UNIVERZITET U BEOGRADU
Matematički fakultet

Biljana Borak

GENETSKI ALGORITAM ZA REŠAVANJE LOKACIJSKOG PROBLEMA SNABDEVAČA OGRANIČENOG KAPACITETA U VIŠE NIVOA

Diplomski - master rad

B e o g r a d

2009.

Mentor:	**Doc dr Miroslav Marić** Matematički fakultet u Beogradu
Komentor:	**dr Jozef Kratica** Viši naučni savetnik Matematički institut SANU
Članovi komisije:	**Doc dr Vladimir Filipović** Matematički fakultet u Beogradu

Datum odbrane: _____

Genetski algoritam za rešavanje lokacijskog problema snabdevača ograničenog kapaciteta u više nivoa

Rezime

U ovom radu opisan je genetski algoritam za rešavanje lokacijskog problema snabdevača ograničenog kapaciteta u više nivoa (Multi-Level Capacitated Facility Location Problem - MLCFLP). Osnovni problem, lokacijski problem snabdevača ograničenog kapaciteta (CFLP) u jednom nivou je takođe NP-težak problem i veoma je zastupljen u literaturi, a ima i značajnu primenu u praksi. MLCFLP se može koristiti dodatno i za određivanje lokacija snabdevača, lokacija banaka i pošti u više nivoa, lokacija zdravstvenih i obrazovnih ustanova.

Pored definisanja matematičkog modela ovog problema, po prvi put je data formulacija kao problema mešovitog celobrojnog linearnog programiranja. Problem je takođe rešavan pomoću genetskog algoritma. Zbog složenosti uslova primenjeno je binarno kodiranje, gde genetski algoritam određuje koje će lokacije snabdevača biti uspostavljene. Zatim, odgovarajući potproblem sa fiksiranim lokacijama snabdevača se rešava kao problem običnog linearnog programiranja pomoću CPLEX-a. Na taj način mogli su biti primenjeni standardni genetski operatori i sve jedinke su bile korektne. Genetski algoritam je dostigao sva poznata optimalna rešenja u dostižnom vremenu izvršavanja.

Ključne reči: Genetski algoritmi, Lokacijski problemi, Kombinatorna optimizacija

A genetic algorithm for the multi-level capacitated facility location problem

Abstract

In this paper new genetic algorithm (GA) for solving the multi-level capacitated facility location problem (MLCFLP) is presented. This NP-hard problem has been widely studied in the literature and it has very important practical implications - the design of a supply chain systems, the multi-level structure of bank and post-offices organizations, the layout of health service system, the layout of education system.

Alongside mathematical formulation of the given problem, for the first time it is being formulated as a problem of mixed integer linear programming and solved using a genetic algorithm. Due to the constrains complexity, the binary coding was deployed where genetic algorithm defines locations of the suppliers. Upon this, related issue of fixed location supplier is being addressed as a problem of regular linear programming using CPLEX. In this manner the standard genetic operators can be applied where all units were found to be correct. Genetic algorithm achieved all known optimal solutions in available execution time.

Keywords: Genetic algorithm, Location problems, Combinatorial optimization

1 UVOD

1.1 Lokacijski problemi

Lokacijski poblemi predstavljaju posebnu klasu problema optimizacije. Najčešće se zahteva minimizacija rastojanja, ukupnog vremena putovanja ili nekog drugog parametra. Ovi problemi se odnose na traženje lokacija objekata koji će biti uspostavljeni. Ovi objekti za čije se lociranje traži mesto obično su centri koji pružaju neke usluge, pa se nazivaju snabdevači, a korisnici tih usluga se nazivaju klijenti. Ovi problemi imaju veliku mogućnost primene u raznim oblastima, zbog čega su veoma popularni, a kao takvi privlače velika interesovanja i često su predmet istraživanja. Svaki lokacijski problem, kao i njegova forma – funkcija cilja, uslovi i promenljive su svaki za sebe specifični, pa ne postoji univerzalni algoritam koji bi se mogao primeniti na rešavanje svih diskretnih lokacijskih problema.

Lokacijski problemi se najčešće dele na osnovu načina predstavljanja promenljivih na diskretne i kontinualne. Diskretni lokacijski problemi su problemi kod kojih su mesta na koja se može uspostaviti objekat elementi nekog konačnog skupa. S druge strane, kod kontinualnih lokacijskih problema, objekti se mogu postaviti na bilo koju lokaciju kontinualnog prostora. Takođe, moguća je i kombinacija diskretnih i kontinualnih problema – mrežni problemi.

Lokacijski problemi se mogu podeliti i na osnovu broja objekata koje treba locirati na endogene, kod kojih je broj objekata unapred poznat, i egzogene, kod kojih je broj objekata nepoznata veličina koja se dobija kao rezultat optimizacije. Predstavnici egzogenih problema su prost lokacijski problem (Simple-Plant

Location ili Uncapacitated Facility Location) [Kra96, Kra98, Kra00, Kra01a, Han07] i problem prekrivanja skupova [Bof97, Chr74, Gal00, Sch93].

Lokacijski problemi se na osnovu ograničenosti kapaciteta mogu podeliti na one sa ograničenim i na one sa neograničenim kapacitetom.

O ovim problemima, njihovoj klasifikaciji i metodama njihovog rešavanja može se informisati u radovima [Aik85, Dea85, Dre02, Fra83, Fra92, Klo04, Lov88, Mir90, ReV05], a o nekim specifičnim metodama u [Abr95, Jai02, Shm97, Sny06].

1.2 Genetski algoritmi

1.2.1 Opšte karakteristike genetskih algoritama

Genetski algoritmi su heuristička metoda optimizacije koja rešava određene računarske probleme simulirajući mehanizam prirodne evolucije.

Pojam genetskog algoritma kao modela predložen je 70-ih godina prošlog veka od strane Džona Holanda (Johan H. Holland) sa ciljem proučavanja adaptivnog ponašanja, što se može i videti iz naslova knjige koju je napisao na početku svog istraživanja: *Adaptacija u prirodnim i veštačkim sistemima (Adaptation in natural and artificial systems)* [HII75]. On je u svom radu predložio genetski algoritam kao računarski proces koji imitira evolutivni proces u prirodi. Ovaj algoritam se često naziva prost genetski algoritam ili kanonski genetski algoritam. Više o konceptu genetskih algoritama se može videti u [Bäc00a, Bäc00b, Bea93a, Bea93b, Gol89, Mic96, Mit99, Mü**h**97], a zastupljeni su i u domaćoj literaturi [Čan96, Fil98, Kra00, Kra01b, Toš04].

Prost genetski algoritam koristi binarnu reprezentaciju, prostu selekciju, ukrštanje sa jednom tačkom prekida i prostu mutaciju. Ova varijanta genetskog algoritma može se naći u Holandovoj knjizi [HII75], a osobine genetskih operatora date su u jednom od narednih poglavlja. Prost genetski algoritam je često bio tema mnogih ranijih studija i jos uvek se koristi kao merilo za novije genetske algoritme. Na žalost, ovaj algoritam ima i puno mana: binarna reprezentacija je suviše restriktivna, mutacija i ukrštanje koje koristi mogu da se primenjuju samo za binarne i celobrojne reprezentacije, selekcija je osetljiva na konvergirajuće populacije sa bliskim vrednostima fitnesa i konvergencija ka rešenju je jako spora.

Kao sto je već spomenuto, genetski algoritmi simuliraju prirodni evolutivni proces. Za evolutivne procese kao i za genetske algoritme može se ustanoviti da:

- postoji populacija jedinki;
- neke jedinke su bolje prilagođene okolini;
- bolje jedinke imaju veću verovatnoću preživljavanja i reprodukcije;
- osobine jedinki zapisane su pomoću genetskog koda;
- deca nasleđuju osobine roditelja;
- jedinke mogu mutirati.

Kod genetskih algoritama jedinke predstavljaju trenutne aproksimacije rešenja problema koji se rešava. Svaka jedinka se kodira i svakoj jedinki se pridružuje određena mera kvaliteta – *fitness*, koja se određuje pomoću funkcije cilja. Prilikom inicijalizacije generiše se početna populacija. Ona se obično generiše slučajnim izborom rešenja iz domena. Dozvoljeno je da se početno rešenje dobijeno nekom drugom metodom optimizacije doda početnoj populaciji. Zatim, sledi proces koji se ponavlja dok god se ne zadovolji uslov zaustavljanja. Taj proces se sastoji od izvršavanja genetskih operatora selekcije, ukrštanja i

mutacije. Osim procene kvaliteta koja se mora obaviti nad jedinkama, sve operacije genetskog algoritma se sprovode nad kodiranim jedinkama. Višestrukom primenom operatora selekcije uglavnom loše jedinke izumiru, a bolje ostaju i u sledećem koraku se ukrštaju. Ukrštanjem se prenose osobine roditelja na decu. Mutacijom se menjaju osobine jedinki slučajnom promenom gena. Jedan ovakav postupak omogućuje da iz generacije u generaciju raste prosečan kvalitet populacije. Osnovni koraci genetskog algoritma prikazani su na sledećoj slici (Slika 1.1):

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

Slika 1.1 : Osnovna varijanta genetskog algoritma

1.2.2 Reprezentacija rešenja

Od problema koji se rešava obično zavise samo dve komponente genetskog algoritma: kodiranje problema, tj. reprezentacija rešenja i funkcija cilja.

