pri tome ispoštovati zahteve skupa klijenata. Ovo se realizuje pravilnim odabirom potencijalnih lokacija iz skupa mogućih lokacija. Pregled problema ovakve vrste se može naći u radovima [Dre02, Klo04, ReV05, Vyg05].

Neki od uspešnih metoda za rešavanje UFLP, prezentovani u literaturi do sada, su : Genetski algoritmi [Fil00, Kra99b, Kra01a], metoda promenljivih okolina (variable neighborhood search) [Han07], tabu pretraživanje (tabu search) [Sun06], simulirano kaljenje (simulated annealing) [Yig03], Lagranževa relaksacija (Lagrangean relaxation) [Cor06], lepezasto izdvajanje (filter and fan) [Gre03, Gre06], hibridna heuristika sa višestrukim startovanjem (hybrid multistart heuristic) [Res06] itd. Takođe, postoje i metode za rešavanje nekih generalizacija osnovnog problema, kao što su: više-kriterijumski (multi-objective) UFLP [Med05, Vil06] i dinamički (dynamic) UFLP [Dia05a, Dia05b] .

Za razliku od osnovnog modela, hijerarhijski model je razmatran u samo nekoliko radova [Aar99, Age02, Age05, Bum01, Edw01, Zha06]. Međutim, sve metode opisane u ovim radovima, osim [Edw01], su teoretski i bez eksperimentalnih rezultata. U [Edw01] implementirana su četiri metoda za rešavanje MLUFLP, i to:

- Približni algoritam zasnovan na linearnoj relaksaciji sa faktorom 3 (Linear program solution rounding 3-approximation algorithm - MLRR), čiji su autori Aardal, Chudak and Shmoys [Aar99]. Faktor 3 označava da je rešenje najviše tri puta gore od optimalnog.

- Algoritam zasnovan na redukciji sa k nivoa na jedan nivo (Path reduction of the k-level facility location problem to a single-level problem - PR-RR), čiji su autori Chudak and Shmoys [Chu99].

- Algoritam lokalnog poboljšanja zasnovan na redukciji sa faktorom 3 (Local improvement 3-approximation algorithm for the path reduction - PR - LI), čiji su autori Charikar and Guha [Cha99].

- Algoritam najkraćeg puta sa računanjem troškova uspostavljanja lokacija (Facility cost oblivious shortest path algorithm - SP), čiji je autor Edwards [Edw01].

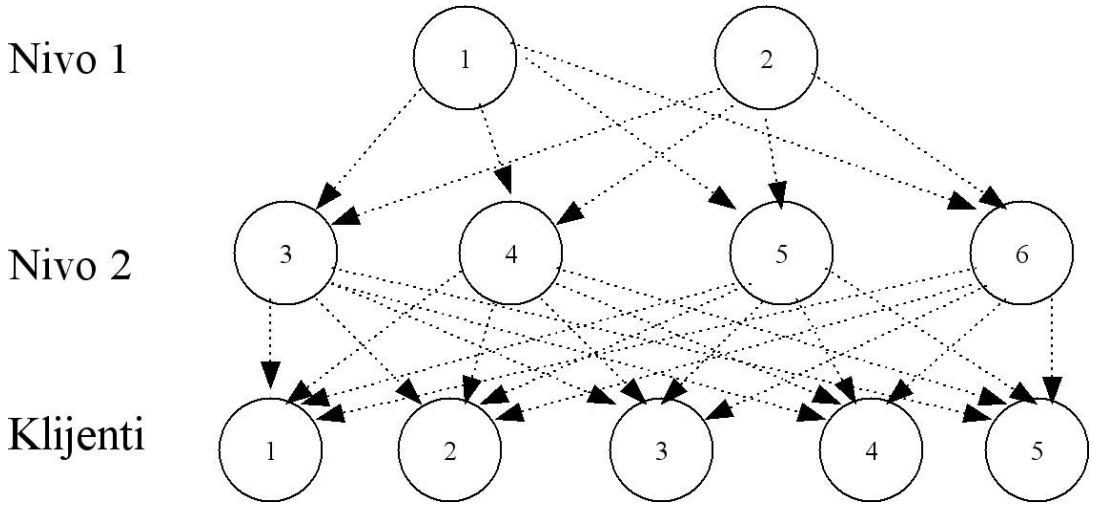
Prva tri algoritma su zasnovana na linearnoj relaksaciji modela (koji će biti opisan u narednom delu), sa veoma velikim brojem uslova i promenljivih. U svakom slučaju, sva tri algoritma mogu da rešavaju samo probleme malih dimenzija (najviše do 52 potencijalne lokacije). Relativna odstupanja u odnosu na donju granicu rešenja dobijena linearnom relaksacijom problema, su do 98.5% kod MLRR algoritma, do 3.3% za PR-RR metod i do 10.8% za PR-LI algoritam. Pri tome, vremena izvršavanja su do 813 sekundi za MLRR, do 183 sekundi za PR-RR algoritam, do 105 sekundi kada je u pitanju implementirani PR-LI metod. Relativna odstupanja su prihvatljiva za sve slučajeve, osim kada je u pitanju MLRR metod. Međutim, vremena izvršavanja su isuviše velika za male instance. U svakom slučaju, ni jedna od navedenih metoda nema mogućnost da rešava probleme većih dimenzija. Sa druge strane, SP metod dolazi do rešenja veoma brzo (do 0.07 sekundi), ali su odstupanja veoma velika, čak i do 732% na nekim instancama. Iz prethodnog se može zaključiti da ni jedan od ovih metoda nije pogodan za rad sa instancama srednjih i velikih dimenzija.

4.2 Matematička formulacija problema

Ulazni podaci za MLUFLP sastoje se od skupa potencijalnih lokacija F (čija je kardinalnost $|F| = m$) koje su podeljene u k nivoa označenih sa $F_1, \ldots, F_k$, skupa klijenata D ($|D| = n$), vrednosti fiksnih troškova f_i za uspostavljanje lokacije $i \in F$ i vrednosti transportnih troškova c_{ij} za svaki

i,j $\in F \cup D$. Na slici 4.1 prikazan je primer sa dva nivoa potencijalnih lokacija i pet klijenata, pri čemu se na prvom nivou nalaze dve a na drugom četiri potencijalne lokacije. Isprekidanim linijama su prikazane moguće veze između lokacija na prvom i drugom nivou i lokacija na drugom nivou i klijenata.

Korektna rešenja svakom klijentu dodeljuju niz od *k* potencijlanih lokacija, pri čemu su te lokacije na različitim nivoima, odnosno na svakom nivou bar po jedna. Optimalno rešenje podrazumeva da ukupna suma fiksnih troškova za uspostavljanje lokacija i troškova transporta bude minimalna. Troškovi transporta predstavljaju sumu troškova transporta od lokacije na prvom nivou preko svih nivoa do svakog od korisnika.



Slika 4.1 Potencijalne lokacije, klijenti i moguće veze

Lokacijski problem snabdevača neograničenog kapaciteta u više nivoa (MLUFLP) je NP- težak kao generalizacija problema UFLP (koji ima samo jedan nivo) za koji se u [Krp83] pokazalo da je NP- težak.

Pri rešavanju lokacijskog problema snabdevača neograničenog kapaciteta u više nivoa korišćena je model - formulacija celobrojnog linearnog programiranja preuzeta iz [Edw01]. Za svakog klijenta $j \in D$ je definisana dozvoljena putanja njegovog snabdevanja, počev od lokacija na prvom pa do lokacija na poslednjem nivou F_1. Skup svih dozvoljenih putanja označava se sa $P = F_k \times F_{k-1} \times \ldots \times F_1$ a troškovi transporta za snabdevanje klijenta putanjom $p = (i_k, \ldots, i_1)$ računaju se kao suma $c_{pj} = c_{ji_k} + c_{i_k i_{k-1}} + \ldots + c_{i_2 i_1}$. Promenljiva y_i ima vrednost 1 ukoliko je lokacija i uspostavljena, a 0 ukoliko lokacija nije

uspostavljena. Slično, promenljiva x_{pj} ima vrednost 1 ukoliko se klijent j snabdeva korišćenjem putanje p, u suprotnom x_{pj} ima vrednost 0. Korišćenjem ovako uvedene notacije matematički model za rešavanje MLUFLP se može definisati na sledeći način:

$$\min\left(\sum_{i\in F} f_i y_i + \sum_{p\in P}\sum_{j\in D} c_{pj} x_{pj}\right), \tag{4.1}$$

$$\sum_{p\in P} x_{pj} = 1, \tag{4.2}$$

$$\sum_{i\in p} x_{pj} \le y_i, \tag{4.3}$$

$$x_{pj} \in 0,1, \tag{4.4}$$

$$y_i \in 0,1. \tag{4.5}$$

Funkcija cilja (4.1) minimizuje sumu fiksnih troškova uspostavljenih lokacija i troškova transporta. Uslov (4.2) obezbeđuje da svaki klijent bude pokriven odgovarajućom putanjom, dok uslov (4.3) pruža garancije da će svaka uspostavljena lokacija biti korišćena od strane bar jednog klijenta. Uslovima (4.4) i (4.5) definisana je binarna priroda promenljivih x_{pj} i y_i.

