SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 958

RJEŠAVANJE PROBLEMA USMJERAVANJA VOZILA GENETSKIM ALGORITMOM

Ivana Stokić

Zagreb, lipanj 2009.

Zahvaljujem mentoru, doc. dr. sc. Domagoju Jakoboviću, na pruženim savjetima tijekom izrade ovog rada.

Sadržaj

1. Uvod

Optimizacijski problemi dijele se na kontinuirane i kombinatorne probleme. Prvi koriste realne varijable i cilj je otkriti vrijednosti tih varijabli za dobivanje najboljeg (optimalnog) rješenja, a drugi rade s diskretnim varijablama. Mnogi kombinatorni problemi su problemi koji se javljaju u stvarnom svijetu (engl. *real – life problems).* Većina takvih problema je NP teška, što znači da ih uobičajeni algoritmi koji bi se možda koristili za pretraživanje prostora stanja ne mogu riješiti u polinomijalnom vremenu. Poznati algoritmi za rješavanje takvih problema u najboljem su slučaju eksponencijalne složenosti. Jedan od takvih problema je i problem usmjeravanja vozila, tj. mnoge njegove inačice koje se razlikuju samo u ograničenjima koja koriste pri dobivanju (optimalnih) rješenja. U ovom će radu biti opisan način na koji se kapacitativni problem usmjeravanja vozila može riješiti genetskim algoritmom.

U sljedećem je poglavlju opisan jednostavan genetski algoritam (genetski operatori i parametri), nakon toga predstavljen je problem usmjeravanja vozila i opisane različite inačice tog problema. U četvrtom poglavlju nabrojani su i ukratko opisani različiti problemi koji imaju sličnosti s problemom usmjeravanja vozila. U petom poglavlju predstavljeni su neki od algoritama kojima se ovaj problem može riješiti (egzaktni, heuristički i metaheuristički algoritmi). U šestom poglavlju opisan je način na koji je riješen problem kapacitativnog usmjeravanja vozila (opisan je model problema i implementirani genetski operatori križanja i mutacije). U sedmom poglavlju prikazano je grafičko sučelje programskog rješenja problema, a u osmom poglavlju nalaze se tablice i grafovi dobiveni mjerenjem te rezultati i zaključci na temelju rezultata.

2. Genetski algoritam

Genetski algoritam, kojeg je 1975. god. prvi put, u knjizi "*Adaption in Natural and Artificial Systems*", predložio John Holland, pripada skupini evolucijskih algoritama. To je stohastička metoda optimiranja nastala po uzoru na prirodni proces evolucije. Evolucija traži najbolje i najprilagođenije jedinke na okolinu i uvjete u prirodi. Tako geni sposobnijih jedinki preživljavaju, dok geni slabijih ne. U prirodi dolazi i do razmjene genetskog materijala u procesu reprodukcije.

Na slici 2.1 prikazan je pseudokod jednostavnog genetskog algoritma [8]. Na početku se stvara početna populacija, zatim se ta populacija evaluira (računa se funkcija dobrote svake jedinke u toj populaciji). Zatim iterativno slijedi niz operacija sve dok ne bude zadovoljen jedan od kriterija zaustavljanja (a to može biti prođen određen broj generacija, dosegnuta vrijednost funkcije dobrote, broj iteracija bez poboljšanja ili vremensko ograničenje). Tu su uključeni operatori selekcije, križanja i mutacije. Jednostavni genetski algoritam koristi binarni prikaz rješenja, jednostavnu selekciju, križanje s jednom točkom prekida i jednostavnu mutaciju.

```
t ← 0;
InicijalizacijaPopulacije[ P(t) ];
EvaluacijaPopulacije[ (t) ];

while not uvjet zadovoljen do
P'(t) ← StvoriNovuPopulaciju[ P(t) ];
EvaluirajPopulaciju[ P'(t) ];
P(t+1) ← PrimijeniGenetske Operatore[ P'(t) ∪ Q ];
t ← t+1;
end while;
```

Slika 2.1 Pseudokod jednostavnog genetskog algoritma

2.1 Predstavljanje rješenja

Svaka bi jedinka morala sadržavati informacije o problemu koji se obrađuje. Za dani je problem jako važno odabrati prikladan prikaz rješenja, jer o tome ovisi učinkovitost genetskog algoritma. Jedna jedinka može biti prikazana binarno (ali za ograničen broj problema), zatim kao broj, niz brojeva, matrica, stablo i sl.

2.2 Početna populacija

Genetski algoritam počinje sa skupinom rješenja predstavljenim kromosomima (jedinkama), tj. početnom populacijom. Svaka jedinka predstavlja jedno potencijalno rješenje zadanog problema. To može biti matematička funkcija, plan rada neke tvornice, put trgovačkog putnika ili više vozila...

2.3 Funkcija dobrote

Svakoj se jedinki pridjeljuje vrijednost funkcije dobrote. Ta je funkcija mjera prilagođenosti pojedinog rješenja. Funkcija dobrote vodi prema najboljem dopuštenom rješenju i potiskuje zabranjena rješenja. Kvaliteta jedinki mjeri se pomoću funkcije cilja *f*. Za neke od postupaka selekcije ona ne smije biti negativna pa se funkcija dobrote *d* najčešće dobija translacijom funkcije cilja, npr. u svakoj se iteraciji od vrijednosti funkcije cilja pojedine jedinke oduzima najmanja vrijednost funkcije cilje u cijeloj populaciji.

2.4 Selekcija

Cilj selekcije je čuvanje i prenošenje dobrog genetskog materijala u sljedeću generaciju. Njome se odabiru jedinke koje će sudjelovati u križanju. Lošije se jedinke moraju odbaciti, dok je dobre poželjno prenijeti u sljedeću generaciju i upravo se time bavi selekcija. Postoje dva bitna načina podjele selekcije. Prvi je način prenošenja dobrih svojstava jedinki u sljedeću generaciju i po tom se kriteriju selekcije dijele na

generacijske i eliminacijske [1]. Generacijskom se selekcijom odabiru bolje jedinke koje će dalje sudjelovati u reprodukciji. Obično se odabere broj jedinki koji je manji od veličine populacije pa se poslije dodaju duplikati jedinki (neke se jedinke mogu odabrati više puta), ali to ne doprinosi poboljšanju kvalitete rješenja, već samo usporava evolucijski proces. Svaka jedinka preživljava samo jednu generaciju. Primjenjuje se i elitizam kojim najbolje jedinke (jedna ili više njih) ostaju sačuvane i nepromijenjene se prenose u sljedeću generaciju.