U prvoj fazi kreiranja genetskog algoritma donosi se odluka o genetskoj reprezentaciji mogućeg rešenja datog problema. Ovo uključuje definisanje jedinki i njihovo pridruživanje mogućim rešenjima. Reprezentacija rešenja može dosta da utiče na efikasnost genetskog algoritma, pa je izbor reprezentacije veoma važan. Međutim, naći "pravu" reprezentaciju za problem koji se rešava jedan je od najtežih delova pravljenja dobrog genetskog algoritma. Uglavnom se polazi od prakse i dobrog poznavanja domena primene.

Za reprezentaciju rešenja obično se koriste binarni brojevi, mada je nekad ipak bolje i jedostavnije koristiti realne ili prirodne brojeve. Sa ciljem da se koristi prirodniji način opisivanja i manipulisanja sa rešenjem, u praksi se, radi testiranja i upoređivanja, može desiti da se koristi više različitih reprezentacija odjednom.

Binarna reprezentacija je jedna od prvih korišćenih i najjednostavnijih reprezentacija. Upravo zbog svoje jednostavnosti pogodna je za implementaciju. Istorijski gledano, mnogi genetski algoritmi su greškom koristili ovu reprezentaciju gotovo odvojeno od problema koje su pokušavali da reše. Ovo je posledica činjenice da binarna reprezentacija nije pogodna za određene problema, zbog čega je neophodno izvršiti izvesne korekcije (nakon ukrštanja i / ili mutacije).

Jedinka se predstavlja nizom binarnih cifara. Dužina ovog niza zavisi od konkretne primene. Znači, jedinka sa k gena se predstavlja vektorom $(x_1,...,x_k)$, gde je $x_i \in \{0,1\}$. Kod izbora funkcije kodiranja specificnog problema, neophodno je da ono bude takvo da sve moguće jedinke mogu da doniraju validno rešenje datog problema, i obrnuto, da sva moguća rešenja mogu da budu reprezentovana. Za neke probleme, kao na primer za probleme sa promenljivima tipa boolean, prirodno je izabrati binarno kodiranje, ali često se niz bitova koristi za kodiranje drugih nebinarnih informacija.

Jedan od problema prilikom kodiranja brojeva u binarne je činjenica da različiti bitovi imaju različitu značajnost. Ovo se moze ublažiti korišćenjem Gray-ovog kodiranja koje je jedna od varijanti kodiranja celih brojeva u niske bitova.

Standardni metod binarnog kodiranja ima jedan nedostatak: Hamingova udaljenost između 2 uzastopna cela broja često je različita od 1. Na primer, imamo za cilj da izvršimo evoluciju celog broja, želimo da mogućnost promene 7 u 8 bude jednaka mogućnosti promene 7 u 6. Međutim, kod binarne reprezentacije, mogućnost promene 0111 u 1000 nije ista kao kod promene 0111 u 0110. Gray-ovo kodiranje je reprezentacija koja garantuje da uzastopni celi brojevi imaju Hamingovu udaljenost jednaku jedinici.

Kao što je već spomenuto, binarna reprezentacija nije uvek odgovarajuća, a posebno ne ako se problem prirodnije predstavlja reprezentacijom u kojoj različiti geni mogu da uzimaju vrednosti iz skupa. U takvim situacijama praktičnije je koristiti celobrojnu reprezentaciju. Jedan očigledan primer ovoga je problem pronalaženja optimalne vrednosti skupa promenljivih koje uzimaju celobrojne vrednosti.

Često najrazumniji način predstavljanja mogućeg rešenja datog problema je pomoću niza realnih brojeva. Ovo se dešava kada vrednosti koje želimo da predstavimo kao gene dolaze iz kontinualne raspodele, a ne iz diskretne. Jedinka sa k gena je sada vektor $(x_1,...,x_k)$, gde je $x_k \in$.

1.2.3 Funkcija cilja

Kao sto je već spomenuto, postoje samo dve komponente genetskog algoritma koje zavise od konkretnog problema: reprezentacija rešenja i funkcija cilja.

Genetski algoritmi, kao i svaka druga metoda optimizacije, zahtevaju neku kvantitativnu meru kvaliteta, tj. ispravnosti predloženog rešenja. Ovu meru koju ćemo zvati *prilagođenost* (na engl. *fitness)* daje funkcija cilja koja se još u literaturi naziva funkcija sposobnosti ili funkcija ocene kvaliteta. Važno je spomenuti da se sve operacije genetskog algoritma sprovode nad jedinkama osim ocene njihovog kvaliteta, tj. ocene prilagođenosti koja se mora odrediti nad dekodiranim jedinkama – potencijalnim rešenjima.

Funkcija cilja je najvažniji deo genetskog algoritma. Ona predstavlja ključ procesa selekcije određujući koje će se jedinke eliminisati, a koje će ostati u populaciji. Na osnovu ove funkcije se određuje koliko je jedinka dobra, tj. računa se njena prilagođenost. Obično, što je prilagođenost jedinke veća, to jedinka ima veću verovatnoću preživljavanja. Znači, jedinke s malom prilagođenošću imaju i malu verovatnoću preživljavanja.

Postoji više načina definisanja funkcije cilja. Ona treba da bude relativno brza jer se primenjuje u svakoj generaciji nad svakim članom populacije. Vredi spomenuti i da je definisanje ove funkcije za zadati problem jedan od najtežih zadataka pravljenja dobrog genetskog algoritma.

1.2.4 Genetski operatori

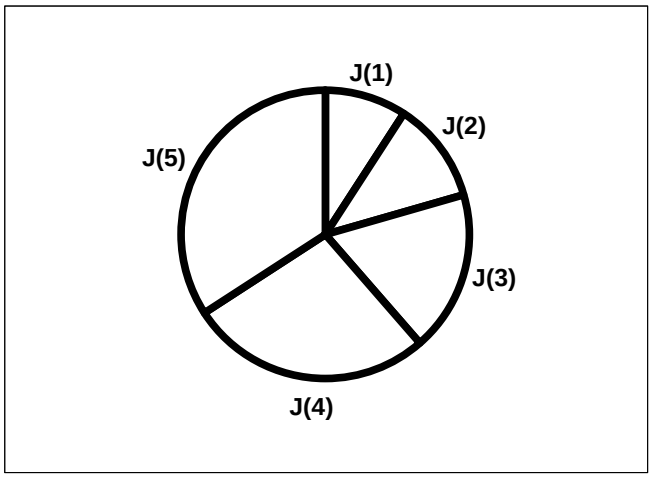
1.2.4.1 Selekcija

Cilj selekcije je čuvanje i prenošenje dobrih osobina na sledeću generaciju jedinki. Selekcijom se biraju *dobre* jedinke koje će učestvovati u sledećem koraku, u ukrštanju. Na taj način se *dobri geni* čuvaju i prenose na sledeće populacije, a loši odumiru. Postupak selekcije bi se mogao ostvariti sortiranjem i izborom N - M najboljih jedinki, gde je N veličina populacije, a M broj jedinki koje će se eliminisati. Veliki nedostaci ovakvog postupka su prerana konvergencija genetskog algoritma (proces optimizacije se završava u samo nekoliko iteracija), kao i gubljenje dobrog genetskog materijala koji mogu sadržati *loše* jedinke. Zbog ovoga je neophodno obezbediti i lošim jedinkama da imaju neku malu verovatnoću preživljavanja. S druge strane, bolje jedinke treba da imaju veću verovatnoću preživljavanja, tj. veću verovatnoću učestvovanja u procesu reprodukcije.

U odnosu na vrstu selekcije, genetske algoritme delimo na generacijske i stacionarne. Generacijski genetski algoritam u jednoj iteraciji raspolaže sa dve generacije, što je ujedno i njegov nedostatak, jer se biraju dobre jedinke stare generacije koje čine novu generaciju i posle selekcije učestvuju u ukrštanju.

Prosta selekcija generiše generaciju koja ima broj jedinki jednak broju jedinki generacije nad kojom se vrši selekcija. Cilj ove selekcije je biranje roditelja čija je verovatnoća selekcije proporcionalna njihovoj prilagođenosti. Obično se za objašnjavanje ove selekcije koristi pojam ruleta, odnosno rulet-točka, zbog čega se prosta selekcija naziva i rulet-selekcija (roulette wheel parent selection). Populacija se posmatra kao da je mapirana na točku (Slika 1.2). Veći delovi točka pripadaju onim jedinkama koje imaju veću prilagođenost. Kada se formira

rulet-točak, vrši se njegovo “okretanje“ onoliko puta koliko će jedinki biti u generaciji. Pri svakom okretanju izvlači se jedna jedinka pomoću pokazivača na rulet-točku. Na ovaj način jedna jedinka se može pojaviti više puta u generaciji. Što je jedinka bolja, to je veća verovatnoća da bude izabrana u svakom izvlačenju. Ovim postupkom se “teoretski“ može dogoditi da u generaciji, pre ukrštanja i mutacije, bude samo jedna jedinka u broju primeraka jednakom veličini polazne populacije.



Slika 1.2 : Rulet-točak sa 5 jedinki

Pojavljivanje duplikata jedinki u generaciji pokazalo se kao veliki nedostatak. Eksperimenti su pokazali da čak 50% od ukupnog broja jedinki mogu biti duplikati, a to samo usporava algoritam.

Problem pojavljivanja duplikata jedinki u novoj populaciji može se rešiti primenom različitih metoda. Najjednostavnije je implicitno brisanje svih višestrukih pojava jedinki iz populacije. To se postiže dodeljivanjem nule prilagođenostima takvih jedinki, čime se sprečava njihovo pojavljivanje u sledećoj generaciji. Ostale tehnike mogu biti:

- stohastičko izvlačenje sa delimičnom zamenom;
- stohastičko izvlačenje ostatka;
- stohastičko izvlačenje ostatka sa zamenom;
- stohastičko izvlačenje ostatka bez zamene;
- stohastičko univerzalno izvlačenje;

Stohastičko izvlačenje sa delimičnom zamenom (Stochastic sampling with partial replacement) prevazilazi problem dupliranja jedinki tako što svakoj izabranoj jedinki smanjuje prilagođenost, a time i udeo u rulet-točku za 1.0.