Primer 4.1: Neka su u Tabeli 4.1 date vrednosti fiksnih troškova za uspostavljanje lokacija. Tabele 4.2 i 4.3 sadrže rastojanja između lokacija na prvom i drugom nivou, odnosno rastojanja od klijenata do lokacija na drugom nivou.

Tabela 4.1 Fiksni troškovi

Lokacije	f1	f2	f3	f4	f5	f6
Fiksni troškovi	20	20	10	10	10	10

Tabela 4.2 Rastojanja između lokacija na prvom i drugom nivou

	f1	f2
f3	12	13
f4	11	15
f5	16	12
F6	12	15

Tabela 4.3 Rastojanja između klijenata i lokacija snabdevača

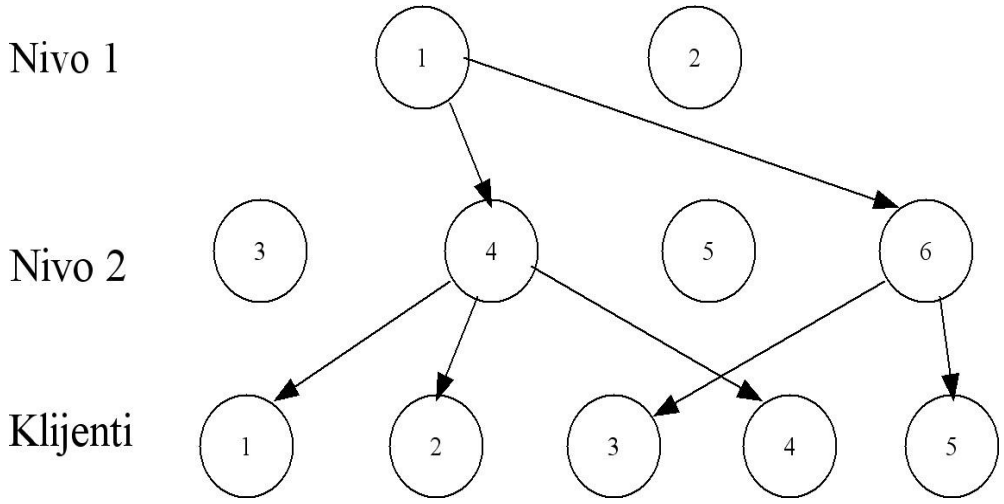
	f3	f4	f5	f6
Klijent 1	5	2	4	3
Klijent 2	4	1	6	8
Klijent 3	1	5	2	3
Klijent 4	8	1	5	1
Klijent 5	4	9	2	1

Za dobijanje optimalnog rešenja, ovako formulisanog problema, korišćen je program zasnovan na totalnoj enumeraciji koji će biti opisan u delu 4.5. Uspostavljene su lokacije *f1* na prvom i *f4* i *f6* na drugom nivou. Vrednost funkcije cilja optimalnog rešenja je *105*. Putanje do klijenata (nizovi lokacija za svakog klijenta) su prikazane u Tabeli 4.4.

Tabela 4.4 Putanje do klijenata

	Nivo 2	Nivo 1
Klijent 1	f4	f1
Klijent 2	f4	f1
Klijent 3	f6	f1
Klijent 4	f4	f1
Klijent 5	f6	f1

Takođe, na slici 4.2 mogu se videti uspostavljene veze optimalnog rešenja.



Slika 4.2 Veze između objekata i klijenata

4.3 Dinamičko programiranje

Dinamičko programiranje je veoma popularan metod za rešavanje problema kombinatorne optimizacije koji imaju optimalnu substrukturu rešenja. Optimalna substruktura predstavlja mogućnost da se za nalaženje optimalnog rešenja problema mogu koristiti optimalna rešenja subproblema. Polinomski broj stanja i koraka je veoma bitan za uspešnu implementaciju. Naravno, to je ispunjeno samo ukoliko je problem polinomske složenosti.

Osnovna ideja pri rešavanju MLUFLP-a je drastično smanjivanje broja potencijalnih putanja, čime bi se olakšalo računanje funkcije cilja. Postojeće metode razmatraju sve moguće putanje, što iziskuje veliku prostornu i vremensku složenost izračunavanja. Upravo, iz tog razloga one nisu pogodne za rešavanje problema većih dimenzija.

Neka je *FixMLUFLP* potproblem od *MLUFLP*, koji se iz njega dobija fiksiranjem uspostavljenih lokacija, uz uslov da na svakom nivou postoji bar jedna uspostavljena lokacija. *MLUFLP* je *NP*-težak, dok je *FixMLUFLP* polinomske složenosti. Ono što je značajno za ovaj rad je činjenica da *FixMLUFLP* ima svojstvo optimalne podstrukture, što će ovde biti dokazano.

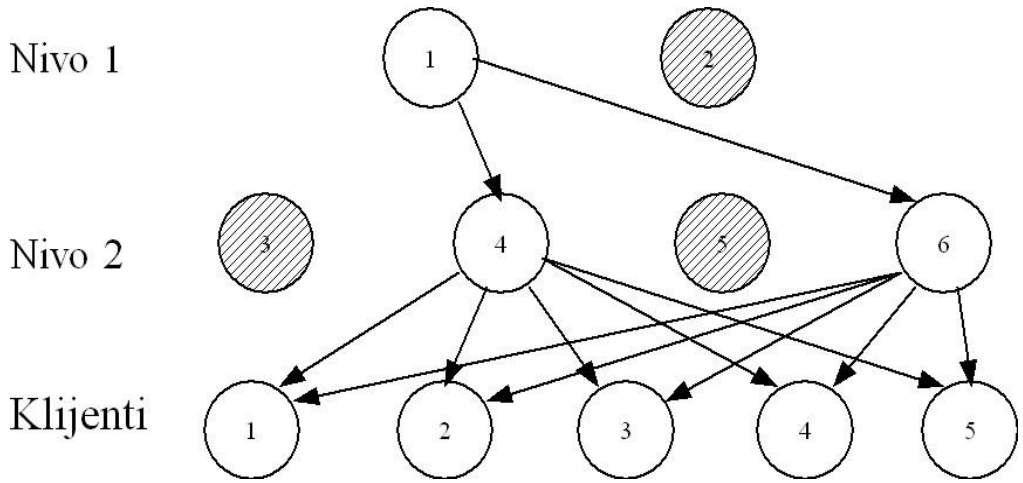
Lema 4.1 *FixMLUFLP* se može polinomski svesti na problem najkraćih puteva u direktnom acikličnom grafu (DAG).

Dokaz

Graf G = < V, E> se može konstruisati na sledeći način:

- Neka je *V = F u D.*
- Neka $(u,v) \in E, u,v \in F$ ako i samo ako $u \in F_l, v \in F_{l+1}$, $y_u = y_v = 1$ za $l \in \{1,2,...,k-1\}$,
- $(u,v) \in E, u \in F, v \in D$ ako i samo ako $u \in F_k, y_u = 1$.
- Težina grane *(u,v)* se definiše kao rastojanje između *u* i *v*.

Na slici 4.3 se može videti graf koji odgovara rešenju iz primera 4.1. Precrtane su lokacije koje se ne uspostavljaju, a prikazane su i sve moguće veze između čvorova. Konstruisani graf *G* je očigledno acikličan jer njegove grane povezuju lokacije na različitim nivoima, odnosno klijente sa lokacijama na prvom nivou. Kod grafa *G* za svakog klijenta postoji putanja od prvog nivoa do njega jer kod *FixMLUFLP* na svakom nivou postoji bar jedna uspostavljena lokacija, te su sve lokacije na podnivoima povezane sa nivoima iznad. Jasno je da za svakog klijenta *j* najkraća putanja od prvog nivoa do njega povlači $x_{pj} = 1$. Vrednosna funkcija *FixMLUFLP* se može računati sumiranjem težina ovih najkraćih puteva i dodavanjem sume fiksnih troškova uspostavljenih lokacija. Iz toga sledi da se *FixMLUFLP* polinomski može svesti na problem pronalaženja najkraćih puteva u acikličnom grafu. □



Slika 4.3 Usptostavljene lokacije i mogući putevi

Pošto se problem pronalaženja najkraćih puteva u acikličnom grafu optimalno može rešiti uz pomoć dinamičkog programiranja, jasno je da je dinamičko programiranje dobar metod za rešavanje i *FixMLUFLP*.