Kod eliminacijskih selekcija biraju se loše jedinke, a bolje jedinke preživljavaju postupak selekcije. U svakoj se iteraciji briše N loših jedinki. Parametar N zove se generacijski jaz ili mortalitet. Reprodukcijom preživjelih jedinki stvaraju se nove koje nadomještaju izbrisane jedinke. Kod ovih se selekcija ne stvaraju duplikati jedinki. Prema metodi odabira boljih jedinki kod generacijskih, tj. lošijih kod eliminacijskih, selekcije se dijele na proporcionalne i rangirajuće. Kod svih je selekcija zajednička velika vjerojatnost preživljavanja bolje od bilo koje lošije jedinke.

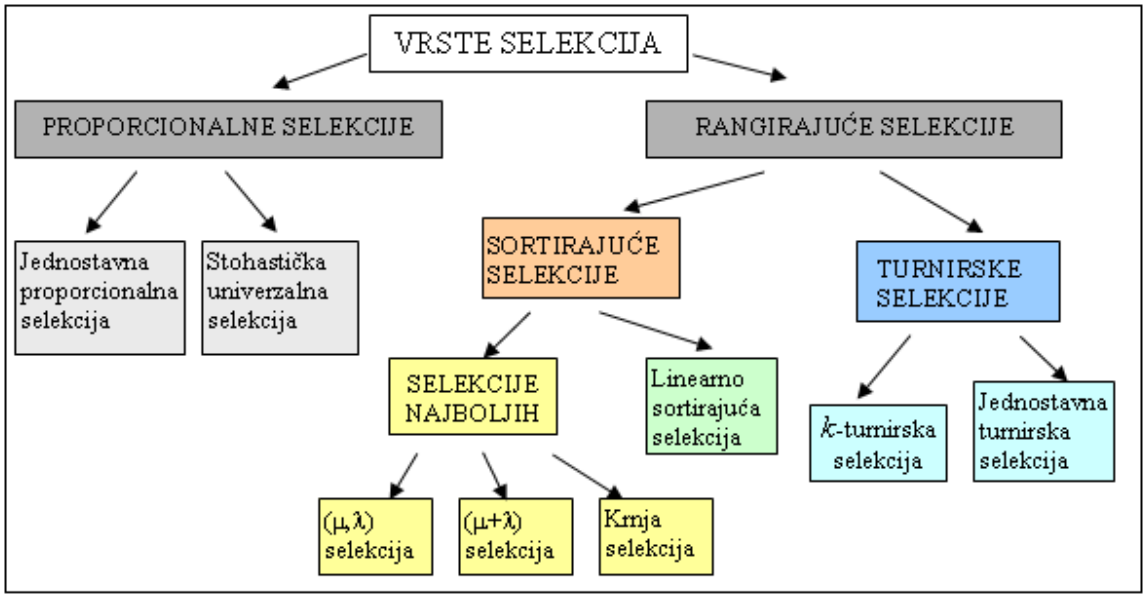
Proporcionalne selekcije ne čuvaju nužno najbolje jedinke i potrebno je u algoritam ugraditi elitizam koji će u sljedeću generaciju uvijek prenijeti nekoliko najboljih nepromijenjenih jedinki iz prethodne generacije. Prilikom sortiranja jedinki troši se procesorsko vrijeme i to može usporiti rad algoritma.

Kod univerzalne stohastičke selekcije na isti se način računa vjerojatnost selekcije, samo što se selekcija ne odvija u N, već u samo jednom koraku. Generira se N ekvidistantnih znački tako da se interval [0,1] podijeli na N+1 jednakih odsječaka. Broj znački koje se nalaze unutar odsječka jedne jedinke određuje koliko će se njenih kopija prenijeti u sljedeću populaciju.

Generacijske proporcionalne selekcije predmet su mnogih teorijskih razmatranja. Vjerojatnost selekcije kod takvih selekcija ovisi o kvocijentu dobrote pojedine jedinke i prosječne dobrote svih jedinki. Kod eliminacijskih proporcionalnih selekcija vjerojatnost odabira jedinke koja će se eliminirati obrnuto je proporcionalna njenoj dobroti. Umjesto funkcije dobrote mijenja se funkcija kazne kao razlika između

vrijednosti dobrote najbolje jedinke u populaciji i jedinke za koju se računa vrijednost funkcije kazne.

Kod rangirajućih selekcija vjerojatnost selekcije ne ovisi izravno o dobroti jedinke, već o položaju jedinke u sortiranom poretku svih jedinki. Sortirati se može ili po padajućim ili po rastućim vrijednostima funkcije dobrote. Za njih nije bitna vrijednost funkcije dobrote jedinke, već njen odnos prema dobrotama ostalih jedinki. Toj skupini pripadaju sortirajuće i turnirske selekcije. Sortirajuće selekcije obavljaju sortiranje jedinki prema njihovoj vrijednosti funkcije cilja. Za turnirske selekcije važan je samo odnos između nekoliko slučajno odabranih jedinki i nije potrebno obavljati sortiranje cijele populacije. Na slici 2.2 prikazane su različite vrste selekcija [1].



Slika 2.2 Podjela selekcija

2.5 Genetski operatori

Genetski operatori su križanje (reprodukcija) i mutacija.

2.5.1 Križanje

Križanje je binarni genetski operator kojim se razmjenjuje genetski materijal. U tom procesu sudjeluju dvije jedinke (roditelji) i nastaju jedna ili dvije nove jedinke (djeca) koje nasljeđuju svojstva svojih roditelja. Ako su roditelji imali dobra svojstva, vrlo je vjerojatno da će takva svojstva imati i djeca, ako ne i bolja. Ako dijete kojim slučajem bude lošije od svojih roditelja, to ne predstavlja problem, jer će ono ionako biti odbačeno tijekom procesa selekcije. Postoje različiti operatori križanja.

Najjednostavnije je križanje s jednom točkom prekida. Na nekom dijelu obaju roditelja odredi se prekid i onda prvi dio prvog i drugi dio drugog roditelja čine prvo dijete, a drugi dio prvog i prvi dio drugog roditelja drugo dijete. Takav način križanja prikazan je na slici 2.3 [9].

Roditelj1	11011 | 00100110110
Roditelj2	11011 | 11000011110
Dijete1	11011 | 11000011110
Dijete2	11011 | 00100110110

Slika 2.3 Primjer križanja s jednom točkom prekida

Postoje križanja s više točaka prekida. Za svaki se konkretan problem mogu odrediti neki posebni operatori križanja i to može pridonijeti učinkovitosti genetskog algoritma. Ako je kao parametar algoritma definirana vjerojanost križanja, on određuje koliko će se jedinki odabranih selekcijom križati, a koliko će ih ostati nepromijenjeno.