Kod stohastičkog izvlačenja ostatka (Remainder stohacking sampling) za svaku jedinku čija je prilagođenost veća od 1.0, celobrojni deo prilagođenosti ukazuje na to koliko će se njenih primeraka ubaciti u generaciju. Nakon toga, sve jedinke, pa i one kojima je prilagođenost bila manja od 1.0, ubacuju dodatne primerke u generaciju, sa verovatnoćom proporcionalnom vrednosti necelobrojnog dela prilagođenosti. Prema tome, jedinka sa prilagođenosti 3.56 ubacuje tri svoja primerka direktno u novu populaciju, a nakon toga ima verovatnoću koja je proporcionalna 0.56 da ubaci novi primerak u populaciju.

Kod stohastičkog izvlačenja ostatka sa zamenom (Remainder stochactic sampling with replacement), pravi se rulet-točak nad ostacima. Da bi se prevazišla teoretska mogućnost da jedna jedinka dopuni ostatak generacije koristi se stohastičko izvlačenje ostatka bez zamene (Remainder stochastic sampling without replacement), gde se u drugoj fazi, sa ostacima, jedinka izbacuje sa rulet-točka, nakon što joj je dodeljen jedan primerak na osnovu ostatka.

Stohastičko univerzalno izvlačenje (Stochastic universal sampling) je algoritam koji ima samo jednu fazu i koji se najčešće koristi. Umesto izvlačenja samo jedne jedinke pri svakom okretanju rulet-točka, formira se *N* pokazivača

koji su ravnomerno raspoređeni na obodu točka (*N* je broj jedinki u generaciji). Ovako je dovoljno samo jednom okrenuti točak da bi se dobile sve potrebne jedinke. Na ovaj način se postiže i najbolje, tj. najpoštenije izvlačenje. Vredi pomenuti i da ovakvo izvlačenje zahteva broj operacija reda N, dok ostale metode zahtevaju $N \log N$ operacija.

Turnirska selekcija je zbog svoje jednostavnosti jedna od popularnijih selekcija i sastoji se u izboru jedinki formiranjem turnira. Turniri su takmičenja između jedinki populacije koje se takmiče radi preživljavanja. Broj turnira jednak je broju jedinki. Osnovni parametar turnirske selekcije je veličina turnira N_{tur}, i ona se obično unapred zadaje. Na slučajan način se biraju podskupovi od N_{tur} jedinki koji predstavljaju turnire. Jedinke se u svakom turniru upoređuju sa ciljem biranja najbolje koja će se ubaciti u generaciju.

Veliki nedostatak turnirske selekcije je nemogućnost izbora pogodnog parametra N_{tur}. Često se dešava da izbor jednog parametra dovodi do veoma spore konvergencije, a korišćenje za jedan većeg parametra, težeći ka lokalnom ekstremumu, dovodi do preuranjene konvergencije. Jedno od mogućih unapređenja turnirske selekcije je fino gradirana turnirska selekcija. U ovoj selekciji se koristi racionalni parametar F_{tur} koji označava prosečnu veličinu turnira. Veličina turnira kod fino gradirane turnirske selekcije nije jedinstvena kao kod turnirske selekcije, već se koriste dve veličine turnira, *p* i *q*. Broj turnira veličine p je k_1, a broj turnira veličine q je k_2, pri čemu mora da važi $F_{tur} \approx \frac{p \cdot k_1 + q \cdot k_2}{N_{nnel}}$, gde je $N_{nnel} = k_1 + k_2$. Kao i kod turnirske selekcije, na slučajan način se biraju jedinke za svaki turnir, a jedinka iz jednog turnira se ubacuje u generaciju samo ako je bolja od ostalih jedinki tog turnira.

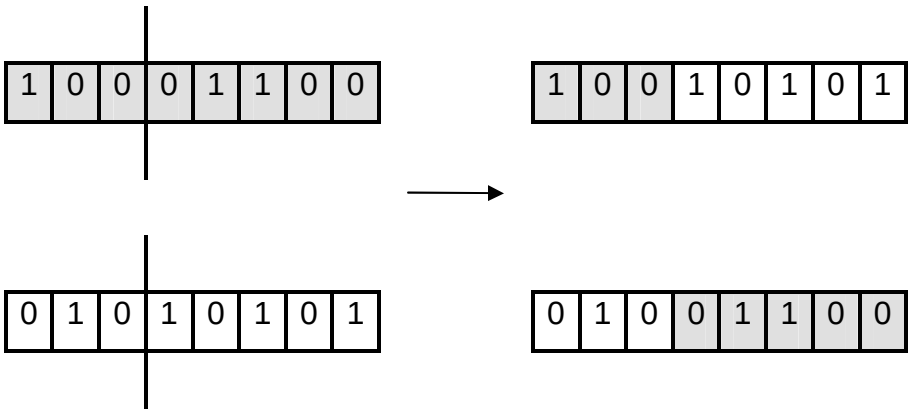
Detaljan opis ove i ostalih tipova selekcije može se naći u [Fil98, Fil06]. Poređenje fino gradirane selekcije sa drugim operatorima i ponašanje ove selekcije u praksi prikazano je u [Fil00, Fil01, Fil03, Fil06].

1.2.4.2 Ukrštanje

Ukrštanje (crossover) je osnovni operator genetskog algoritma koji omogućava stvaranje novih jedinki. Ukrštanje predstavlja razmenu genetskog materijala između dve jedinke analogno istoimenom procesu nad živim jedinkama.

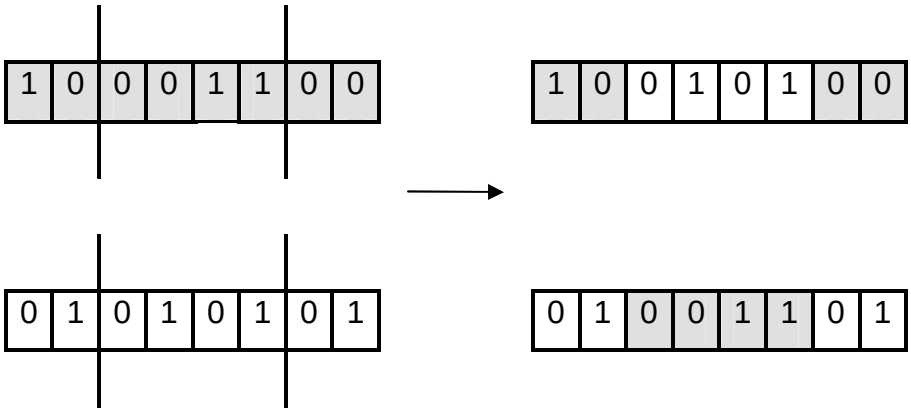
Ukrštanje je binarni operator nad dvema jedinkama koje se nazivaju *roditelji*. Ukrštanjem nastaju jedna ili dve nove jedinke koje se nazivaju *deca*. Najvažnija karakteristika ukrštanja je da deca nasleđuju osobine svojih roditelja. Znači, ako su roditelji dobri, tada će najverovatnije i dete biti dobro, a možda i bolje od svojih roditelja. Postoji više načina ukrštanja definisanih u proizvoljnom broju prekidnih tačaka [Müh97].

Jednopoziciono ukrštanje je najprostiji način ukrštanja jedinki. Na slučajan način bira se ceo broj iz intervala [0, l-1], gde je l dužina jedinke - roditelja. Izabrani ceo broj predstavlja tačku prekida od koje će se vršiti rascep genetskog materijala roditelja. Nove jedinke koje predstavljaju decu nastaju zamenom dobijenih repova (Slika 1.3).



Slika 1.3 : Jednopoziciono ukrštanje

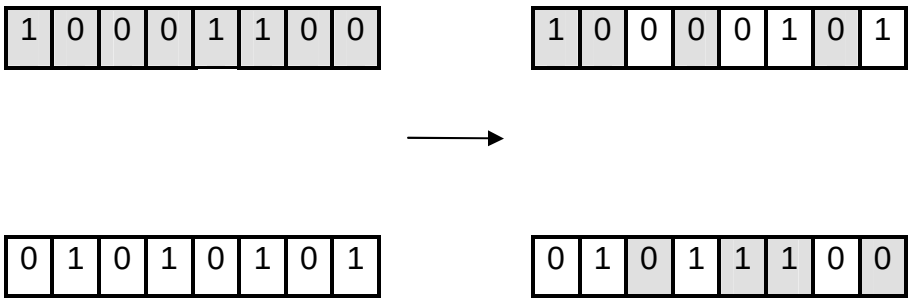
Kako delovi jedinke koji najviše utiču na uspešnost izvršavanja genetskog algoritma ne moraju da se nalaze u susednim genima, uvode se ukrštanja u više tačaka. N-poziciono ukrštanje je prošireno ukrštanje u jednoj tački prekida, s tim što se ovde umesto jedne tačke slučajno bira N tačaka prekida (Slika 1.4).



Slika 1.4 : Dvopoziciono ukrštanje

Uniformno ukrštanje je ukrštanje sa l-1 tačaka prekida, gde je l dužina jedinke, pri čemu je verovatnoća da dete nasledi osobine roditelja $p = 0.5$. Moguće je čak i definisati vektor koji će za svaki gen sadržati verovatnoću (koja može biti različita za svaki gen) da će baš on biti nasleđen od prvog roditelja.