Dinamičko programiranje se u konkretnom slučaju može koristiti na sledeći način. Konstruiše se niz minimalnih troškova (*cs*), koji sadrži vrednosti ukupnih transportnih troškova za svaku uspostavljenu lokaciju od lokacije na prvom nivou. Ovom nizu se dodaju i klijenti kao lokacije *k+1* – vog nivoa, tako da niz *cs* ima *m+n* članova.

Na početku, niz *cs* se inicijalizuje tako da se na pozicijama koje odgovaraju lokacijama na prvom nivou postave 0, a na ostalim pozicijama dovoljno veliki broj (10^{30} se pokazao kao dobar izbor). Dalje se za sve nivoe *l = 2, 3, ..., k+1* i sve uspostavljene lokacije na tim nivoima $i \in F_l$ i $y_i = 1$, odgovarajući članovi niza *cs* rekurzivno računaju po formuli (4.6):

$$cs[i] = min_{k \in F_{l-1} \wedge y_k = 1}(cs[k] + c_{ik}) . \tag{4.6}$$

Na kraju, funkcija cilja FixMLUFLP se računa na sledeći način:

$$obj_{ind} = \sum_{i \in F} f_i y_i + \sum_{i=m+1}^{m+n} cs[i] . \tag{4.7}$$

Jasno je da je broj stanja (broj članova niza *cs*) jednak *m* + *n*. Složenost je $O(n * \max_i(num[i]))$, $1 \le i \le k$, pri čemu niz *num* sadrži informacije o broju lokacija na svakom nivou. U najgorem slučaju, kada je *k = 1*, složenost je $O(n^2)$.

4.4 Genetski algoritam za rešavanje MLUFLP

Pri rešavanju MLFULP korišćeno je binarno kodiranje jedinki populacije. Skup svih potencijalnih lokacija je predstavljen kao binarni niz dužine *m*. Jedinica na poziciji *i* označava da je *i*-ta potencijalna lokacija uspostavljena – y_i = 1, dok nula označava da *i*-ta lokacija nije uspostavljena - y_i = 0.

Fiksiranjem uspostavljenih potencijalnih lokacija dobijenih iz genetskog koda, računanje funkcije cilja se svodi na rešavanje *FixMLUFLP* problema.

Kako se u *FixMLUFLP* pretpostavlja da na svakom nivou postoji bar jedna uspostavljena lokacija, proverava se da li ovo svojstvo zadovoljava potencijalno rešenje dobijeno iz genetskog koda. Ukoliko ovaj binarni niz ima na svakom nivou bar po jednu uspostavljenu lokaciju, funkcija cilja se računa primenom dinamičkog programiranja koje je opisano u prethodnom odeljku. U suprotnom, rešenje nije validno pa se jedinka označava nekorektnom.

Primer 4.2 Ako je genetski kod 100101, to znači da su uspostavljene lokacije 1, 4 i 6, odnosno da je $y_1 = y_4 = y_6 = 1$, dok je $y_2 = y_3 = y_5 = 0$. Ukoliko pretpostavimo da je datim genetskim kodom predstavljeno jedno od rešenja primera 4.1, to znači da su uspostavljene lokacije: 1 na prvom nivou, 4 i 6 na drugom nivou, odnosno da ovaj genetski kod odgovara optimalnom rešenju primera 4.1.

Funkcija cilja korektne jedinke se računa u četiri koraka :

1. U prvom koraku očitavaju se vrednosti promenljivih y_i dobijene iz genetskog koda. Složenost ovog koraka je $O(m)$.

2. U drugom koraku se inicijalizuje niz minimalnih troškova *cs.* Kao što se može videti u prethodnom odeljku, niz cs čuva informacije o ukupnim transportnim troškovima za svaku lokaciju, osim lokacija na prvom nivou do kojih ne postoje transportni troškovi. Ukupni transportni troškovi predstavljaju ukupnu cenu transporta od prvog nivoa do odgovarajuće lokacije. U ovom koraku vrednosti članova niza *cs*, koje odgovaraju lokacijama na prvom nivou, se inicijalizuju na 0, dok ostali članovi niza dobijaju vrednost velike konstante INF = 10^{30}. Složenost ovog koraka je $O(n+m)$.

3. Minimalni troškovi transporta za svaku lokaciju ili klijenta se računaju primenom dinamičkog programiranja. Niz *cs* koji je inicijalizovan u prethodnom koraku se ažurira tako da je minimalna vrednost troškova transporta minimum sume transportnih troškova do lokacije na višem nivou i minimalnih troškova za tu lokaciju, kao što pokazuje formula (4.6). Slična procedura se primenjuje i za svakog klijenta – među uspostavljenim lokacijama na poslednjem nivou bira se ona čija je suma minimalnih troškova transporta i troškova transporta do odgovarajućeg klijenta, minimalna. Složenost ovog koraka, u najgorem slučaju kada je broj nivoa k = 1, je $O(n^2)$.

4. U poslednjem koraku računa se vrednost funkcije cilja, a složenost tog izračunavanja je $O(n+m)$.

Lema 4.2 Složenost izračunavanja funkcije cilja predloženog genetskog algoritma za rešavanje problema lokacije snabdevača bez ograničenja kapaciteta u više nivoa je $O(m+n^2)$.

Dokaz

Imajući u vidu gore navedene korake(1 - 4) za izračunavanje funkcije cilja, jasno je da je ukupna složenost izračunavanja funkcije cilja $O(m+n^2)$. □

Kada su genetski operatori u pitanju, za selekciju je iskorišćena fino gradirana turnirska selekcija sa vrednošću parametra selekcionog metoda 5.4, koja je opisana u sekciji 2.2.3. Korišćeno je jednopoziciono ukrštanje (sekcija 2.2.4) sa verovatnoćom ukrštanja od 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. Za operatora mutacije genetskog algoritma izabrana je prosta mutacija sa zaleđenim bitovima, koja je imala 2.5 puta veću verovatnoću mutacija na zaleđenim bitovima (videti sekciju 2.2.5).

Posebna pažnja je posvećena višestrukoj pojavi istih jedinki u populaciji (koje imaju isti genetski kod). Ovakve jedinke su suvišne, pa se njihov prolaz u sledeću generaciju sprečava time što se vrednost njihove funkcije prilagođenosti postavi na nulu. Ovakva modifikacija se izvršava na svim višestrukim pojavama jedinke, osim na prvoj koja se pojavljuje u populaciji i koja u trenutku pojavljivanja nema svoju kopiju. Takođe, jedinke koje nemaju isti genetski kod, ali imaju istu vrednost funkcije cilja, mogu preovladati populacijom i time umanjiti prisustvo drugih jedinki sa dobrim genetskim materijalom. Ovakva pojava se sprečava ograničavanjem broja pojavljivanja ovakvih jedinki na određeni (dozvoljeni) broj N_{ifc}. Primena ovakvih tehnika je veoma bitna za poboljšanje performansi genetskog algoritma, jer one omogućavaju raznovrsnost genetskog materijala i sprečavaju preuranjenu konvergenciju. Praktično, ovo se za svaku jedinku u populaciji postiže u dva koraka:

1. Proverava se da li je genetski kod jedinke isti kao genetski kod neke jedinke koja se već pojavila. Ukoliko postoji kopija jedinke, vrednost njene funkcije prilagođenosti se postavlja na nulu, u suprotnom se prelazi na korak 2.

2. Prebrojava se koliko već obrađenih jedinki ima istu vrednost funkcije cilja kao posmatrana jedinka (ako je njena vrednost različita od nule). Ukoliko je dobijeni broj veći ili jednak N_{ifc} vrednost funkcije prilagođenosti te jedinke se postavlja na nulu.

Pri rešavanju MLUFLP korišćena je veličina populacije od $N_{pop} = 150$ jedinki, koja je slučajno generisana u prvom koraku. Primenjena je elitistička strategija tako da $N_{elit} = 100$ jedinki sa najboljom funkcijom prilagođenosti direktno prelazi u sledeću generaciju, dok se ostalih 50 jedinki dobija primenom genetskih operatora. Dozvoljeni broj jedinki koje imaju različitie kodove a istu funkciju cilja je ograničen na $N_{ifc} = 40$.