2.5.2 Mutacija

Mutacija je unarni operator, tj. obavlja se samo na jednoj jedinci. To je slučajna promjena genetskog materijala pod djelovanjem vanjskih utjecaja. Njen je cilj unošenje raznolikosti u genetski materijal jedinki i obnavljanje izgubljenog genetskog

materijala i tako se sprječava padanje svih jedinki u populaciji u lokalni optimum. Bez mutacije, samo sa selekcijom i križanjem, genetski bi algoritam bio sličan algoritmima lokalne pretrage, jer se križanjem, u potrazi za najboljim rješenjem, samo pretražuje prostor u okolini roditelja. Također postoji više različitih operatora mutacije. Jedan od parametara genetskog algoritma je i vjerojatnost mutacije. Izabire se slučajan broj i uspoređuje se s unaprijed zadanom vjerojatnošću mutacije te se tako odabiru jedinke koje će se mutirati.

Primjer jednostavne mutacije za jedinke predstavljene u obliku niza binarnih brojeva prikazan je na slici 2.4 [9]. Slučajno odabrani bitovi se invertiraju.

Dijete1	1101111000011110
Dijete2	1101100100110110
Mutirano dijete1	1100111000011110
Mutirano dijete2	1101101100110110

Slika 2.4 Primjer jednostavne mutacije

2.6 Parametri genetskog algoritma

Parametri su: veličina populacije, broj generacija (iteracija) i vjerojatnost mutacije. Ako se radi o generacijskom genetskom algoritmu potrebno je navesti i vjerojatnost križanja, a ako se radi o eliminacijskom genetskom algoritmu dodaje se broj jedinki za eliminaciju. Za različite vrijednosti parametara, algoritam brže ili sporije dolazi do boljeg ili lošijeg rješenja [1].

3. Problem usmjeravanja vozila

1959. godine Dantzig i Ramser predstavili su, u časopisu *Management Science,* problem usmjeravanja vozila (engl. *vehicle routing problem, VRP*).

Taj se problem vrlo često sreće u transportu i distribuciji robe i ima vrlo veliku praktičnu, ali i znanstvenu važnost. Zbog svoje je važnosti predmet intenzivnih istraživanja. Neki od primjera iz stvarnog svijeta su prikupljanje krupnog otpada, zatim čišćenje ulica, prijevoz djece u škole školskim autobusima, prijevoz osoba s invaliditetom, transport robe, transport pisama, doprema robe iz skladišta i sl. Radi se o raspodjeli dobara u određenom vremenu, određenom broju korisnika određenim brojem vozila koji su smješteni u jednoj ili više centrala (engl. *depot).* Vozila se kreću po zadanoj mreži prometnica. Rješavanje takvog problema obično se sastoji u pronalaženju ruta, gdje po svakoj ruti vozi jedno vozilo koje kreće iz centrale i vraća se u nju, sva zadana ograničenja moraju biti zadovoljena i ukupna cijena puta (tj. udaljenost koju će vozila prijeći) mora biti najmanja moguća.

Taj optimizacijski kombinatorni problem pripada kategoriji NP – teških problema; vrlo je teško pronaći optimalno rješenje. Zato se za njegovo rješavanje najčešće koriste metaheuristički i heuristički algoritmi. To su najčešće tzv. *population – based* ili *nature – inspired* algoritmi.

3.1 Opis problema

Mreža prometnica po kojoj se vozila kreću predstavljena je grafom. Bridovi grafa predstavljaju prometnice, a vrhovi korisnici koji moraju biti posluženi; obično samo jedan vrh predstavlja centralu u kojoj se nalaze sva vozila. Graf može biti usmjeren (engl. *directed)* ili neusmjeren (engl*. undirected*), a to ovisi o tome u kojem se smjeru vozila smiju kretati (jednosmjerne ili dvosmjerne ulice). Grafovi su uvijek težinski, jer se svakom bridu pridjeljuje cijena koja može predstavljati duljinu tog brida (ceste) ili vrijeme vožnje, tj. vrijeme u kojem vozilo prijeđe tu duljinu puta.

Svaki korisnik (engl. *customer*) ima zahtjev za isporuku robe (engl. *demand*) kolika joj se količina dobara mora dostaviti ili kod nje pokupiti. Osim toga, može postojati i vremensko ograničenje (engl. *time window*), što može biti npr. radno vrijeme neke trgovine kojoj se dostavlja roba, zatim vrijeme koje je potrebno da se roba ili teret ukrcaju, odnosno iskrcaju iz vozila (engl. *unloading or loading times*), što ovisi o tipu vozila. Može se još pojaviti i ograničenje na skup vozila koja mogu sudjelovati u prijevozu. Ponekad nije moguće zadovoljiti sva ograničenja korisnika pa se ta ograničenja smanjuju ili se neki korisnici ostavljaju neposluženi. Tada se korisnicima mogu pridijeliti određene kazne ili različite prednosti pri posluživanju. Svaka centrala (ako ih ima više) određena je brojem vozila koje je u njoj smješteno i određenom količinom tereta s kojom može raspolagati.

Robu ili teret prevoze vozila čija veličina i kapacitet mogu biti unaprijed određeni ili mogu biti definirani prema zahtjevima korisnika. Sva vozila kreću iz centrale, ali se, ako centrala ima više, ne moraju vratiti u istu centralu iz koje su krenuli. Svako vozilo ima i kapacitet koji predstavlja najveću masu ili volumen tereta kojeg prevozi. Ona mogu biti podijeljena i u dijelove ovisno o tome koji dijelovi vozila prevoze koju robu i sl. Vozači koji voze vozila također mogu imati ograničenja koja moraju zadovoljiti, npr. periodi dana u kojima se radi, broj i trajanje pauza tijekom radnog vremena, maksimalno vrijeme vožnje, prekovremeno... I rute mogu imati postavljena neka ograničenja, npr. vozeći određenom rutom trenutna količina tereta koji se prevozi ne smije prijeći najveći mogući kapacitet vozila, zatim korisnici koji su posjećene u toj ruti mogu zahtijevati ili samo ukrcaj ili samo iskrcaj robe ili oboje u isto vrijeme. Korisnici mogu biti posjećeni samo tijekom radnog vremena korisnika i vozača koji vozi neko vozilo koje dostavlja robu ili teret tom korisniku.