Ovakvo ukrštanje se naziva p-uniformno ukrštanje, a odgovarajući vektor - *maska*. Ako pri ukrštanju nastaju dva deteta, drugo dete nastaje na suprotan način od prvog. Ovo ne znači da je drugo dete negacija prvog, jer ako oba roditelja imaju isti gen, onda će taj isti gen imati i dete na istom mestu (Slika 1.5). Detaljan opis operatora uniformnog ukrštanja se može naći u [Spe91].



Slika 1.5 : Uniformno ukrštanje sa maskom [0.63, 0.36, 0.23, 0.74, 0.46, 0.83, 0.51, 0.49]

Pretpostavlja se da se genetski algoritmi razlikuju od ostalih metoda optimizacije upravo po operatoru ukrštanja. Ovo se ne može reći za operator mutacije koji se može sresti i kod simuliranog kaljenja i kod evolutivnih strategije.

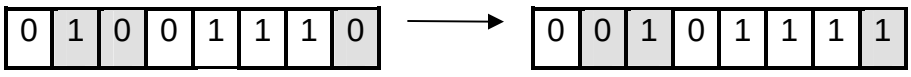
1.2.4.3 Mutacija

Mutacija je najvažniji operator genetskog algoritma čijim delovanjem se vrši izmena slučajno izabranih gena jedinke. Kako deluje nad samo jednom jedinkom, mutacija je unarni operator koji kao rezultat daje izmenjenu jedinku.

Mutacija je operator koji omogućava vraćanje korisnog genetskog materijala koji je izgubljen u selekciji i ukrštanju. Može se reći i da mutacija predstavlja odličan mehanizam za izbegavanje lokalnih ekstremuma pretragom okoline rešenja. Tačnije, ako bi cela populacija završila u nekom od lokalnih ekstermuma, jedino se slučajnom pretragom okoline rešenja pronalazi bolje, te je dovoljno da jedna jedinka nastala mutacijom bude bolja, pa da se cela populacija u nekoliko narednih generacija "preseli" u okolinu boljeg rešenja.

Parametar koji određuje verovatnoću mutacije p_m jednog gena predstavlja nivo mutacije i zadaje se na početku izvršavanja genetskog algoritma. Ako nivo mutacije teži jedinici, algoritam se pretvara u algoritam slučajne pretrage rešenja, a ako teži nuli, može se očekivati zaustavljanje procesa već na samom početku procesa optimizacije u nekom lokalnom ekstremumu. Kod genetskih algoritama, mutacija se obično primenjuje sa niskom verovatnoćom, od 0.001 do 0.01.

Kod binarne reprezentacije najčešće se koristi prosta mutacija. Kod ove mutacije svaki gen se posmatra posebno i svakom bitu koji predstavlja gen dozvoljeno je da se promeni (1→0 ili 0→1) sa malom verovatnoćom p_m. Tačan broj gena koji će se promeniti nije fiksiran i zavisi od nasumično izabranih brojeva. Prema tome, za kodiranje dužine l, u proseku $p_m \times l$ gena će biti promenjeno (Slika 1.6).



Slika 1.6 : Primer proste mutacije (nasumično izabrani, drugi, treći i osmi bit mutiraju)

Može se desiti da skoro sve jedinke u jednoj generaciji imaju isti gen na određenom mestu. Takve gene nazivamo "*zaleđenim*" genima i može ih biti više u generaciji. Pojavljivanje ovih gena uzrokuje smanjivanje pretraživačkog prostora, a samim tim i mogućnost preuranjene konvergencije, što predstavlja veliki nedostatak u izvršavanju genetskog algoritma. Ovi geni se selekcijom i ukrštanjem ne mogu promeniti (ako oba roditelja imaju isti gen, onda će taj isti gen imati i dete). Jedino mutacija može popraviti ovaj nedostatak. Kako se prevelikim nivoom mutacije genetski algoritam ponaša kao algoritam slučajne pretrage, a kako mali nivo mutacije ne može uticati na veliko povećanje prostora pretrage, to se koristi mutacija sa povećanim nivoom mutacije samo na zaleđenim genima.

Kod proste mutacije ispituje se bit po bit i za svaki bit se određuje da li će mutirati ili ne. Kako je nivo mutacije mali, broj gena koji će mutirati je takođe mali (ako je $p_m = 0.01$, jedan od sto bitova će biti promenjen), pa uvek nije neophodno ispitivati ceo genetski materijal. Proces mutacije može se ubrzati korišćenjem binomne ili normalne raspodele. Kod mutacije pomoću binomne raspodele broj mutiranih gena ima binomnu raspodelu $B(N_{bitova}, p_m)$, gde je N_{bitova} broj bitova koji predstalja jedinku, tj. dužina genetskog koda jedinke, a p_m nivo mutacije. Prvo se bira slučajan broj $q \in [0,1]$, a zatim se pronalazi n_m takvo da važi $F(n_m) \le F(q) < F(n_m + 1)$, gde je n_m broj gena koji će mutirati, a F funkcija binomne raspodele, a zatim se na slučajan način bira n_m gena koji će mutirati. Ako je proizvod dužine genetskog koda jedinke i nivoa mutacije dovoljno veliki, binomna raspodela se aproksimira normalnom raspodelom $N(N_{bitova} \cdot p_m; N_{bitova} \cdot p_m \cdot (1 - p_m))$. Detaljnije informacije o navedenim operatorima mutacije mogu se naći u [Kra00, Toš04].

U slučajevima kada geni nisu međusobno ravnopravni, tj. kada je neke gene potrebno mutirati sa većim ili manjim nivoom mutacije, koristi se normalna mutacija sa normalnom raspodelom ili eksponencijalna mutacija kod koje broj gena u jedinki koji mutiraju eksponencijalno raste. Detaljnije o ovim operatorima može se naći u [BeD93a, BeD93b].

1.2.5 Uslov zaustavljanja

Nakon generisanja početne populacije, nad njom se izvršava proces koji se sastoji od selekcije, ukrštanja i mutacije dok se ne zadovolji uslov zaustavljanja. Ovaj uslov definiše i sam kraj genetskog algoritma. Kako su genetski algoritmi stohastička metoda pretrage, teško je formalno definisati uslov zaustavljanja.

Kada se zadovolji uslov zaustavljanja, rešenje genetskog algoritma ne mora predstavljati optimalno rešenje problema koji se rešava. Najčešće primenjivani uslovi zaustavljanja genetskog algoritma su:

- dostignut maksimalan broj generacija,
- sličnost jedinki u populaciji,
- najbolja jedinka ponovljena određen broj puta,
- dostignuto unapred zadato optimalno rešenje,
- dokazana optimalnost najbolje jedinke (ako je to moguće),
- ograničeno vreme izvršavanja genetskog algoritma,
- prekid od strane korisnika itd.

1.2.6 Ostali aspekti genetskog algoritma

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

- Generacijska – u svakoj generaciji menjaju se sve jedinke populacije.
- Stacionarna – u svakoj generaciji generiše se samo deo populacije dok se preostale jedinke prenose iz prethodne populacije.
- Elitistička strategija – Postoji opasnost da se dobro rešenje dobijeno nakon puno iteracija izgubi izmenom od strane genetskih operatora mutacije ili selekcije. Zbog toga se javlja potreba da se najbolje jedinke zaštite od izmene ili eliminacije tokom evolutivnog procesa. Takav mehanizam se naziva *elitizam*. Genetski algoritam sa ugrađenim elitizmom, iz generacije u generaciju, asimptotski teži ka rešenju problema - globalnom ekstremumu. Međutim, da bi se u svakom koraku evolucije zaštitila najbolja jedinka od izmena ili eliminacije, potrebno ju je prethodno pronaći. Pretraživanje ili sortiranje zahteva dodatno procesorsko vreme zbog čega se može znatno usporiti genetski algoritam.

Genetski algoritmi često se kombinuju sa drugim heuristikama, koje se mogu koristiti kako za poboljšavanje početne populacije tako i svake naredne generacije, primenom na celu populaciju, jedan njen deo ili na pojedinačnu jedinku. Performanse genetskog algoritma bitno se mogu poboljšati paralelizacijom i keširanjem, o čemu se detaljnije može videti u [Kra00].

Genetski algoritmi su se pokazali veoma uspešnim pri rešavanju široke klase problema optimizacije. Veliki broj primena genetskih algoritama može se naći u [Ala08]. Neke od novijih primena su:

- diskretni lokacijski problemi [Alp03, Dre03, Dvo99, Fil00, Kra98, Kra99b, Kra00, Kra01a, Kra01b, Mar09, Mar08a, Mar08b, 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, Ognj04],
- višedimenzioni problem ranca [Puc06, Rai05],
- maksimalno balansirana povezana particija grafa [Đur08],
- izbor indeksa baze podataka [Kra03],
- rutiranje vozila [Kra08e],
- pokrivanje čvorova grafa [Mil08],
- bojenje grafa [Juh06].

2 Lokacijski problem snabdevača ograničenog kapaciteta u više nivoa

2.1 Pregled problema lokacije snabdevača ograničenog kapaciteta

Lokacijski problem snabdevača ograničenog kapaciteta (The Capacitated Facility Location Problem – CFLP) veoma je važan problem u teoriji lokacijskih problema. Zadatak ovog problema je minimizovati sumu fiksnih troškova i troškova transporta, pri čemu je neophodno poštovati zahteve skupa klijenata i ograničenost kapciteta. Ovo se ostvaruje pažljivim izborom potencijalnih lokacija iz skupa mogućih lokacija. Više o problemima ove vrste može se naći u [Fab04, Zha06]. Za razliku od osnovnog modela, hijerarhijski model ovog problema do sada nije razmatran.