4.5 Eksperimentalni rezultati i poređenja

Zbog potrebe da se verifikuju rešenja dobijena genetskim algoritmom na malim instancama implementirana je metoda totalne enumeracije. Ova tehnika jednostavno proverava sve podskupove skupa *F*, tražeći minimalnu funkciju cilja MLUFLP-a. Funkcija cilja se računa na isti način kao što je opisano u prethodnom delu (4.4). Kako su, ovom metodom, svi podskupovi *F* provereni, dobijeno rešenje je sigurno optimalno za MLUFLP instancu koja se razmatra. Takođe, na osnovu formula (4.1) – (4.5), koje predstavljaju matematičku formulaciju problema, implementiran je CPLEX program za pronalaženje optimalnog rešenja instanci malo većih dimenzija. Program je testiran na CPLEX 8.1.0 programskom paketu.

U ovom delu prikazani su dobijeni rezultati genetskog algoritma i njihovo poređenje sa poznatim optimalnim rešenjima. Sva testiranja vršena su na računaru sa AMD-ovim procesorom čiji je radni takt 1.8GHz i koji ima 512 MB radne memorije. Algoritmi su kodirani na programskom jeziku C.

Genetski algoritam je testiran na instancama iz ORLIB biblioteke (The Imperial College OR library) [Bea90, Bea96]. Ova biblioteka sadrži instance pogodne za testiranje mnogih problema operacionog istraživanja, među kojima je i lokacijski problem snabdevača neograničenog kapaciteta. Međutim, ove instance su dizajnirane samo za UFLP sa jednim nivoom lokacija i nemaju veliki broj potencijalnih lokacija. Zbog toga su, za testiranje genetskog algoritma na instancama većih dimenzija korišćene M* instance iz [Kra01a]. Instance na kojima su vršena testiranja prezentovana u [Edw01] nisu javno dostupne, ali su mahom bazirane na standardnim ORLIB instancama.

U ovom radu korišćene su instance generisane na osnovu ORLIB instanci, na sličan način kao i u [Edw01]. Kako ORLIB instance imaju samo jedan nivo hijerarhije, bilo je potrebno generisati rastojanja između lokacija na susednim nivoima, kao i rastojanja klijenata do poslednjeg nivoa lokacija, vodeći pri tome računa da se očuva nejednakost trougla. To podrazumeva da rastojanje između svake dve lokacije treba da bude manje ili jednako zbiru rastojanja (Su_{client}) tih lokacija do bilo kog klijenta. Takođe, svako rastojanje između dve lokacije treba da bude veće ili jednako od apsolutne vrednosti razlike (Di_{client}) rastojanja lokacija do bilo kog izabranog klijenta. Pri generisanju instanci autor u [Edw01] je vodio računa samo o prvom od dva navedena uslova, odnosno rastojanja dve lokacije su bila jednaka vrednosti najmanje sume rastojanja tih lokacija do nekog od klijenata $min_{client}(Su_{client})$. Pri generisanju instanci za testiranje genetskog algoritma korišćena je mnogo realističnija formula $\frac{min_{client}(Su_{client}) + max_{client}(Di_{client})}{2}$. Ista tehnika generisanja instanci u više nivoa primenjena je i na UFLP instance M* iz [Kra01a] koje su većih dimenzija u odnosu na ORLIB instance. Time je dobijen širok spektar instanci za testiranje implementiranog genetskog algoritma.

Kriterijumi zaustavljanja genetskog algoritma bili su maksimalan broj generacija $N_{gen} = 5000$ i ponavljanje najboljeg rešenja funkcije cilja u $N_{rep} = 2000$ uzastopnih generacija. Pošto su rezultati genetskog algoritma nedeterministički, algoritam je izvršavan po 20 puta na svakoj instanci. U Tabelama 4.5 i 4.6 su prikazani rezultati dobijeni samo na instancama iz ORLIB biblioteke i M* instancama.

Prva kolona u tabelama sadrži ime instance. Ime čuva informacije o: originalnoj instanci od koje je nastala, broju nivoa potencijalnih lokacija, broju potencijalnih lokacija na svakom nivou i broju klijenata. Na primer, instanca *capb_3l_12_25_63.1000* je nastala modifikovanjem ORLIB instance *capb*, ima 3 nivoa sa po 12, 25, 63 lokacija i 1000 klijenata.

Druga kolona sadrži vrednosti optimalnih rešenja odgovarajuće instance, odnosno znak – ukoliko optimalno rešenje nije poznato. U sledećoj koloni GA_{naj} su prikazana optimalna rešenja dobijena primenom genetskog algoritma na odgovarajućoj instanci. Prikazane su dobijene vrednosti, odnosno oznaka *optimalno* - ukoliko algoritam dostiže optimalno rešenje poznato u praksi (dobijeno primenom algoritma totalne enumeracije ili CPLEX programskog paketa), ili oznaka *nds* (najbolje do sada) – ukoliko genetski algoritam dostiže rešenje jednodimenzionog slučaja za koje se ne može dokazati da je optimalno.

Prosečno vreme potrebno da se dođe do optimalnog rešenja genetskog algoritma prikazano je u koloni *t,* dok kolona t_{tot} prikazuje ukupno (totalno) vreme potrebno da genetski algoritam završi sa radom. Genetski algoritam u proseku završava sa radom nakon *gen* generacija.

Kvalitet rešenja u svih 20 izvršavanja se meri relativnom greškom rešenja izraženom u procentima (*prg*), koja prikazuje odnos Opt_{re} i GA_{naj}. Procentualna greška rešenja se računa po formuli: $prg = \frac{1}{20}\sum_{i=1}^{20} rg_i$, pri čemu se sa rg_i označava relativna greška u *i*-tom izvršavanju genetskog algoritma. U situacijama kada je vrednost Opt_{re} poznata, rg_i se izračunava na osnovu formule: $rg_i = 100 * \frac{GA_i - Opt_{re}}{Opt_{re}}$,dok se u suprotnom, kada Opt_{re} nije poznato, dobija na osnovu formule: $rg_i = 100 * \frac{GA_i - NDS}{NDS}$. GA_i označava rešenje dobijeno u *i*-tom izvršavanju. Kvalitet rešenja ocenjuje se i sa σ, odnosno standardnom devijacijom relativne greške za rg_i , (*i = 1, 2, ..., 20)* koja se dobija iz formule: $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20} rg_i - prg^{\,2}}$.

Poslednje dve kolone odnose se na keširanje pri čemu *eval* – prikazuje prosečan broj izračunavanja (evaluacija), dok kolona ***keširanje*** prikazuje procentualnu uštedu korišćenjem tehnike keširanja.

Rezultati prikazani u Tabelama 4.5 i 4.6 pokazuju da genetski algoritam dostiže sva poznata optimalna rešenja dobijena na ORLIB i M* instancama za *k = 1*. Kada su u pitanju dve instance veće dimenzije: *ms1_1l_1000.1000* i *mt1_1l_2000.2000,* do danas nisu utvrđena optimalna rešenja, ali je genetski algoritam dostigao sva najbolja poznata rešenja prezentovana u [Kra01a]. Za instance sa više nivoa, a manjih dimenzija, do optimalnih rešenja se došlo primenom algoritma totalne enumeracije ili CPLEX programskog paketa. Genetski algoritam je pokazao odlične rezultate i na ovakvim instancama, dostigavši optimalna rešenja za vrlo kratko vreme izvršavanja. Predloženi algoritam dostigao je rešenja i pri testiranju sa M* i modifikovanim ORLIB instancama u razumnom vremenu izvršavanja.

U želji da se potvrdi potreba za izbacivanjem dupliranih jedinki pri rešavanju problema genetskim algoritmom urađeni su eksperimenti (GA_{dup}) bez njihovog uklanjanja. Za instancu malih dimenzija cap71_2_6_10.50, dostignuto je optimalno rešenje GA_{naj} i primenom GA_{dup} algoritma. Međutim, kada je u pitanju instanca capa_3l_15_30_55.1000, GA_{dup} je u 20 izvršavanja dostigao optimalno rešenje 43400224.60963, čija procentualna relativna greška u odnosu na GA_{naj} ima vrednost 6.16 %. Ovo odstupanje je bilo veoma veliko pa se odustalo od daljeg testiranja algoritma GA_{dup}.

Pomoću genetskih algoritama ne može da se dokaže optimalnost rešenja i ne postoji zadovoljavajući efektivni kriterijum zaustavljanja. Kao što kolona t_{tot} prikazuje algoritam se izvršava i za dodatno vreme $t_{tot} - t$ (sve dok kriterijum zaustavljanja ne bude zadovoljen) i ako je optimalno rešenje već dostignuto.