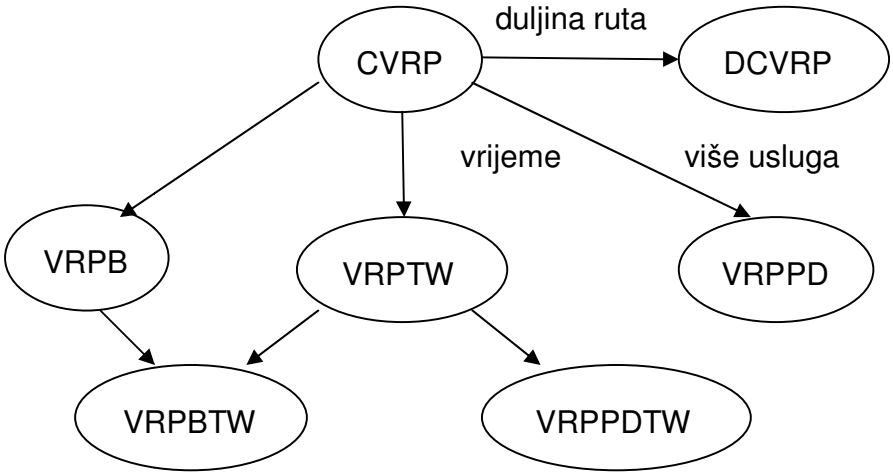
Mogu postojati i ograničenja u prednosti obilaska korisnika u rutama (engl. *precedence constraints*). To je slučaj kod problema kod kojih se roba mora od nekog korisnika pokupiti i drugom dostaviti (engl. *pickup and delivery*).

Potrebno je izračunati ukupnu duljinu puta ili vrijeme obilaska između svaka dva korisnika u grafu i između centrale i korisnika. Tada se, radi jednostavnosti, graf koji

predstavlja mrežu prometnica pretvara u potpuni graf (engl. *complete graph*). Ako je dan graf G(V, E), gdje je V skup vrhova u grafu, a E skup bridova grafa, tada su određeni bridovi e(i, j) između svaka dva vrha grafa, a c_{ij} je cijena između vrha *i* i vrha *j.* Sve se cijene u grafu mogu izraziti uz pomoć matrice cijene. Ona može biti simetrična ($c_{ij} = c_{ji}$) ili asimetrična ($c_{ij} \neq c_{ji}$), ovisno o tome je li graf usmjeren ili nije.

3.2 Inačice problema

Postoji mnogo različitih vrsta problema usmjeravanja vozila, ovisno o ograničenjima koja su uzeta u obzir. Na slici 3.1 prikazane su pojedine vrste problema [2], a u nastavku je opisana svaka pojedina vrsta.



Slika 3.1 Vrste problema usmjeravanja vozila

3.2.1 Kapacitativni VRP

Kapacitativni problem usmjeravanja vozila (engl. *capacitated vehicle routing problem,* u daljnjem tekstu CVRP) utemeljen je na ograničenju kapaciteta. Kod te su vrste problema zahtjevi korisnika i količina robe koja im se dostavlja deterministički, tj. već unaprijed poznati i sva vozila su jednaka i kreću iz jedne centrale i vraćaju se u nju.

Cilj je minimizirati ukupnu cijenu (to je funkcija broja ruta i njihove duljine, odnosno vremena unutar kojeg se te rute obiđu).

CVRP je također opisan grafom G = (V, E). V je skup vrhova grafa (neka ih ima n+1), a E je skup bridova e_{ij} između vrhova *i* i *j.* Centrali može odgovarati vrh 0 ili vrh n+1. Svakom bridu grafa pridijeljena je nenegativna cijena c_{ij}. Vrijedi da je $c_{ii} = +\infty$, što znači da petlje u grafu nisu dopuštene. Ako je G usmjeren graf, tada je matrica cijene asimetrična pa se radi o asimetričnom kapacitativnom problemu (ACVRP), a ako graf nije usmjeren, matrica cijene je simetrična i radi se o simetričnom problemu (SCVRP). Ako je cijena brida između vrhova *i* i *j* jednaka najkraćoj udaljenosti kojom se može doći iz vrha *i* u vrh *j* i ako to vrijedi za svaki brid grafa, tada matrica cijene zadovoljava tzv. nejednakost trokuta prikazanu u jednadžbi (3.1).

$$c_{ik} + c_{kj} \geq c_{ij}, \forall\, i, j, k \in V \tag{3.1}$$

Ponekad se svakom vrhu grafa pridjeljuju koordinate, ovisno o tome gdje je vrh smješten pa se za cijenu brida uzima Euklidska udaljenost između dva vrha koja čine taj brid. Tada je matrica cijene simetrična i također zadovoljava nejednakost trokuta.

Svaki vrh ima i nepromjenjivu nenegativnu veličinu zahtjeva (engl. *demand*), d_i, a centala ima zahtjev $d_0 = 0$.

U centrali je smješteno K jednakih vozila, svaki može prevoziti najveću količinu tereta C (što odgovara njegovom kapacitetu). Pretpostavlja se da za i - ti vrh grafa ($\forall$ i $\in$ V) vrijedi $d_i \leq C$. Svako vozilo može obići najviše jednu rutu i pretpostavlja se da K nije manji od K_{min}, gdje je K_{min} najmanji broj vozila potreban za posluživanje svih korisnika.

Rješenje ovog problema sastoji se u pronalaženju točno K jednostavnih ruta s minimalnom cijenom (definiranu kao zbroj cijena svih bridova koji pripadaju toj ruti) tako da vrijedi:

1. Svaka ruta počinje i završava u centrali
2. Svaki je korisnik (vrh, čvor) posjećen najviše jednom

3. Zbroj vrijednosti svih zahtjeva za isporukom robe svih korisnika koji čine neku rutu ne smije prijeći kapacitet vozila koji obilazi tu rutu

Matematički izraženo, traži se niz ciklusa W = {W_1, ..., W_m}, gdje je W_i = {0, x_i (1), x_i (2), ..., x_i (n_j)} koji počinju i završavaju u centrali, a traži se takav niz ciklusa W^*, koji daje minimalnu vrijednost kriterija definiranu jednadžbom (3.2), ako je centrala označena indeksom vrha 0.

$$J = \sum_{i=1}^{n} \left(c_{0x_i(n_i)} + \sum_{j=1}^{n_j-1} c_{x_i(j)x_i(j+1)} + c_{x_i(n_j)0} \right) \tag{3.2}$$

Postoji još nekoliko različitih vrsta CVRP – a. Npr. može vrijediti K $\neq$ K_{min}, što znači da, pri transportu i posluživanju korisnika, ne moraju biti iskorištena sva vozila. Može vrijediti i da vozila imaju različite kapacitete C_k, k = 1, ..., K. Također, rute koje sadrže samo jednog korisnika mogu u nekim inačicama CVRP – a biti zabranjene. Jedna od najčešće obrađivanih inačica je DVRP (engl. *Distance – Constrained VRP*), gdje, umjesto kapacitativnih, postoje ograničenja u udaljenosti ili vremenu. Umjesto kapaciteta C postavlja se najveća moguća duljina rute T i tada se mora paziti da ukupna duljina svih bridova (a svakom je bridu pridijeljena određena nenegativna duljina t_{ij}) koji pripadaju nekoj ruti ne bude veća od vrijednosti T. Ako postoje ograničenja u kapacitetu, ali i ograničenja u udaljenosti ili vremenu, tada se govori o još jednoj inačici CVRP –a ; to je tzv. DCVRP [2].