U literaturi se pojavljuje i lokacijski problem snabdevača ograničenog kapaciteta sa jednim izvorom (Single Source Capacitated Facility Location Problem - SSCFLP). Zadatak ovog problema je minimizovati sumu fiksnih troškova i troškova transporta, pri čemu se svaki klijent ili lokacija snabdeva iz tačno jedne lokacije prethodnog nivoa. Više o ovom problemu može se naći u [Ahu02, Kum05, Sca04].

2.2 Matematička formulacija problema

Lokacijski problem snabdevača ograničenog kapaciteta (CFLP) sa samo jednim nivoom je NP-težak problem kao uopštenje lokacijskog problema

snabdevača neograničenog kapaciteta (The Uncapacitated Facility Location Problem - UFLP) za koji je poznato da pripada klasi NP-teških problema. Jedan od dokaza ovog tvrđenja može se naći u preglednom radu [Krr83]. Kako je lokacijski problem snabdevača ograničenog kapaciteta u više nivoa (MLCFLP) generalizacija CFLP, ovaj problem je takođe NP-težak problem.

Neka je *F* skup potencijalnih lokacija (kardinalnosti |*F*|=m) koje su podeljene u n_l nivoa označenih sa $F_1,...,F_{n_l}$ ($F_1 \cup F_2 ... \cup F_{n_l-1} \cup F_{n_l} = F$), *D* skup klijenata (kardinalnosti |*D*|=n), f_i vrednosti fiksnih troškova za uspostavljanje lokacije $i \in F$, d_{ik} vrednost transportnih troškova po jedinici robe od *i* do *k* za svaki $i,k \in F \cup D$, b_i kapacitet lokacije $i \in F$ i c_j količina robe koju zahteva korisnik $j \in D$.

Promenljive y_i i x_{ik} koje figurišu u formulaciji problema mogu se definisati tako da važi:

1. $y_i = 1$ za $i \in F$, ako je uspostavljena lokacija *i,*
2. $y_i = 0$ za $i \in F$,ako lokacija *i* nije uspostavljena,
3. $x_{ik} =$ *količina robe koja iz lokacije k dolazi do klijenta ili lokacije i,* $x_{ik} \in \ ^{+} \cup \{0\}$.

Model celobrojnog linearnog programiranja za rešavanje lokacijskog problema snabdevača neograničenog kapaciteta iz rada [Kra09]:

Minimizovati:

$$\min \sum_{i \in F} f_i y_i + \sum_{l=2}^{n_l+1} \sum_{i \in F_l} \sum_{k \in F_{l-1}} d_{ik} x_{ik} \qquad (2.1)$$

tako da važi:

$$\sum_{k\in F_{nl}} x_{ik} = 1, \qquad \text{pri čemu je } i \in D \tag{2.2}$$

$$\sum_{k\in F_{l-1}} x_{ik} = \sum_{j\in F_{l+1}} x_{ji}\ , \qquad (l = 2,...,n_l,\ \ i \in F_l) \tag{2.3}$$

$$x_{ik} \le n \cdot y_k\ , \qquad (k \in F_{l-1},\ \ i \in F_l,\ \ l = 2,...,n_l + 1) \tag{2.4}$$

može se na relativno jednostavan način dopuniti tako da predstavlja model mešovitog celobrojnog linearnog programiranja za rešavanje MLCFLP:

Minimizovati:

$$\min \sum_{i\in F} f_i y_i + \sum_{l=2}^{n_l+1} \sum_{i\in F_l} \sum_{k\in F_{l-1}} d_{ik} x_{ik}\ , \tag{2.5}$$

tako da važi:

$$\sum_{k\in F_{nl}} x_{ik} = c_i\ , \qquad \text{pri čemu je } i \in D \tag{2.6}$$

$$\sum_{k\in F_{l-1}} x_{ik} = \sum_{j\in F_{l+1}} x_{ji}\ , \qquad (l = 2,...,n_l,\ \ i \in F_l) \tag{2.7}$$

$$x_{ik} \le b_k y_k\ , \qquad (k \in F_{l-1},\ \ i \in F_l,\ \ l = 2,...,n_l + 1) \tag{2.8}$$

$$\sum_{k\in F_{l-1}} x_{ik} \le b_i y_i\ , \qquad (l = 2,...,n_l,\ \ i \in F_l). \tag{2.9}$$

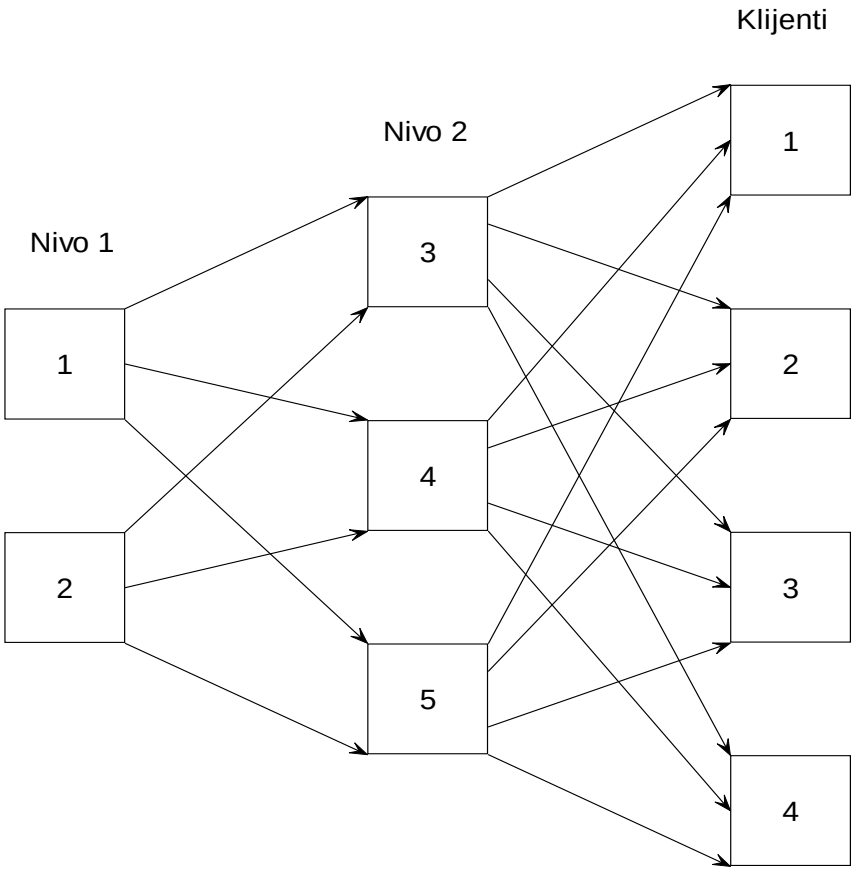
Optimalno rešenje problema je takvog rešenje da je suma fiksnih troškova za uspostavljanje lokacija i troškova transporta minimalna, a dobija se

minimizacijom funkcije cilja (2.5). Troškovi transporta predstavljaju sumu troškova transporta od lokacije na prvom nivou preko svih nivoa do svakog korisnika. Uslov (2.6) obezbeđuje da svaki klijent dobije upravo onu količinu robe koju je zahtevao, dok je uslovom (2.7) obezbeđeno da svaka lokacija prethodnog nivoa (osim prvog nivoa na kom su proizvođači) snabdeva upravo onom količinom robe koja je potrebna za snabdevanje lokacija naredog nivoa koje se snabdevaju iz odgovarajuće lokacije tekućeg nivoa. Uslovom (2.8) onemogućeno je da se iz neke lokacije prethodnog nivoa dopremi količina robe koja je veća od samog kapaciteta te lokacije, a uslovom (2.9) onemogućeno je da se ukupna količina robe iz odgovarajućih lokacija prethodnih nivoa dopremi do lokacije sa manjim kapacitetom od te ukupne količine. Uslovima (2.8) i (2.9) takođe je onemogućeno snabdevanje lokacija koje nisu uspostavljene.

Kao korektno rešenje ovog problema, u zavisnosti od količine robe koju zahteva klijent i kapaciteta lokacije, svakom klijentu se pridružuje jedan ili više nizova od n_l potencijalnih lokacija. Svaki niz potencijalnih lokacija koji se pridružuje korisniku mora biti takav da potencijalne lokacije moraju biti na različitim nivoima. Važno je napomenuti da na svakom nivou mora biti uspostavljena bar jedna lokacija. Ako se npr. klijent *j* snabdeva preko lokacija $i_{n_l}, i_{n_l-1}, ..., i_1$, gde je i_k uspostavljena lokacija *k*-tog nivoa, tada su troškovi transporta njegovog snabdevanja jednaki $d_{ji_{n_l}} + d_{i_{n_l}i_{n_l-1}} + ... + d_{i_2 i_1}$. U slučaju da je količina robe koju klijent zahteva veća od kapaciteta lokacija, njemu se pridružuje više nizova potencijalnih lokacija preko kojih se snabdeva, a ukupni troškovi transporta jednaki su sumi troškova transporta snabdevanja preko tih lokacija.

Primer 2.1: Neka je dato pet potencijalnih lokacija raspoređenih u dva nivoa, neka su u prvom dve, a u drugom nivou tri lokacije i neka se zahteva opsluživanje četiri klijenta. Na slici 2.1 prikazane su sve mogućnosti povezivanja

klijenata sa lokacijama drugog nivoa, kao i lokacija drugog nivoa sa lokacijama prvog nivoa.