Tabela 4.5 Rezultati

Ime instance	Opt_{re}	GA_{naj}	t sec	t_{tot} sec	gen	N_{naj}	prg %	σ %	eval	keširanje %
cap71_1l_16.50	932615.750000	optimalno	0.012	0.606	2010.1	20	0.000	0.000	3507.7	96.5
cap101_1l_25.50	796648.437500	optimalno	0.030	0.781	2026.3	20	0.000	0.000	9614.3	90.5
cap131_1l_50.50	793439.562500	optimalno	0.199	2.037	2139.8	20	0.000	0.000	30453.8	71.6
mq1_1l_300.300	3591.273000	optimalno	14.288	63.289	2306.7	20	0.000	0.000	68927.1	40.3
mr1_1l_500.500	2349.856000	optimalno	74.852	220.295	2595.3	20	0.000	0.000	82950.8	36.1
ms1_1l_1000.1000	4378.632000	nds	534.888	1097.619	2979.7	20	0.000	0.000	100084.3	32.9
mt1_1l_2000.2000	9176.509000	nds	4134.325	5580.506	3871.2	20	0.000	0.000	136795.1	29.4
capa_1l_100.1000	17156454.478300	optimalno	13.409	77.864	2319.3	20	0.000	0.000	52072.8	55.1
capb_1l_100.1000	12979071.581430	optimalno	43.642	120.241	3047.2	18	0.060	0.186	76178.9	50.1
capc_1l_100.1000	11505594.328780	optimalno	34.542	102.535	2907.3	12	0.073	0.135	70952.4	51.2
cap71_2l_6_10.50	1813375.512500	optimalno	0.006	0.610	2009.5	20	0.000	0.000	3917.9	96.1
cap71_3l_2_5_9.50	4703216.306250	optimalno	0.005	0.557	2010.4	20	0.000	0.000	4300.5	95.7
cap101_2l_8_17.50	1581551.393750	optimalno	0.018	0.646	2017.4	20	0.000	0.000	7198.9	92.9
cap101_3l_3_7_15.50	3227179.812500	optimalno	0.019	0.612	2018.3	20	0.000	0.000	6797.8	93.3
cap131_2l_13_37.50	1592548.450000	optimalno	0.118	1.319	2078.0	20	0.000	0.000	16751.4	83.9
cap131_3l_6_14_30.50	3201970.462500	optimalno	0.105	1.276	2108.2	6	0.125	0.084	19877.3	81.3
cap131_4l_3_7_15_25.50	3630297.668750	optimalno	0.071	1.183	2048.7	20	0.000	0.000	18447.0	82.0

Tabela 4.6 Rezultati

Ime instance	Opt_{re}	GA_{naj}	t sec	t_{tot} sec	gen	N_{naj}	prg %	σ %	eval	keširanje %
capa_2l_30_70.1000	-	31524957.410085	8.380	49.235	2308.2	5	0.106	0.063	50829.1	56.0
capa_3l_15_30_55.1000	-	40725103.253535	3.076	27.518	2120.1	20	0.000	0.000	37316.4	64.9
capa_4l_6_12_24_58.1000	-	54643362.801190	4.688	36.480	2194.9	10	0.881	0.904	46738.3	57.5
capb_2l_35_65.1000	-	25224163.282875	18.414	58.981	2780.6	14	0.860	1.406	59650.7	57.3
capb_3l_12_25_63.1000	-	34978486.506140	3.137	23.788	2089.6	20	0.000	0.000	25763.8	75.4
capb_4l_6_13_31_50.1000	-	53034149.833035	4.652	29.840	2222.3	4	0.110	0.057	41192.1	63.0
capc_2l_32_68.1000	-	22762468.837500	22.625	63.857	3020.8	8	0.770	0.717	63538.1	58.1
capc_3l_13_27_60.1000	-	35540649.433070	8.539	38.997	2406.2	12	0.675	1.176	43836.3	63.5
capc_4l_4_9_27_60.1000	-	57017358.038275	4.601	39.784	2166.0	13	0.150	0.210	44774.9	58.7
mq1_2l_100_200.300	-	8341.287000	19.313	60.511	2670.4	20	0.000	0.000	77167.7	42.2
mq1_3l_30_80_190.300	-	12994.871000	16.980	55.666	2616.2	3	2.273	1.369	75305.6	42.4
mq1_4l_18_39_81_162.300	-	18048.030500	14.876	49.008	2619.9	8	0.736	0.876	75445.1	42.4
mq1_4l_20_40_80_160.300	-	17648.009500	17.423	51.654	2698.7	11	1.764	2.124	77449.5	42.6
mr1_2l_160_340.500	-	6707.505000	83.116	204.099	2918.3	14	0.611	0.988	91545.4	37.3
mr1_3l_55_120_325.500	-	10911.319000	76.009	187.238	2858.1	2	1.341	0.814	89511.4	37.4
mr1_4l_30_65_140_265.500	-	15311.469000	61.234	159.532	2773.3	2	1.544	0.879	86230.1	37.8
ms1_2l_320_680.1000	-	13416.805000	540.534	1032.551	3322.9	11	0.510	0.485	110437.6	33.5
ms1_3n_120_250_630.1000	-	21881.384000	501.034	998.641	3227.8	2	2.260	1.312	107150.1	33.6
ms1_4l_64_128_256_552.1000	-	30936.585000	418.521	833.667	3224.6	7	2.258	1.019	106793.9	33.8
ms1_5l_25_55_120_250_550.1000	-	40191.230500	396.686	814.261	3104.9	12	1.738	1.118	103072.1	33.7
mt1_2l_650_1350.2000	-	27733.057000	3949.573	5494.346	4309.1	16	0.331	0.704	150321.4	30.2
mt1_3l_256_600_1144.2000	-	46095.090000	3247.064	4874.861	4169.9	11	2.021	1.105	145827.9	30.0
mt1_4l_120_250_520_1110.2000	-	65044.002500	3084.885	4644.064	4153.6	7	1.126	0.649	145332.5	30.0
mt1_5l_60_120_250_500_1070.2000	-	83523.753000	2944.08	4444.225	4143.9	11	1.678	0.830	144979.3	30.0

U koloni *keširanje* se može videti veliki procenat uštede vremena izvršavanja korišćenjem tehnike keširanja. Prosečno, genetski algoritam koristi između 71.6% i 96.5% vrednosti iz heš-red strukture kada je u pitanju rešavanje instanci manjih dimenzija, odnosno između 29.4% i 75.4% vrednosti pri rešavanju problema velikih dimenzija.

Poređenje postojećih metoda sa genetskim algoritmom na instancama većih dimenzija M* i instancama iz ORLIB biblioteke je praktično nemoguće. Algoritmi MLRR, PR-RR i PR-LI su zasnovani na linearnoj relaksaciji modela koja ima enormno veliki broj uslova i promenljivih. Na primer, za instancu koja ima *n = 2000* klijenata i *m = 2000* lokacija, ove metode bi za rešavanje morale da uvedu $2 * 10^{15}$ promenljivih i 2000 * 2000 = 4 000 000 uslova. Kako je celobrojni linearni problem NP – težak, sve metode imaju eksponencijalnu složenost, odnosno vreme izvršavanja je $O(c^{2*10^{15}})$. Očigledno je da, osim kada je u pitanju CPLEX programski paket i dovoljno malo *c*, ove metode nisu u mogućnosti da reše problem u razumnom vremenu izvršavanja. Metod SP je u stanju da rešava probleme i većih dimenzija, ali ima neprihvatljivo veliko odstupanje koje uzima vrednosti i do 732%. Prostor pretrage genetskog algoritma je očigledno 2^m, jer je korišćeno binarno kodiranje sa *m* gena. Za pomenute instance prostor pretrage je veliki ($2^{2000} \approx 10^{600}$), ali u svakom slučaju mnogo manji nego kada je u pitanju celobrojni linearni model zasnovan na uslovima (4.1) – (4.5). Ovo je direktna posledica korišćenja dinamičkog programiranja pri računanju funkcije cilja.

Kao što rezultati iz tabela pokazuju, genetski algoritam veoma lako dolazi do rešenja kada su u pitanju instance normalne veličine. Kako je genetski algoritam u razumnom vremenu izvršavanja dostigao sva optimalna/najbolja poznata rešenja kada su u pitanju instance sa jednim nivoom i instance manje dimenzije u više nivoa, logično je pretpostaviti da ovaj algoritam dostiže optimalna rešenja i u ostalim slučajevima na kojima je testiran.

Genetski algoritam veoma brzo dostiže optimalno rešenje pri radu sa instancama male dimenzije (do 50 lokacija i do 50 klijenata), što pokazuju rezultati u koloni t_{tot}. Maksimalno vreme t_{tot} nije veće od 2.04 sekunde na ovim instancama. Instance približnih dimenzija su korišćene i za testiranja u [Edw01], ali je SP metod dostizao rešenja sa velikim procentualnim odstupanjima. Prezentovani algoritam optimalno rešenje dostiže u razumnom vremenu izvršavanja (do 5580 sekundi), što se može videti iz predstavljenih tabela.