3.2.2 VRP s vremenskim prozorima

Kod ove inačice problema, osim kapacitativnog ograničenja, svakom je korisniku pridijeljen i vremenski interval [a_i, b_i] koji se naziva vremenski prozor (engl. *time window*). Vremenski je prozor definiran najranijim trenutkom dolaska i najkasnijim trenutkom odlaska vozila. Ako vozilo do korisnika dođe kasnije od najkasnije definiranog dopuštenog trenutka, ono će se kazniti, a ako dođe ranije od najranije dopuštenog definiranog trenutka, ono će čekati. Vozila, također, moraju proći svoju rutu unutar predefiniranog vremena putovanja $t_{ij.}$ Ako se radi o mekanim vremenskim prozorima (engl. *soft time windows*), tada se ne smatra da je rješenje, kod kojega je

do korisnika neko vozilo došlo nakon gornje vremenske granice određene vremenskim prozorom, neispravno, već se kod računanja funkcije dobrote u obzir uzima i dodatna kazna [8].

3.2.3 VRP s više centrala

Ako su korisnici grupirani oko centrala, tada se može govoriti o nekoliko različitih nezavisnih VRP – a [8]. Ako su međusobno pomiješani, tada se radi o problemu usmjeravanja vozila s više centrala (engl. *Multuple depots VRP*). Svakoj je centrali pridijeljen određen skup vozila koji iz njega kreću. Cilj je minimizirati broj vozila i ukupna vremena putovanja. Rješenje je ispravno ako zadovoljava ograničenja kao i za klasičan VRP, uz to što se svako vozilo mora vratiti u centralu iz koje je krenulo [2].

3.2.4 Periodični VRP

Klasičan VRP podrazumijeva posluživanje svih korisnika u jednom danu. Periodični VRP (engl. *Periodic VRP, PVRP*) proširuje posjećivanje korisnika na M dana, pri čemu se vozilo ne mora vratiti u centralu u istom danu kad je i krenulo, već unutar M dana (svaki se korisnik mora posjetiti bar jednom unutar tih M dana).

3.2.5 VRP s podijeljenom isporukom

Kod problema usmjeravanja vozila s podijeljenom isporukom (engl. *VRP with Pickup and Delivering, VRPPD*) svakom je korisniku pridijeljena vrijednost zahtjeva za isporukom robe d_i, a i vrijednost za količinom robe koja se mora pokupiti p_i. Pretpostavlja se da se isporuka vrši prije preuzimanja. Trenutno opterećenje vozila definira se kao razlika kapaciteta vozila i robe koja je već isporučena te ta vrijednost zbojena s količinom robe koju je vozilo pokupilo (preuzelo). Sva ograničenja koja vrijede za CVRP vrijede i za VRPPD, a dodatno je još uključeno i ograničenje u prednosti, tj. u nekoj se ruti korisnici (tj. vrhovi grafa) koji zahtijevaju samo isporuku robe moraju poslužiti prije korisnika koji zahtijevaju njeno preuzimanje. Ujedno, vozilo

mora imati dovoljan kapacitet za robu koju mora isporučiti i za onu koju mora preuzeti [2].

3.2.6 VRP s povratom

Kod ove vrste usmjeravanja vozila (engl. *VRP with Backhauls, VRPB*) korisnici, tj. vrhovi V \ {0} (svi osim centrale), podijeljeni su u dva skupa. Prvi sadrži n Linehaul korisnika (engl. *Linehaul customers*), a drugi m Backhaul korisnika (engl. *Backhaul customers*).

Korisnicima iz prvog skupa L samo se dostavlja, a od onih iz drugog skupa B samo se dobavlja. Vrhovi grafa se imenuju, tako da vrijedi L = {1, ..., n}, a M = {m+1, ..., n+m}. Ako neka ruta prolazi kroz korisnike iz oba skupa, tada korisnici iz prvog skupa imaju uvijek prednost pred korisnicima iz drugog skupa. Korisnici, također, imaju unaprijed zadane zahtjeve za količinom robe koja će im biti isporučena ili od njih preuzeta. Ako je matrica cijene asimetrična, tada se radi o problemu AVRPB.

Rute koje bi sadržavale samo korisnike iz skupa B, u pravilu, nisu dopuštene. Ovaj je problem generalizacija SCVRP – a i ACVRP – a u slučaju kad je B = { $\emptyset$ }. Postoji i inačica problema s vremenskim prozorima (engl. *VRPBTW*)

3.2.7 Stohastički VRP

Kod stohastičkog problema usmjeravanja vozila (engl. *Stochastic VRP, SVRP*) neki od elemenata je slučajan. On se dijeli na tri podvrste, ovisno o tome što je slučajno. Korisnici mogu biti prisutni s vjerojatnošću p_i, a ne prisutne s vjerojatnošću 1- p_i. Zatim slučajno izabrani mogu biti i zahtjevi korisnika za isporukom robe. Također, slučajna mogu biti i vremena putovanja t_{ij} i vremena posluživanja. U tim se slučajevima ne moraju zadovoljiti sva ograničenja, a ako se neka i ne zadovolje, uvode se kazne.

3.2.8 VRP s dopunom

Kod problema usmjeravanja vozila s dopunom (engl. *VRP with Satellite Facilities, VRPSF*) omogućeno je dopunjavanje tereta u vozilu dok je ono na putu (dok obilazi neku rutu). To se često susreće kod distribucije goriva.

3.2.9 Raspodijeljeni VRP

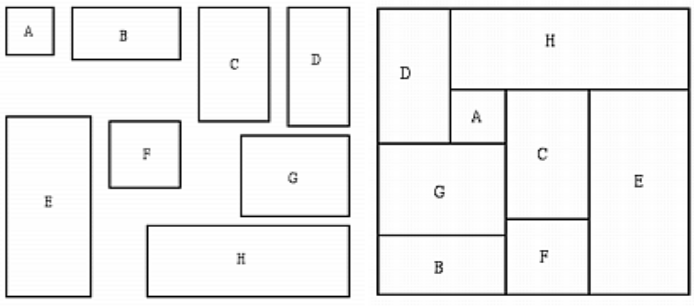
Raspodijeljeni VRP (engl. *Split Delivery VRP, SDVRP)* dopušta da više vozila posjeti jednog korisnika. Vrijede ograničenja kao i kod klasičnog problema.