Slika 2.1 : Moguće veze između klijenata i potencijalnih lokacija

U Tabeli 2.1 date su vrednosti fiksnih troškova za uspostavljanje lokacija, a u Tabelama 2.2 i 2.3 dati su kapaciteti tih lokacija i odgovarajuće količine zahtevane od strane korisnika. Tabela 2.4 sadrži rastojanja između lokacija na prvom i drugom nivou, a Tabela 2.5 sadrži rastojanja između klijenata i lokacija na drugom nivou.

Tabela 2.1 : Fiksni troškovi

Lokacije	Lok. 1	Lok.2	Lok. 3	Lok. 4	Lok. 5
Fiksni troškovi	30	30	20	20	20

Tabela 2.2 : Kapaciteti lokacija

Lokacije	Lok. 1	Lok.2	Lok. 3	Lok. 4	Lok. 5
Kapaciteti	20	20	10	10	10

Tabela 2.3 : Količine robe koju zahtevaju klijenti

Klijenti	Klijent 1	Klijent 2	Klijent 3	Klijent 4
Količine robe	2	8	3	5

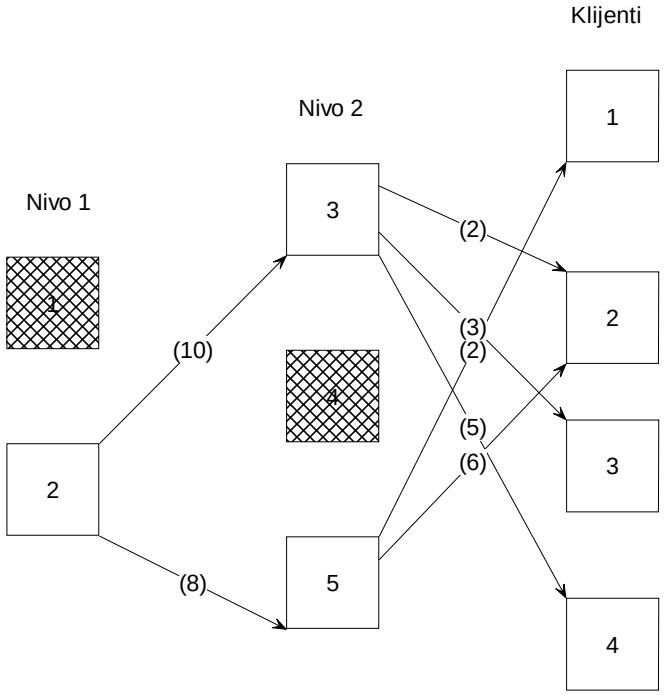
Tabela 2.4 : Troškovi transporta između lokacija na prvom i drugom nivou

	Lok. 1	**Lok. 2**
Lok. 3	15	14
Lok. 4	17	18
Lok. 5	13	12

Tabela 2.5 : Troškovi transporta između klijenata i lokacija na drugom nivou

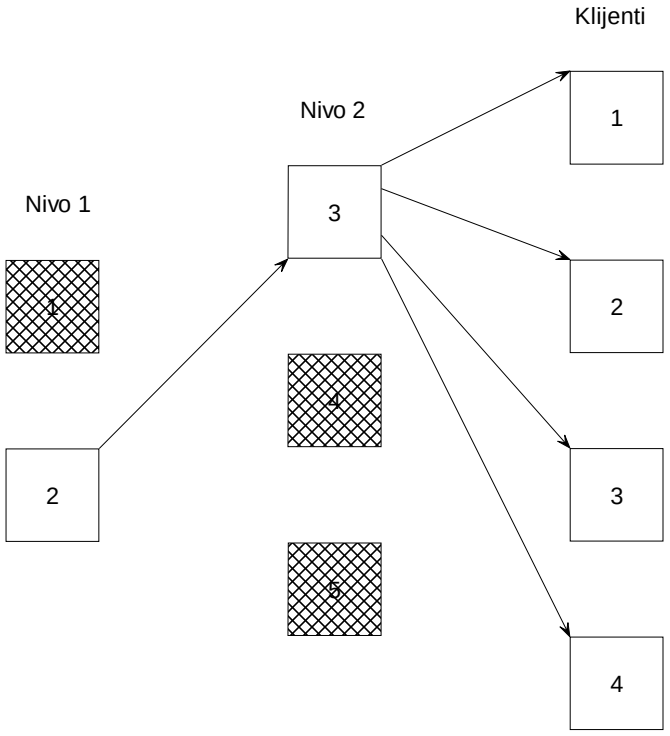
	Lok. 3	**Lok. 4**	**Lok. 5**
Klijent 1	4	7	5
Klijent 2	2	3	7
Klijent 3	1	4	6
Klijent 4	5	8	9

Optimalno rešenje ovako formulisanog problema dobijeno je korišćenjem CPLEX programskog paketa koji će biti opisan u jednoj od narednih glava i iznosi 139.194. Lokacije koje su uspostavljene su Lok.2 sa prvog nivoa i Lok.3 i Lok.5 sa drugog nivoa. Na Slici 2.2 prikazane su uspostavljene veze između klijenata i uspostavljenih lokacija sa količinama robe koje se iz lokacija dopremaju do klijenata.



Slika 2.2 : Uspostavljene veze između klijenata i lokacija sa količinama robe

Dobijeno optimalno rešenje se razlikuje od optimalnog rešenja lokacijskog problema snabdevača neograničenog kapaciteta u više nivoa (MLUFLP), formulisanog na isti način. Optimalno rešenje ovog problema je takođe dobijeno pomoću CPLEX programskog paketa i iznosi 118.000, a na Slici 2.3 su pretstavljene uspostavljene lokacije, Lok.2 sa prvog nivoa i Lok.3 sa drugog nivoa.



Slika 2.3 : Uspostavljene veze između klijenata i lokacija
pri rešavanju MLUFLP-a

3 Genetski algoritam za rešavanje MLCFLP

3.1 Reprezentacija i funkcija cilja

Pri rešavanju MLCFLP-a, kako razmatranje svih mogućih rešenja zahteva veliku prostornu i vremensku složenost izračunavanja, u cilju lakšeg računanja funkcije cilja, osnovna ideja je eliminisati što je moguće veći broj potencijalnih rešenja. Ovo se ostvaruje kombinacijom linearnog programiranja i programskog paketa CPLEX. Pomoću odgovarajućeg genetskog algoritma se uspostavljaju potencijalne lokacije koje se fiksiraju, a zatim se funkcija cilja računa pomoću CPLEX-a.

U implementaciji genetskog algortma primenjena je binarna reprezentacija jedinki populacije. Binarnim nizom dužine m (broj potencijalnih lokacija) predstavlja se skup svih potencijalnih lokacija. Za i-ti element ovog niza važi da je jednak jedinici ako je i-ta lokacija uspostavljenja, tj. nuli ako lokacija nije uspostavljena. Kako na svakom nivou mora biti uspostavljena bar po jedna lokacija, proverava se da li je ovo svojstvo zadovoljeno, a onda se funkcija cilja računa pomoću CPLEX-a. Mana ovakvog načina rešavanja algoritma je ta što je algoritam prilično usporen jer se za svaku jedinku u svakoj generaciji poziva CPLEX. Međutim, sada je dati potproblem dobijen fiksiranjem uspostavljenih lokacija problem samo linearnog programiranja (ne i mešovitog celobrojnog), pa se rešava relativno jednostavno.

3.2 Genetski operatori

3.2.1 Selekcija

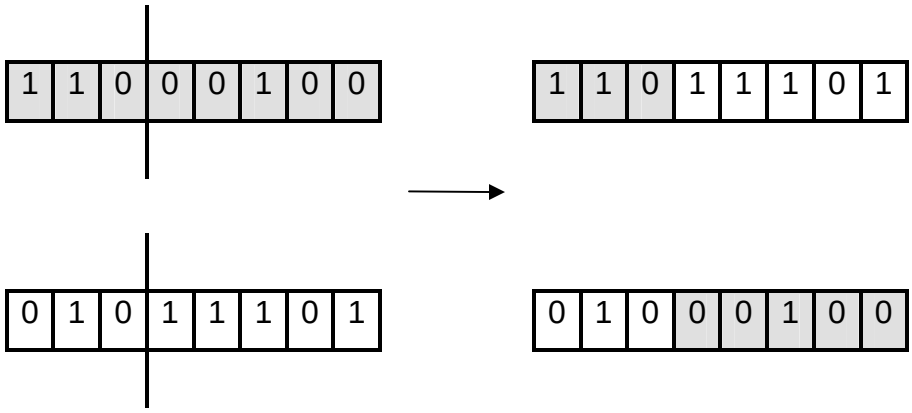
Selekcijom se biraju *dobre* jedinke sa ciljem čuvanja i prenošenja dobrih osobina na sledeću generaciju. Na taj način se *dobri geni* čuvaju i prenose na sledeće populacije, a loši odumiru.