5 ZAKLJUČAK

U ovom radu opisani su genetski algoritmi koji su posebno dizajnirani za rešavanje hijerarhijsko-lokacijskih problema. Implementirani algoritmi su primenjeni na tri *NP*-teška problema ovakve vrste: HWSP, HCLP, MLUFLP. Rešavani problemi imaju velike primene u praksi. Za svaki od rešavanih problema detaljno su opisani teoretski i praktični aspekti predložene implementacije. Izvršeno je i poređenje rezultata dobijenih pomoću genetskih algoritama sa rezultatima iz literature.

Hijerarhijsko-lokacijski problemi, koji su razmatrani u radu, su teški za rešavanje jer su čak i njihove varijante sa jednim nivoom, takođe NP-teške. Između ostalog, to je jedan od razloga što postojeće egzaktne metode mogu da se uspešno primene samo na konkretne probleme relativno malih dimenzija. Predloženi evolutivni pristup je vrlo značajan, jer robustnost i adaptivnost predloženih genetskih operatora omogućuju vrlo uspešno rešavanje datih problema, čak i na instancama vrlo velikih dimenzija.

Korišćene su dve vrste kodiranja - binarno i realno. Za rešavanje HWSP korišćeno je realno kodiranje, a potencijalna rešenja su se iz genetskog koda dobijala primenom D'ont-ovog sistema. Ovaj sistem se koristi za određivanje broja mandata stranaka, na osnovu rezultata izbora u nekim državama. Pri rešavanju ostala dva problema korišćeno je binarno kodiranje prilagođeno prirodi problema koji se rešava.

Dizajnirani su i implementirani modifikovani genetski operatori, koji vode računa o načinu kodiranja i prirodi problema. Ovi operatrori, između ostalih funkcija, čuvaju i korektnost jedinki, čime se isključuje pojava nekorektnih jedinki u svakoj generaciji. Genetski operatori koji su pokazali najbolje rezultate pri rešavanju razmatranih problema su: Fino gradirana turnirska selekcija, modifikovani opreratori ukrštanja i operatori mutacije sa zaleđenim bitovima.

Kod implementacije genetskih algoritama keširanje u velikoj meri poboljšava performanse, ali ne utiče na ostale aspekte algoritama. Takođe, primenjene su i razne metode za sprečavanje preuranjene konvergencije, gubljenja raznovrsnosti genetskog materijala i spore konvergencije genetskog algoritma. Prikazane implementacije koriste elitističku strategiju koja omogućava direktno prenošenje vrednosti elitnih jedinki u narednu generaciju, bez ponovnog računanja njihovih funkcija cilja.

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

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

5.1 Naučni doprinos rada

Najvažniji novi rezultati dobijeni istraživanjem prikazanim u ovom radu su:

- Korišćenje koncepta dinamičkog programiranja za drastično smanjivanje prostora pretraživanja mogućih putanja pri rešavanju lokacijskog problema snabdevača neograničenog kapaciteta u više nivoa.
- Definisanje novih načina kodiranja i odgovarajućih funkcija cilja takvih da se čuva korektnost jedinki i omogućava efikasna implementacija.
- Osmišljavanje novih genetskih operatora koji odgovaraju prirodi problema, primenjenim načinima kodiranja i čuvaju dopustivost rešenja.
- Dobijanje rešenja na instancama problema većih dimenzija koje do sada nisu rešavane.
- Analiza ponašanja rešenja u graničnim situacijama.
- Procena složenosti funkcija cilja.
- Ocena uticaja podešavanja pojedinačnih parametara na rad genetskog algoritma.

Kao što se može videti iz eksperimentalnih rezultata, predloženi genetski algoritmi su veoma uspešni pri rešavanju hijerarhijskih diskretnih lokacijskih problema. Zbog svega gore navedenog, naučno istraživanje opisano u ovom radu daje doprinos oblastima kombinatorne optimizacije, lokacijskih problema i genetskih algoritama. Dobijeni rezultati su već objavljeni u međunarodnim časopisima i publikovani, odnosno prezentovani na međunarodnim i domaćim naučnim i stručnim konferencijama.

LITERATURA

[Aar99] **Aardal K., Chudak F. Shmoys D.B**., "A 3-approximation algorithm for the k-level uncapacitated facility location problem", Information Processing Letters, Vol. 72, pp. 161-167, (1999).

[Abr95] **Abramson D., De Silva A.**, "A Parallel Interior Point Algorithm and its Application to Facility Location Problems", Research Report CIT:95-12,School of Computing and Information Technology, Griffith University, Nathan (1995).

[Age02] **Ageev A.**, "Improved approximation algorithms for multilevel facility location problems", Operations Research Letters, Vol. 30, pp. 327-332, (2002).

[Age05] **Ageev A. Ye Y. Zhang J.**, "Improved combinatorial approximation algorithms for the k-level facility location problem", SIAM Journal on Discrete Mathematics, Vol. 18, pp. 207-217, (2005).

[Ala08] **Alander, J.T.**,"An indexed bibliography of genetic algorithms", Technical Report DRAFT 2008/05/21, Department of Electrical Engineering and Automation, University of Vaasa, Finland (2008), http://garbo.uwasa.fi/pub/cs/report94-1/gaALANDERbib.pdf

[Aik85] **Aikens C.H.**, "Facility location models for distribution planning", European Journal of Operational Research 22, pp.263-279 (1985).

[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).

[Bäc00a] **Bäck T., Fogel D. B., Michalewicz Z.**: "Basic Algorithms and Operators", in: Evolutionary Computation 1, Institute of Physics Publishing, Bristol-Philadelphia, (2000).

[Bäc00b] **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

[Bea88] **Beasley J.E.**, "An algorithm for solving large capacitated warehouse location problems", European Journal of Operational Research 33, pp. 314-325 (1988).

[Bea90] **Beasley, J. E.**: OR Library: distributing test problems by electronic mail. Journal of the Operational Research Society, Vol. 41, pp. 1069-1072, (1990).

[Bea93a] **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

[Bea93b] **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).

[Bea95] **Beasley J.E.**, "Lagrangean Relaxation", In: Modern Heuristic Techniques for Combinatorial Problems, Reeves C.R. (ed), McGraw-Hill, pp. 243-303 (1995).

[Bea96] **Beasley J.E.**, "Obtaining Test Problems via Internet", Journal of Global Optimization, Vol. 8, pp. 429-433 (1996). ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview2.ps

[Bëc92] **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).

[BeD93a] **Beasley D.**, **Bull D.R.**, **Martin R.R.**, "An Overview of Genetic Algorithms, Part 1, Fundamentals", University Computing 15, 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 15, pp. 170-181 (1993).

[Ber03] **Berman, O., Drezner, Z., Wesolowsky. G.O**., "The expropriation location problem", Journal of Operational Research Society, Vol. 54, pp. 769–776, (2003).

[Bil99] **Billionnet A.**, "Integer programming to schedule a hierarchical workforce with variable demands", European Journal of Operational Research 114, pp. 105–114, (1999).

[Bof97] **Boffey T.B., Narula S.C.,** "Multiobjective covering and routing problems", In: Karwan MH, Spronk J, Wallenius J, editors. Essays in decision making: a volume in honor of Stanley Zionts. Berlin: Springer (1997).

[Bow89] **Bowler P.**, "The Mendelian Revolution: The Emergence of Hereditarian Concepts in Modern Science and Society", BaltimoreGenetics: The life of DNA Johns Hopkins University Press (1989).

[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).

[Brm91] **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).

[Bum01] **Bumb, A. F. Kern, W.,** "A Simple Dual Ascent Algorithm for the Multilevel Facility Location Problem", Proceedings of the 4th International Workshop on Approximation Algorithms for CombinatorialOptimization Problems and 5th International Workshop on Randomization and Approximation Techniques in Computer Science: Approximation, Randomization and Combinatorial Optimization, pp. 55-62, (2001).

[Cha99] **Charikar M., Guha S.**, "Improved combinatorial algorithms for the facility location and k-median problems", In FOCS: IEEE Symposium on Foundations of Computer Science (FOCS) (1999).

[Chr74] **Church R.L., ReVelle C.S.,** "The maximal covering location problem", Papers of the Regional Science Association 32, pp. 101–18 (1974).

[Chu99] **Chudak F., Shmoys D.**,"Improved approximation algorithms for capacitated facility locationproblem", In Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 875-876, (1999).