4. Problemi koji su slični problemu usmjeravanja vozila

VRP predstavlja presjek dvaju NP – problema, a to su problem trgovačkog putnika (engl. *Traveling Salesman Problem, TSP*) i problem pakiranja (engl. *Bin Packing Problem, BPP*). Problem usmjeravanja vozila teže je riješiti od problema trgovačkog putnika, njega je moguće riješiti nizom različitih algoritama (uz relativno skromne resurse), dok su najsofisticiraniji algoritmi za točno rješavanje VRP – a u stanju riješiti problem tek s oko 100 korisnika [6]. Najveći VRP koji je riješen je "F-n135-k7" (135 korisnika, 7 vozila), a najmanji VRP koji još uvijek nije riješen je "B-n50-k8" (50 korisnika, 8 vozila). Postoji mnogo poznatih testnih primjera (npr. *Augerat et al. instances; Christofides, Mingozzi and Toth instances; Fisher instances; Taillard instances)* [8].

Problem se pretvara u problem trgovačkog putnika ako je K = 1 i C = ∞. Trgovački putnik mora pronaći, krećući iz svog sjedišta, najkraći put kojim bi obišao sve korisnike samo jednom i pritom se mora natrag vratiti u svoje sjedište.

Ako su sve cijene jednake nuli, tada se govori o problemu pakiranja (engl. *Bin Packing Problem, BPP*). Rješenje problema pakiranja sastoji se u pronalaženju najmanjeg broja kutija kapaciteta V u koje se moraju rasporediti predmeti određenog volumena. Mogu se zadati ograničenja u prednosti, a kutije se pakiraju tako da volumen, masa i ostalo ne prijeđu neku unaprijed zadanu vrijednost. Tipičan primjer početnog problema i rješenja prikazani su na slici 4.1 [8].

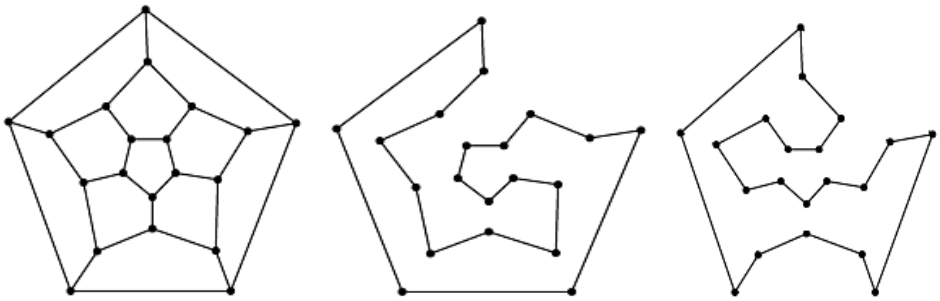


Slika 4.1 Primjer problema pakiranja (lijevo) i odgovarajućeg rješenja (desno)

Jedan od sličnih problema je i problem raspoređivanja poslova (engl. *Job Scheduling Problem*). Kod problema raspoređivanja poslova postoji konačan skup od n poslova, svaki se posao sastoji od nekoliko operacija. Uz to postoji i skup od m strojeva, pri čemu svaki stroj može izvršavati najviše jednu operaciju u jednom trenutku. Svaka operacija treba biti izvršena na nekom od strojeva unutar nekog neprekinutog perioda vremena. Cilj je pronaći raspored (raspodjelu operacija u vremenu na strojeve) najmanje duljine [7].

Zatim još postoji i *Generic Assignment Problem (GAP)* kod kojega se skup od m zadadataka (operacija) pridjeljuje skupu od n agenata. Definira se i cijena pridjeljivanja zadatka *i* agentu *j*, kapacitet svakog agenta i količina sredstava potrebnih agentu kako bi obavio zadatak. Ako je nekom agentu potrebno više različitih vrsta sredstava, radi se o *Multi – Resource GAP problemu*.

Jedan od sličnih problema je i pronalaženje Hamiltonovskih ciklusa (engl. *Hamiltonian Cycle Problem*). Pronalaženje Hamiltonovskog ciklusa u nekom grafu sastoji se od pronalaženja i uređenja vrhova u grafu, tako da je svaki vrh posjećen samo jednom (isto kao i TSP). Na slici 4.2 prikazani su graf (krajnje lijeva slika) i dva primjera Hamiltonovskih ciklusa u tom grafu (desno) [8].



Slika 4.2 Primjer grafa i Hamiltonovskih ciklusa u tom grafu

5. Algoritmi za rješavanje problema usmjeravanja vozila

Algoritmi uz čiju se pomoć ovaj problem može riješiti dijele se u dvije skupine. To su egzaktni i heuristički algoritmi. Heuristički se algoritmi dijele na klasične heurističke i metaheurističke algoritme.

5.1 Egzaktni algoritmi

Egzaktni algoritmi ne mogu pronaći optimalno rješenje u razumnom vremenu, pogotovo ako je broj korisnika koji moraju biti posluženi jako velik. Istražuju se sva moguća rješenja, sve dok ono najbolje nije postignuto. U nastavku su ukratko objašnjeni neki od najčešće korištenih.

5.1.1 Branch and bound algoritam

Branch and bound algoritmom dobiveno je rješenje za CVRP koji je imao 71 korisnika [8]. Taj se algoritam koristi strategijom *podijeli pa vladaj* i prostor problema dijeli na više manjih podskupova koje zatim rješava zasebno. U početku se prostor rješenja S dijeli i definiraju se donje i gornje granice. Neka se rješenja, za koja se uvidi da bi mogla biti neispravna, odbacuju. Ako se S podijeli na n potprostora, tada se ti potprostori nazivaju djecom i dodaju se u skup mogućih potproblema (tj. onih koji čekaju procesiranje). To je *branching.* Ako se pronađe ispravno rješenje koje je bolje od onog trenutnog, tada se trenutno rješenje zamjenjuje boljim. Ako se otkrije da potproblemi nemaju rješenja, tada se oni odbacuju. Inače, uspoređuje se donja granica potproblema s globalnom gornjom granicom i ako je ona jednaka ili veća, tada se on odbacuje. Ako se ne može odbaciti, tada stablo treba podrezati i nastalu djecu ponovo unijeti u skup potproblema. Proces završava kad je skup potproblema prazan.

5.1.2 Branch and cut algoritam

Branch and cut algoritam koristio se za rješavanje CVRP – a. Još uvijek je uporaba ovog algoritma u rješavanju danog problema ograničena.

5.2 Heuristički i metaheuristički algoritmi

Heuristički algoritmi obavljaju dosta ograničenu pretragu prostora rješenja, ali nalaze dobra i optimalna rješenja u razumnom vremenu. Oni se dijele na algoritme s konstruktivnom heuristikom i na algoritme za poboljšavanje rješenja. Neki od njih opisani su u nastavku.