Pri rešavanju ovog problem koristi se fino gradirana turnirska selekcija – unapređenje turnirske selekcije. Turniri su takmičenja između jedinki populacije koje se takmiče radi preživljavanja. Broj turnira jednak je broju jedinki, a veličina turnira N_{tur} se obično unapred zadaje. Jedinke se u svakom turniru upoređuju i bira se najbolja koja će se učestvovati u ukrštanju. Veliki nedostatak turnirske selekcije je nemogućnost izbora pogodnog parametra N_{tur}. Često se dešava da izbor jednog parametra dovodi do veoma spore konvergencije, a korišćenje za jedan većeg parametra težeći ka lokalnom ekstremumu dovodi do preuranjene konvergencije. Upravo zbog ovoga se pri rešavanju MLCFLP koristi fino gradirana turnirska selekcija [Fil00, Fil03]. U ovoj selekciji se koristi racionalni parametar F_{tur} koji označava prosečnu veličinu turnira. Veličina turnira kod fino gradirane turnirske selekcije nije jedinstvena kao kod turnirske selekcije, već se koriste dve veličine turnira *p* i *q*. Broj turnira veličine p je k_1, a broj turnira veličine q je k_2, pri čemu mora da važi $F_{tur} \approx \frac{p \cdot k_1 + q \cdot k_2}{N_{nnel}}$, gde je $N_{nnel} = k_1 + k_2$ broj ne-elitnih jedinki, odnosno broj turnira koje treba organizovati. Na slučajan način se biraju jedinke za svaki turnir, a najbolja jedinka iz jednog turnira će učestvovati u ukrštanju. U ovoj implementaciji je F_{tur} = 5.4, dok su k_1 = 30 i k_2 = 20 za *N* = 50 ne-elitnih jedinki.

3.2.2 Ukrštanje

Jedinke koje su odabrane u toku selekcije dalje učestvuju u ukrštanju. Ukrštanje je binarni operator nad dvema jedinkama koje se nazivaju *roditelji*. Ukrštanjem nastaju jedna ili dve nove jedinke koje se nazivaju *deca*. Najvažnija karakteristika ukrštanja je da deca nasleđuju osobine svojih roditelja.

U implementaciji genetskog algoritma kojim se rešava ovaj problem korišćeno je jednopoziciono ukrštanje. Ovo ukrštanje predstavlja najprostiji način ukrštanja jedinki. Na slučajan način bira se ceo broj iz intervala [0, l -1], gde je l dužina jedinke – roditelja. Izabrani ceo broj predstavlja tačku prekida od koje će se vršiti rascep genetskog materijala roditelja. Zamenom dobijenih repova nastaju nove jedinke – deca (Slika 2). Pri rešavanju ovog problema korišćen je nivo ukrštanja od 0.85, a to znači da 85% jedinki učestvuje u ukrštanju.



Slika 2.3: Primer jednopozicionog ukrštanja

3.2.3 Mutacija

Mutacija je najvažniji operator genetskih algoritama čijim delovanjem se vrši izmena slučajno izabranih gena jedinke i koji omogućava vraćanje korisnog genetskog materijala koji je izgubljen u selekciji i ukrštanju. Kako deluje nad samo jednom jedinkom, mutacija je unarni operator koji kao rezultat daje izmenjenu jedinku. Parametar koji određuje verovatnoću mutacije p_m jednog gena predstavlja nivo mutacije. U implementaciji genetskog algoritma, osnovni nivo mutacije je $p_m = 0.4 / m$.

Može se desiti da skoro sve jedinke u jednoj generaciji imaju isti gen na određenom mestu. Takve gene definišemo kao "*zaleđene*" gene i njihovo pojavljivanje uzrokuje smanjivanje pretraživačkog prostora, a samim tim i mogućnost preuranjene konvergencije. Ovi geni posle selekcije i ukrštanja ostaju isti i jedino ih mutacija može promeniti. Kako se prevelikim nivoom mutacije genetski algoritam ponaša kao algoritam slučajne pretrage, a kako mali nivo mutacije ne može uticati na veliko povećanje prostora pretrage, da bi se ovo sprečilo, u implementaciji genetskog algoritma se koristi mutacija sa 2.5 puta većim nivoom mutiranja zaleđenih gena, odnosno 1.0/*m*.

3.3 Ostali aspekti genetskog algoritma

U implementaciji genetskog algoritma korišćena je populacija od 150 jedinki generisanih na slučajan način. Što se politike zamene generacija tiče, ovde je, u cilju skraćivanja vremena izvršavanja algoritma i čuvanja dobrih rešenja, primenjena elitistička strategija. Ovom strategijom se bez primene genetskih operatora i računanja funkcije prilagođenosti u svaku generaciju propušta određen broj dobro prilagođenih jedinki, takozvanih elitnih jedinki. U

svakoj generaciji 2/3 (100 jedinki) populacije se propušta u narednu generaciju, dok se 1/3 (50 jedinki) dobija primenom genetskih operatora.

Kako može dovesti do preuranjene konvergencije, pojavljivanje duplikata jedinki pokazalo se kao veliki nedostatak. Pri rešavanju MLCFLP ovo se rešava implicitno, postavljanjem prilagođenosti svih višestrukih pojavljivanja jedinki na nulu (osim na prvu), čime se sprečava njihovo dalje pojavljivanje u narednim generacijama. Takođe, može se desiti da prilagođenosti različitih genetskih kodova jedinki budu jednake, što se sprečava ograničavanjem broja pojavljivanja ovakvih jedinki na $N_{ifc} = 40$.

3.4 Keširanje

U ovoj implementaciji genetskog algoritma, radi poboljšanja performansi genetskog algoritma, primenjena je tehnika keširanja, koja je prvi put primenjena u [Kra99a]. Ova tehnika ne utiče na rezultate koji se dobijaju, već samo smanjuje vreme izvršavanja genetskog algoritma.

Kod pojavljivanja istih genetskih kodova jedinki, u toku izvršavanja genetskog algoritma, često je pogodnije zapamtiti date genetske kodove i njihove prilagođenosti u prvom pojavljivanju, a zatim u sledećim pojavljivanjima direktno očitati vrednosti, umesto ponovnog računanja. Tehnika keširanja zasnovana je na strategiji najstarijeg nekorišćenog bloka (Last Recently Used – LRU). Ova strategija implementirana je pomoću heš-red tabele u kojoj se čuvaju izračunate prilagođenosti. Konkretno, pri rešavanju ovog problema broj keširanjih vrednosti funkcije cilja u heš-red tabeli ograničen je na $Ncache = 5000$.

4 Eksperimentalni rezultati

U ovom delu prikazani su dobijeni rezultati genetskog algoritma za rešavanje lokacijskog problema snabdevača ograničenog kapaciteta u više nivoa. Sva testiranja su vršena na računaru sa Intel-ovim procesorom radnog takta 1.46 GHz koji ima 2056 MB radne memorije. Algoritam je kodiran na programskom jeziku C.

4.1 Test primeri

Genetski algoritam je testiran na instancama iz rada [Mar09]. Instance u ovom radu su dobijene modifikovanjem instanci iz ORLIB biblioteke (The Imperial College OR library) [Bea90, Bea96]. Ova biblioteka sadrži instance koje su veoma pogodne za testiranje mnogih problema operacionog istraživanja, među kojima je i lokacijski problem snabdevača ograničenog kapaciteta. Kako ORLIB instance imaju samo jedan nivo hijerarhije, tj. dizajnirane su samo za probleme sa jednim nivoom lokacija, one su modifikovane generisanjem rastojanja između lokacija na susednim nivoima, kao i rastojanja između klijenata i lokacija na poslednjem nivou. Kako su transportni troškovi u [Mar09] dati zbirno, a u MLCFLP moraju biti zadati po jedinici robe, u svim instancama su transportni troškovi podeljeni sa prosečnom količinom robe.

Što se naziva instanci tiče, oni čuvaju informacije o originalnoj instanci od koje su nastali, broju nivoa potencijalnih lokacija, broju potencijalnih lokacija na svakom nivou i broju klijenata. Tako je, na primer, instanca capa_3n_15_30_55.1000 nastala modifikovanjem ORLIB instance capa, ima 3 nivoa sa po 15, 30 i 55 lokacija i 1000 klijenata.

4.2 Rezultati CPLEX-a

Na osnovu formula (2.5) – (2.9) koje predstavljaju matematičku formulaciju MLCFLP-a, implementiran je CPLEX program za pronalaženje optimalnog rešenja instanci malo većih dimenzija. Program je testiran na CPLEX 10.1 programskom paketu i dobijeni rezultati su prikazani u Tabeli 4.1. Kako je za svaku instancu CPLEX završio izračunavanje bez prekidanja, sva dobijena rešenja su optimalna.

Tabela 4.1 : Rezultati CPLEX-a

Ime instance	Optimalna vrednost	Vreme izvršavanja (u sekundama)
cap71_2n_6_10.50	1855797.443313	0.0780
cap71_3n_2_5_9.50	4730369.720843	0.0940
cap101_2n_8_17.50	1700596.267026	0.3590
cap101_3n_3_7_15.50	3254576.460557	0.1560
cap131_2n_13_37.50	1927350.876832	0.3900
cap131_3n_6_14_30.50	3320490.188191	0.4050
cap131_4n_3_7_15_25.50	4082096.250000	0.1870
capa_2n_30_70.1000	15020155.885894	8.9390
capa_3n_15_30_55.1000	25806183.139332	13.8680
capa_4n_6_12_24_58.1000	35873713.526558	10.7170
capb_2n_35_65.1000	14766566.334533	14.5060
capb_3n_12_25_63.1000	26545701.486854	18.8890
capb_4n_6_13_31_50.1000	42203154.938342	12.6850
capc_2n_32_68.1000	14347387.387349	15.8550
capc_3n_13_27_60.1000	27018609.591453	13.5490
capc_4n_4_9_27_60.1000	47605598.892787	20.7510

4.3 Rezultati genetskog algoritma

Uslovi zaustavljanja genetskog algoritma koji su ovde bili primenjeni su maksimalan broj generacija N_{gen} = 500 i ponavljanje najboljeg rešenja funkcije cilja u N_{rep} = 200 uzastopnih generacija. Zbog nedeterminističke prirode rezultata genetskog algoritma, za svaku instancu algoritam je izvršen 20 puta. U Tabeli 4.2 su prikazani rezultati dobijeni na modifikovanim instancama iz ORLIB biblioteke.