[Coo71] **Cook S. A.**, "The complexity of theorem-proving procedures", Proc. 3rd Ann. ACM Symp on Theory of Computing, ACM, New York, pp. 151-158 (1971).

[Crr01] **Correa E. S., Steiner M.T., Freitas A., Carnieri C.,** "A Genetic Algorithm for the p-median Problem", GECCO, (2001).

[Cor06] **Correa F. A., Lorena L. A., N. Senne E. L. F.**, "Lagrangean Relaxation with Clusters for the Uncapacitated Facility Location Problem", Proceedings of the XIII Congreso Latino-Iberoamericano de Investigacion Operativa - CLAIO, Montevideo, Urugvay, (2006).

[Cpl08] http://www.ilog.com/products/cplex/

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

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

[Dar59] **Darwin C.**, "The origin of species", London (1859).

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

[Dia05a] **Dias J. M., Caprtivo M. E., Clímaco, J.**, "A Memetic Algorithm for Dynamic Location Problems", MIC2005: Proceedings of the Sixth Metaheuristics International Conference, pp. 284 – 289 (2005).

[Dia05b] **Dias J. M., Caprtivo M. E., Clímaco J.,** "A Hybrid Algorithm for Dynamic Location Problems", Instituto de Engenharia de Sistemas e Computadores de Coimbra, working paper No. 3 (2005).

[Don08] http://sr.wikipedia.org/sr-Д'Онтов_систем

[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).

[Dor04] **M. Dorigo and T. Stutzle**, "Ant colony optimization", MIT Press, Cambridge, (2004).

[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).

[Dvo99] **Dvorett J.**, "Compatibility-Based Genetic Algorithm: A New Approach to thr p-Median Problem" (1999).

[Đur08] **Đurić B., Kratica J., Tošić D., Filipović V.,** "Solving the maximally balanced connected partition problem in graphs by using genetic algorithm", Computing and Informatics, Vol. 27, pp. 341-354 (2008).

[Edw01] Edwards, N. J. "Approximation algorithms for the multi-level facility location problem", Ph.D. Thesis, Cornell University, (2001).

[Emm85] **Emmons H.**, "Workforce scheduling with cyclic requirements and constraints on days off, and workstretch", IIE Transactions 17 pp. 8–16, (1985).

[Emm91] **Emmons H., Burns**, "Off-day scheduling with hierarchical worker categories", Operations Research 39, pp. 494–497, (1991).

[Ern04] **Ernst A.T., Jiang H., Krishnamoorthy M., Sier D.,** "Staff scheduling and rostering: A review of applications, methods and models", European Journal of Operational Research 153, pp. 3–27, (2004).

[Esp03] **Espejo, L.G.A., Galvao, R.D., Boffey, B**., "Dual-based heuristics for a hierarchical covering location problem", Computers & Operations Research, Vol. 30, pp. 165-180 (2003).

[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).

[Fis04] **Fisher M.**, "The Lagrangian Relaxation Method for Solving Integer Programming Problems", Management Science 50, pp. 1872-1874 (2004).

[Fra83] **Francis R.L., McGinnis L.F., White J.A.**, "Locational analysis", EuropeanJournal of Operational Research 12, pp. 220-252 (1983).

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

[Gal93] **Galvao R.D.**, "The use of Lagrangean Relaxation in the solution of uncapacitated facility location problems", Location Science 1, pp. 57-79 (1993).

[Gal96] **Galvao, R.D., ReVelle, C.S.**, "A Lagrangean heuristic for the maximal covering location problem", European Journal of Operational Research, Vol. 88, pp. 114–123 (1996).

[Gal00] **Galvao R.D., Espejo L.G.A., Boffey B.**, "A comparison of Lagrangean and surrogate relaxations for the maximal covering location problem", European Journal of Operational Research 124, pp. 377–389 (2000).

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

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

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

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

[Glo97] **F. Glover, F. Laguna**, "Tabu search", Kluwer Academic Publishers, Norwell, MA, USA (1997).

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

[God05] Godor I., Magyar G., "Cost-optimal topology planning of hierarchical access networks", Computers and Operations Research 32, pp. 59–86 (2005).

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

[Gre03] **Greistorfer P., Regoy C., Alidaee B.**, "A Filter-and-Fan Approach for the Facility Location Problem", MIC2003: Proceedings of the the Fifth Metaheuristics International Conference, pp. 271-276, (2003).

[Gre06] **Greistorfer P., Rego C.**, "A simple filter-and-fan approach to the facility location problem", Computers and Operations Research, Vol. 33, No. 9, pp. 2590-2601, (2006).

[Gui88] **Guignard M.**, "A Lagrangean dual ascent algorithm for simple plant location problems", European Journal of Operational Research 5, pp. 193-200 (1988).

[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).

[Han01] **P. Hansen and N. Mladenović**, "Variable neighborhood search: Principles and applications," European Journal of Operational Research 130, pp. 449-467, 2001.

[Han07] Hansen P., Brimberg J., Urošević D., Mladenović N., "Primal-Dual Variable Neighborhood Search for the Simple Plant-Location Problem", INFORMS Journal on Computing 19, pp. 552-564 (2007).

[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).

[Hun93] **Hung R., Emmons H.**, "Multiple-shift workforce scheduling under the 3–4 compressed workweek with a hierarchical workforce", IIE Transactions 25 (5), pp. 82–89 (1993).

[Hun94] **Hung R.**, "Single-shift off-day scheduling of a hierarchical workforce with variable demands", European Journal of Operational Research 78 (1), pp. 49–57 (1994).

[Hun96] **Hung R.**, "Using compressed workweeks to reduce work commuting", Transportation Research 30, pp. 1–19 (1996).

[Hun04] **Hung R.**, "Using compressed workweeks to save labour cost", European Journal of Operational Reseach, http://www.sciencedirect.com, 2004.

[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).

[Jai02] **Jain K., Mahdian M., Saberi A.**, "A New Greedy Approach for Facility Location Problem", *Proc. Ann. ACM Symp. Theory of Computing* (2002).

[Juh06] **Juhos, I., Van Hemert, J.I**., "Improving graph colouring algorithms and heuristics using a novel representation", Lecture Notes in Computer Science, Vol. 3906, pp. 123-134 (2006).

[Ken01] **J. Kennedy, R.C. Eberhart ,Y. Shi**, "Swarm intelligence",Morgan Kaufmann Publishers, San Francisco, USA (2001).

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

[Klo04] **Klose A., Drexl A.,** "Facility location models for distribution system design", European Journal of Operational Research, Vol. 162, pp. 4-29, (2005).

[Kov08] **Kovačević, J.**, "Hybrid Genetic Algorithm For Solving The Low-Autocorrelation Binary Sequence Problem", Yugoslav Journal of Operations Research (2008).

[Kra96] **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).

[Kra99a] **Kratica J.**, "Improving Performances of the Genetic Algorithm by Caching", Computers and Artificial Inteligence, Vol. 18, No. 3, pp. 271-283 (1999).

[Kra99b] Kratica J., "Improvement of Simple Genetic Algorithm for Solving the Uncapacitated Warehouse Location Problem", Advances in Soft Computing - Engineering Design and Manufacturing, R.Roy, T. Furuhashi and P.K. Chawdhry (Eds), Springer-Verlag London Limited, 1999, pp. 390-402.

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

[Kra01a] **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).

[Kra01b] **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).

[Kra06] **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).

[Kra07] **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, No. 1, pp 15-28 (2007).

[Kra08a] **Kratica J., Kovačević-Vujčić V., Čangalović M.,** "Computing the Metric Dimension of Graphs by Genetic Algorithms", submitted to Computational Optimization and Applications, DOI 10.1007/ s10589-007-9154-5 (2008).

[Kra08b] **Kratica J., Kovačević-Vujčić V., Čangalović M.**, "Computing strong metric dimension of some special classes of graphs by genetic algorithms", Yugoslav Journal of Operations Research 18, pp. 143-151 (2008).

[Kra08c] **Kratica J., Kovačević-Vujčić V., Čangalović M.**, "Computing minimal doubly resolving set of graphs", Computers & Operations Research doi:10.1016/j.cor.2008.08.002 (2008)

[Kra08d] **Kratica J., Tošić D., Filipović V., Dugošija Đ.,** "Two hybrid genetic algorithms for solving the super-peer selection problem", WSC 2008.

[Kra08e] **Kratica J., Kostić T., Tošić D., Dugošija Đ., Filipović V.,** "A genetic algorithm for the routing and carrier selection problem", Asia-Pacific Journal of Operational Research (2008).

[Krp83] **Krarup J., Pruzan P. M.**, "The simple plant location problem: Survey and synthesis", European Journal of Operational Research, Vol. 12, pp. 36-81 (1983).