5.2.1 Algoritmi s konstruktivnom heuristikom

Ovi algoritmi grade rješenje korak po korak.

5.2.1.1 Algoritam uštede

Algoritam uštede definirali su Clarke i Wright 1964. god. Taj se algoritam temelji na postepenom stvaranju rute. Algoritam započinje s n nezavisnih ruta, od kojih svaka započinje i završava u centrali 0.

Definiraju se uštede prema jednadžbi (5.1).

$$s_{ji} = c_{i0} + c_{0j} - c_{ij}, \forall i, j = 1, \dots, n, i \neq j \tag{5.1}$$

Uštede se sortiraju prema padajućem poretku. Nakon toga kreće se od najgornje vrijednosti uštede s_{ij} i gleda se postoji li ruta koja započinje s (0, j) i postoji li ruta koja završava s (i, 0). Ako takve dvije rute postoje, tada se one spajaju, tako što se briše (0, j) i (i, 0) i uvodi se nova ruta (i, j). Najbolja je paralelna inačica ovog algoritma kod koje se u svakoj iteraciji spajaju rute s najvećom vrijednošću uštede [6]. Postoje tri slučaja kad se rute mogu objediniti: ako nijedan od korisnika *i* i *j* nije u ruti s više korisnika, tada se stvara ruta s dvama korisnicima i centralom; ako je samo jedan od korisnika (*i* ili *j*) uključen u rutu i ako je, u toj ruti, susjedan centrali, tada se drugi

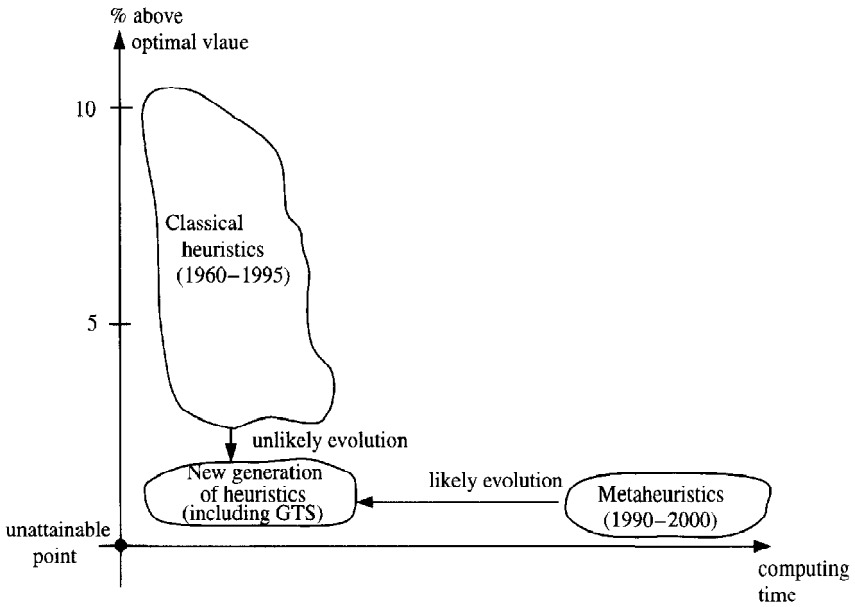
korisnik dodaje toj ruti; a ako su oba korisnika (i *i* i *j*) uključena u različite rute, a nijedan od njih nije susjedan centrali, tada se te dvije rute povezuju u jednu.

5.2.2 Algoritmi za poboljšavanje rješenja

Ovi algoritmi (engl. *improvement algorithms*) mogu raditi na samo jednoj ruti (engl. *Single – Route*) ili na više njih (engl. *Multi - Route*).

5.2.3 Metaheuristički algoritmi

Postoji nekoliko različitih metaheurističkih algoritama za rješavanje problema usmjeravanja vozila: simulirano kaljenje (engl. *Simulated Annealing*, determinističko kaljenje (engl. *Deterministic Annealing*), tabu pretraživanje (engl. *Tabu - Search*), genetski algoritmi (engl. *Genetic algorithms),* kolonije mrava (engl. *Ant Systems*) i neuronske mreže (engl. *Neural Networks*) [2]. Svi ti algoritmi mogu pronaći optimalno rješenje za problem od nekoliko stotina korisnika. U budućnosti će se više isplatiti ulagati u pronalaženje rješenja s metaheurističkim, nego s klasičnim heurističkim algoritmima. Na slici 5.1 prikazan je razvoj heuristike i metaheuristike za VRP [2].



Slika 5.1 Razvoj (meta)heuristike za VRP

6. Rješavanje problema kapacitativnog usmjeravanja vozila uz pomoć genetskog algoritma

U nastavku će biti opisan način na koji je ostvareno korištenje jednostavnog genetskog algoritma pri rješavanju problema kapacitativnog usmjeravanja vozila (u daljnjem tekstu CVRP), opis algoritma i ideje.

Problemski model koji se rješava za mrežu prometnica uzima potpun usmjeren graf (pretpostavlja se da su sve prometnice međusobno povezane i da nema jednosmjernih ulica), zato je matrica cijene simetrična. Centrala ima indeks 0. Svi korisnici, osim centrale, imaju unaprijed definirane različite zahtjeve za isporukom robe. Vozila su jednaka (imaju jednake kapacitete) i od korisnika samo otpremaju robu i teret. Udaljenost između korisnika (vrhova grafa) računa se uz pomoć Euklidske udaljenosti.

Ideja rješenja ovog problema implementirana je programskom jeziku C#, u programskom alatu *Microsoft Visual Studio 2008.*

6.1 Graf

Za definiranje mreže prometnica korišten je potpun usmjeren graf. Koordinate vrha mogu se zadati slučajno ili se mogu pročitati iz datoteke. Isto to vrijedi i za zahtjeve korisnika. Sve je ovo određeno u razredu *Graph.*

6.2 Predstavljanje rješenja

Jedinka je predstavljena nizom cijelih brojeva koji predstavljaju korisnike u mreži prometnica (tj. u grafu). Pojedinu rutu čini niz korisnika koje je obišlo jedno vozilo (podrazumijeva se da svaka ruta započinje centralom (vrh 0) i u njoj završava pa to nije prikazano u jedinci). Pojedine rute odvojene su oznakom vozila (npr. njegovim identifikacijskim brojem). Ti se brojevi kreću od ukupnog broja korisnika zbrojenog s

brojem vozila koji mogu posluživati korisnike. Npr. ako je izabrano 10 korisnika i 4 vozila, tada će rute u sebi sadržavati neke od brojeva 0 - 9, a odvajat će ih brojevi od 10 – 13:

4 – 5 – 2 – **11** - **10** – 3 – 1 – **13** – 7 – 8 – 9 – **12** – 6

Slika 6.1 Prikaz jedne jedinke

Ruta ima koliko i vozila. Na Slika 6.1 6.1 prikazana je jedna moguća jedinka (10 korisnika i 4 vozila, 4 rute odvojene, na slici podebljanim, ID brojevima vozila). Kao što se vidi sa slike, može se dogoditi da su neka vozila pridijeljena praznim rutama (vozilo 10 i 11).