Prva kolona u tabeli sadrži ime instance. U drugoj koloni je prikazana vrednost optimalnog rešenja odgovarajuće instance. U sledećoj koloni GA_{naj} su prikazana optimalna rešenja dobijena primenom genetskog algoritma na odgovarajućoj instanci. Prikazane su dobijene vrednosti, odnosno oznaka *opt* - ukoliko algoritam dostiže optimalno rešenje poznato u praksi (dobijeno primenom CPLEX programskog paketa). U koloni *t* je prikazano prosečno vreme potrebno da se dođe do optimalnog rešenja genetskog algoritma, dok je u koloni t_{tot} prikazano ukupno vreme potrebno da genetski algoritam završi sa radom. U koloni *gen* prikazano je nakon koliko generacija genetski algoritam završava sa radom. U svih 20 izvršavanja se kvalitet rešenja ocenjuje relativnom greškom izraženom u procentima (prg), koja predstavlja odnos Opt_{re} i GA_{naj}. Relativna greška rešenja izražena u procentima se računa po formuli $prg = \frac{1}{20}\sum_{i=1}^{20} rg_i$, gde je rg_i relativna greška u i-tom izvršavanju genetskog algoritma. Vrednost rg_i se računa po formuli $rg_i = 100 * \frac{GA_i - Opt_{re}}{Opt_{re}}$. GA_i predstavlja rešenje dobijeno u *i*-tom izvršavanju. Kvalitet rešenja meri se i standardnom devijacijom relativne greške σ za rg_i (i = 1, …, 20), koja se računa po formuli $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(rg_i - prg)^2}$. Kolona *eval* prikazuje prosečan broj izračunavanja, a kolona *keširanje* prikazuje procentualnu uštedu korišćenjem tehnike keširanja.

Pomoću genetskih algoritama ne može da se dokaže optimalnost rešenja i ne postoji zadovoljavajući efektivni kriterijum zaustavljanja. Kolona t_{tot} prikazuje da se, iako je optimalno rešenje već dostignuto, genetski algoritam izvršava sve dok se ne zadovolji kriterijum zaustavljanja, tj. i za dodatno vreme $t_{tot} - t$.

Tabela 4.2 Rezultati genetskog algoritma

Ime instance	Opt_{re}	GA_{naj}	T sec	t_{tot} sec	*gen*	*prg* %	σ %	*eval*	*keširanje* %
cap71_2n_6_10.50	1855797.443313	*Opt*	1.026	3.391	210.9	0.000	0.000	1774.8	83.4
cap71_3n_2_5_9.50	4730369.720843	*Opt*	1.592	3.507	211.4	0.000	0.000	1738.5	83.8
cap101_2n_8_17.50	1700596.267026	*Opt*	1.934	6.318	215.8	0.000	0.000	2341.6	78.6
cap101_3n_3_7_15.50	3254576.460557	*Opt*	1.954	5.527	218.6	0.000	0.000	2419.7	78.2
cap131_2n_13_37.50	1927350.876832	*Opt*	8.341	22.502	242.3	0.000	0.000	4376.9	64.3
cap131_3n_6_14_30.50	3320490.188191	*Opt*	7.936	21.353	235.7	0.000	0.000	4172.7	65.0
cap131_4n_3_7_15_25.50	4082096.250000	*Opt*	5.364	15.301	230.4	0.000	0.000	4161.7	64.3
capa_2n_30_70.1000	15020155.885894	*Opt*	1967.763	3340.677	376.5	0.002	0.004	10009.0	47.1
capa_3n_15_30_55.1000	25806183.139332	*Opt*	1228.541	2149.086	341.9	0.036	0.028	8599.9	50.0
capa_4n_6_12_24_58.1000	35873713.526558	*Opt*	992.541	1767.115	314.1	0.005	0.008	7866.4	50.3
capb_2n_35_65.1000	14766566.334533	*Opt*	2229.400	3860.953	393.9	0.024	0.053	10480.5	47.1
capb_3n_12_25_63.1000	26545701.486854	*Opt*	1251.188	2230.703	330.2	0.017	0.075	8581.6	48.5
capb_4n_6_13_31_50.1000	42203154.938342	*Opt*	978.185	1772.987	310.1	0.000	0.000	7835.6	50.0
capc_2n_32_68.1000	14347387.387349	*Opt*	1707.108	3177.074	353.2	0.015	0.026	9517.9	46.5
capc_3n_13_27_60.1000	27018609.591453	*Opt*	1420.852	2507.515	336.4	0.000	0.002	8790.3	48.2
capc_4n_4_9_27_60.1000	47605598.892787	*Opt*	1355.574	2359.426	355.2	0.000	0.000	8887.2	50.2

5 Zaključak

U ovom radu detaljno je opisan genetski algoritam za rešavanje lokacijskog problema snabdevača ograničenog kapaciteta u više nivoa. Rad takođe sadrži opis, matematičku formulaciju problema, kao i obrazloženje praktičnog značaja datog problema. Genetski algoritam je testiran na javno dostupnim instancama.

Lokacijski problemi su NP-teški problemi, pa njihovo rešavanje podrazumeva primenu neke heurističke metode, osim u slučaju instanci malih dimenzija. Primena genetskog algoritma se pokazala veoma uspešnom za rešavanje ove vrlo složene varijante lokacijskog problema.

Za rešavanje ovog problema binarna reprezentacija se pokazala najpogodnijom reprezentacijom niza potencijalnih lokacija snabdevača. Za fiksirane potencijalne lokacije snabdevača, zbog složenih uslova problema, dobijeni potproblem, koji je problem linearnog programiranja, se rešava pomoću CPLEX programskog paketa. Implementirani su osnovni genetski operatori koji vode računa o načinu kodiranja i prirodi problema čuvajući korektnost jedinki, čime se isključuje pojava nekorektnih jedinki u svakoj generaciji. Genetski operatori koji su se pokazali najboljim pri rešavanju ovog problema su: fino gradirana turnirska selekcija, jednopoziciono ukrštanje i prosta mutacija sa zaleđenim bitovima. Koristi se elitistička strategija koja omogućava direktno propuštanje elitnih jedinki u narednu generaciju, bez ponovnog računanja njihovih prilagođenosti. Kod implementacije genetskog algoritma keširanje u velikoj meri poboljšava performanse, ali ne utiče na ostale aspekte algoritma. Takođe, primenjene su i razne metode za sprečavanje preuranjene

konvergencije, gubljenja raznovrsnosti genetskog materijala i spore konvergencije genetskog algoritma.

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

- paralelizacija opisanog genetskog algoritma 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

Osim primene postojećih aspekata u ovom radu su dobijeni i neki novi naučni rezultati:

- Novi model mešovitog celobrojnog linearnog programiranja za MLCFLP.
- Korišćenje CPLEX programskog paketa u okviru genetskog algoritma za rešavanje potproblema sa fiksiranim snabdevačima.

Kao što se može videti iz eksperimentalnih rezultata, predloženi genetski algoritam je dostigao sva optimalna rešenja na testiranim primerima. Zbog svega gore navedenog, naučno istraživanje opisano u ovom radu daje neki doprinos oblastima kombinatorne optimizacije, lokacijskih problema i genetskih algoritama.

Literatura

[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).

[Ahu02] **Ahuja R.K., Orlin J.B., Pallottino S., Scaparra M.P., Scutellà M.G.**, "A multi-exchange heuristic for the single source capacitated facility location problem", Technical Report 02-11, Dip. di Informatica, Università di Pisa (to appear in *Management Science*) (2002).

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

[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

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

[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

[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).

[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

[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).

[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).

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

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

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

[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).

[Fab04] **Fabian A. Chudak, David P.Williamson**, “Improved approximation algorithms for capacitated facility location problems”, ETH Zurich, Institut für Operations Research (2004).

[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), pp. 143-161 (2003).

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

[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).

[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).

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

[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).

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

[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).

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

[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, pp. 390-402 (1999)

[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).

[Kra09] **Kratica J., Marić M., Tošić D., Filipović V.,** "A New Integer Linear Programming Model for the Multi Level Uncapacitated Facility Location Problem", submitted to Computers & Operations Research (2009).

[Krr83] **Krarup J.**, **Pruzan P.M.**, "The simple plant location problem: Survey and synthesis", *European Journal of Operational* Research, Vol. 12, pp. 36-81 (1983).

[Kum05] **Kumweang K., Kawtummachai R.**, "Solving a SSCFLP in a supply chain with ACO" (2005).

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

[Mar08b] **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)

[Mar09] **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).

[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., Francis R.L.**, "Discrete Location Theory", John Wiley & Sons (1990).

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

[M**ü**h97] **Mühlenbein H.**, "Genetic Algorithms", Local Search in Combinatorial Optimization, eds. Aarts E.H.L., Lenstra J.K., John Wiley & Sons Ltd., pp. 137-172 (1997).

[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).

[Ognj04] **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).

[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).

[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).

[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).

[Sca04] **Scaparra M.P., Pallottino S., Scutellà M.G.**, "Large scale local search heuristics for the Capacitated Vertex *p*-Center Problem", *Networks* (2004).

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

[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

[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).

[Tošo4] **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).

[Zha06] **Zhang J**., **Zhang X.**, **Wu L.,"**Capacitated facility location problem with general setup cost" , Computers & Operations Research, Volume 33, Issue 5, Page 1226-1241(2006).

Sadržaj