[Leg04] **Legg S, Hutter M, Kumar A.**: "Tournament versus Fitness Uniform Selection", Technical Report, IDSIA-04-04 (2004).

[Lov88] **Love R.F., Moris J.G., Wesolowsky G.**, "Facility location: Models and methods", Publication in Operations Research 7, North-Holland, New York (1988).

[Lju00a] **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).

[Lju00b] **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).

[Lju03] Ljubić, I., Raidl, G.R., A memetic algorithm for minimum-cost vertex-biconnectivity augmentation of graphs, Journal of Heuristics, Vol. 9, pp. 401-427 (2003).

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

[Mar08a] **Marić M**., "An Efficient Genetic Algorithm For Solving The Multi-Level Uncapacitated Facility Location Problem", accepted for publication in Computing and Informatics, (Ref N° 1829).

[Mar08b] **Marić M., Kratica J., Tuba M.**, "Parameter Adjustment for Genetic Algorithm for Two-Level Hierarchical Covering Location Problem" WSEAS Transactions, Volume 7, pp. 746 – 755 (2008).

[Mar08c] **Marić M., Kratica J., Tuba M.**, "One Genetic Algorithm for Hierarchical Covering Location Problem", Proceedings of the 9th WSEAS International Conference on Advanced Topics On Evolutionary Computing (EC'08)

[Med05] **Medaglia A. L., Gutiérrez E., Villegas J. G.**, "A Tool for Rapid Development of Multi-Objective Evolutionary Algorithms (MOEAs) with Application to Facility Location Problems", MO-JGA:Implementing MOEAs in Java (2005).

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

[Mil08] **Milanović M.**, "Solving the generalized vertex cover problem by genetic algorithm", Computing and Informatics (2008).

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

[Mis05] **A. Misevicius**, "A tabu search algorithm for the quadratic assignment problem," Computational Optimization and Applications 30, pp. 95-111, 2005.

[Mit99] **Mitchell M.,** "An Introduction to Genetic Algorithms", MIT Press, Cambridge, Massachusetts, (1999).

[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).

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

[Nar84] Narula S.C., "Hierarchical location-allocation problems: a classification scheme", European Journal of Operational Research 15, pp 93–99, (1984).

[Nar97] **Narasinhan R.**, "Algorithm for single shift scheduling of hierarchical workforce", European Journal of Operational Research 96 (1), pp. 113–121 (1997).

[Nar00] **Narasinhan R**., "Algorithm for multiple shift scheduling of hierarchical workforce on four-day or three-day workweeks", INFOR Journal 38 (1), pp. 14–32 (2000).

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

[Ognj01] **Ognjanović Z., Kratica J., Milovanović M.**, "A Genetic Algorithm for Satisfiability Problem in a Probabilistic Logic: A First Report", Springer Lecture Notes in Artificial Intelligence - LNAI, Vol. 2143, pp. 805-816 (2001).

[Ognj04a] **Ognjanović Z., Krdžavac N.,** "Uvod u teorijsko računarstvo", Fakultet organizacionih nauka, Beograd (2004).

[Ognj04b] **Ognjanović Z., Midić U., Kratica J.**, "A Genetic Algorithm for Probabilistic SAT Problem", Lecture Notes in Artificial Intelligence - LNAI, Vol. 3070, pp. 462–467, 2004.

[Ong04] **Ongsakul W., Petcharaks N.**, "Unit commitment by enhanced adaptive Lagrangian relaxation", IEEE Transactions Power Systems 19, pp. 620-628 (2004).

[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).

[Pap82] **Papadimitrou C. H., Steiglitz K.,** "Combinatorial Optimization: Algorithms and Complexity", Prentice-Hall, Englewood Cliffs (1982).

[Pla06] **Plaxton G**., "Approximation algorithms for hierarchical location problems"**,** Journal of Computer and System Sciences 72, pp. 425 - 443 (2006).

[Puc06] **Puchinger, J., Raidl, G.R., Pferschy, U.**, "The core concept for the multidimensional knapsack problem", Lecture Notes in Computer Science, Vol. 3906, pp. 195-208 (2006).

[Rai05] **Raidl, G.R., Gottlieb, J.**, "Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem", Evolutionary Computation, Vol. 13, No. 4, pp. 441-475 , (2005).

[Res06] **Resende M. G. C., Werneck R. F.**, "A hybrid multistart heuristic for the uncapacitated facility location problem", European Journal of Operational Research, Vol. 174, No. 1, pp. 54-68 (2006).

[ReV05] **ReVelle C. S., Eiselt H. A.**, "Location analysis: A synthesis and survey", European Journal of Operational Research, Vol. 165, 2005, pp. 1–19 (2005).

[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).

[Sav08a] **Savić A., Tošić D., Marić M, Kratica J.,** "An genetic algorithm approach for solving the task assignment problem", Serdica Journal of Computing (2008).

[Sav08b] **Savić A.,** "An genetic algorithm approach for solving the machine-job assignment with controllable processing times", Computing and Informatics (2008).

[Sah07] **Şahin G., Süral H.**, "A review of hierarchical facility location models", Computers and Operations Research 34, pp. 2310-2331 (2007).

[Sch93] **Schilling D.A., Vaidyanathan J., Barkhi R.**, "A review of covering problems in facility location", Location Science pp. 25–55 (1993).

[Sec07] Seckiner S. U., Gokcen H. , Kurt M., "An integer programming model for hierarchical workforce scheduling problem", European Journal of Operational Research, Vol 183, pp. 694 – 699 (2007).

[Shm97] **Shmoys D.B., Tardos E., Aardal K.**, "Approximation algorithms for facility location problems", Techical Report UU-CS-1997-39, Department of Computer Science, Utrecht University, 21 p. (1997).

[Sny06] **Snyder L.**, "Facility location under uncertainty: a review", IIE Transactions, Volume 38, pp. 547-564 (2006).

[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).

[Sta07a] **Stanimirović Z. Kratica J. Dugošija Đ.**, "Genetic algorithms for solving the discrete ordered median problem", European Journal of Operational Research, Vol. 182, No. 3, pp. 983-1001 (2007).

[Sta07b] **Stanimirović Z.**, "Genetic algorithms for solving some NP-hard hub location problems", Ph.D. thesis, University of Belgrade, Faculty of Mathematics (2007).

[Sta08] **Stanimirović, Z.**, "A genetic algorithm approach for the capacitated single allocation p-hub median problem", Computing and Informatics 27, (in press).

[Sum02] **Suman B.**, "Multiobjective simulated annealing—a metaheuristic technique for multiobjective optimization of a constrained problem", Foundations of Comput Decision Sci 27, pp. 171–191 (2002).

[Sum06] **Suman B., Kumar P.**, "A survey of simulated annealing as a tool for single and multiojbective optimization", Journal of the operational research society 57 pp. 1143–1160 (2006).

[Sun06] **Sun M.,** "Solving the uncapacitated facility location problem using tabu search", Computers and Operations Research, Vol. 33, No. 9, pp. 2563-2589 (2006).

[Tan06] **Tang L., Xuan H., Liu, J.**, "A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time", Computers and Operations Research 33 , pp. 3344 – 3359 (2006).

[Top02] **Topaloglu S., Ozkarahan I.,** "Implicit optimal tour scheduling with flexible break assignments", Computers & Industrial Engineering 44, pp. 75–89 (2002).

[Tošˇ04] **Tošić D., Mladenović N., Kratica J., Filpović V.,** "Genetski algoritmi", Matematički institut SANU, Beograd (2004).

[Trm00] **Trmčić (Ljubić) I.**, "Primena genetskih algoritama na probleme povezanosti grafova", Magistarski rad, Univerzitet u Beogradu, Matematički fakultet (2000).

[Vil06] **Villegas J., G. Palacios F., Medaglia, A.L.**, "Solution methods for the bi-objective (cost-coverage) unconstrained facility location problem with an illustrative example", Annals of Operations Research, Vol. 147, No. 1, pp. 109-141 (2006).

[Vyg05] **Vygen J.,** "Approximation Algorithms for Facility Location Problems" (Lecture Notes), Research Institute for Discrete Mathematics, University of Bonn, Report No. 05950-OR (2005).

[Yig03] **Yigit V., Turkbey O.**, "An approach to the facility location problems with hill-climbing and simulated annealing", Journal of The Faculty of Engineering and Architecture of Gazi University, Vol. 18, No. 4,pp. 45-56 (2003).

[Zha06] **Zhang J.**, "Approximating the two-level facility location problem via a quasi-greedy approach", Mathematical Programming, Vol. 108, pp. 159-176 (2006).

[Živ00] **Živković M.**, Algoritmi, Matematički fakultet (2000).

SADRŽAJ