Jedinke se mogu prikazati i na druge načine. Npr. jedinka se sastoji od niza kromosoma, a svaki kromosom predstavlja jednu rutu.

Jedinka ima svoj razred *Kromosom*, u kojem je definiran identifikacijski broj jedinke (ID jedinke), zatim vrijednost funkcije dobrote jedinke (*fitness*) i redObilaska (lista koja predstavlja korisnike i vozila koja su obišla te korisnike, kao na slici 6.1). Svaka jedinka ujedno pozna i izabrani graf (mrežu prometnica).

6.3 Funkcija dobrote

Funkcija dobrote uzima u obzir cijenu pojedinih bridova, ali i njihove zahtjeve za isporukom robe (kao i kapacitete vozila koji prolaze tim bridovima). Cijena pojedinog brida računa se uz pomoć obične Euklidske 2D udaljenosti između dva vrha prikazane jednadžbom (6.1).

$$d = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \tag{6.1}$$

Kako bi kod CVRP – a rješenje bilo ispravno, ono mora zadovoljiti ograničenja kapaciteta, tj. zbroj količine zahtjeva za isporukom robe svih korisnika u nekoj ruti ne smije biti veći od kapaciteta vozila koje opslužuje tu rutu. Sva vozila imaju jednak

kapacitet. U metodi *IzracunajFitness()* ispituje se ispravnost pojedine jedinke i prema tome se računa vrijednost funkcije dobrote (engl. *fitness)*. Računa se cijena i provjerava je li zadovoljeno ograničenje za svaku pojedinu rutu. U listu *PrekoracenjeKapaciteta* dodaje se ukupna količina zahtjeva korisnika (od koje se oduzima kapacitet vozila) i još se sve dijeli s kapacitetom vozila; tako se dobiva prekoračenje kapaciteta za svaku pojedinu rutu (ako prekoračenja nema, tada će se smatrati da prekoračenje ima vrijednost nula i konačno će se uzimati samo zbrojene cijene bridova koje pripadaju ruti). Računanje vrijednosti funkcije dobrote prikazano je jednadžbom (6.2). *FitnessZaRutu* ukupni je *fitness* za svaku pojedinu rutu.

$$\text{fitness} = 1 / (\text{fitnessZaRutu} * (1 + 2 * \text{PrekoracenjeKapaciteta})) \quad (6.2)$$

Najbolja jedinka je, prema tome, ona koja ima najveću dobrotu.

6.4 Selekcija

Implementirana je *Roulette – Wheel selekcija* u razredu *GA* u metodi *RouletteWheelSelekcija* (ova metoda kao parametar prima populaciju, a vraća listu odabranih jedinki). Zbrajaju se dobrote svih jedinki trenutne populacije i određuje se slučajan broj (*cilj*) između nule i prethodno izračunate ukupne dobrote. Zatim se obilazi po svim jedinkama i opet se zbrajaju dobrote jedinki (svih do trenutne jedinke) i ako se ustanovi da je suma prešla cilj, tada se trenutna jedinka sprema u listu odabranih jedinki. Postupak se ponavlja dok broj jedinki u listi odabranih jedinki nije jednak veličini populacije.

6.5 Operatori križanja

U razredu *GA* implementirana su 4 različita operatora križanja. Kao parametar genetskog algoritma definirana je vjerojatnost križanja koja određuje u kojoj će se mjeri jedinke križati. Npr. ako je vjerojatnost križanja 60%, tada će se prvih 60% jedinki iz liste odabranih za križanje križati (na način prva s drugom, treća s četvrtom itd., a ako je postotak jedinki u listi za križanje neparan, tada će se zadnja jedinka

križati s najboljom iz liste za križanje); dok će ostalih 40% ostati nepromijenjeno. U nastavku su opisani implementirani operatori križanja.

6.5.1 Partially Mapped Crossover (PMX)

Ovaj operator križanja predstavili su Goldberg i Lingle [10]. Gradi dijete tako da uzme dio reda obilaska jednog roditelja i pokušava u što većoj mjeri sačuvati poziciju i redoslijed korisnika iz drugog roditelja. Križanje je implermentirano metodom *PMX* koja za parametar uzima populaciju, odabrane jedinke (jedinke koje bi mogle sudjelovati u križanju) i vjerojatnost križanja, a vraća listu novih jedinki nastalih križanjem.

Iz liste reda obilaska miču se vozila (uz to da se, uz pomoć asocijativnog polja – *Dictionary*, pamti mjesto gdje su se vozila nalazila, kako bi se poslije križanja mogla vratiti na ista mjesta) i križanje se vrši samo nad redom obilaska.

Primjer: U listi za križanje nalaze se dvije jedinke : 1 – 2 – **10** – 3 – 4 – 5 – **12** – 6 – **11** – 7 – 8 – 9 – **13** i jedinka 4 – **11** – 5 – 3- 1 – 8 – **12** – **13** – 7 – 6 - 9 – 3 – **10**. Iz reda obilaska maknut će se vozila i odredit će se dva slučajna broja (koja podijele red obilaska na dva dijela; podjela predstavljena separatorom " | ").

Roditelj 1: 1 – 2 – 3 | 4 – 5 – 6 - 7 | 8 – 9

Roditelj 2: 4 – 5 - 3 | 1 – 8 – 7 - 6 | 9 - 3

Dijete1: x – x – x - | 1 – 8 – 7 – 6 | x – x

Dijete2: x – x – x | 4 – 5 – 6 – 7 | x – x

Stvaraju se tzv. *mapping relacije* na temelju oba roditelja, za ovaj primjer to su 1 → 4, 4 → 1; 5 → 8, 8 → 5; 6 → 7 i 7 → 6. Na temelju tih relacija popunit će se u djeci dijelovi označeni s "x". Tako slijedi:

Dijete1: 4 – 2 – 3 – 1 – 8 – 7 – 6 – 5 – 9

Dijete2: 1 – 8 – 3 – 4 – 5 – 6 – 7 – 5 – 9

Kad se djeci dodaju vozila, slijede dvije nove jedinke: 4 – 2 – **10** – 3 – 1 – 8 – **12** – 7 – **11** – 6 – 5 – 9 – **13** i 1 – **11** – 8 – 3 – 4 – 5 – **12** – **13** – 6 – 7 – 5 – 9 - **10**.

6.5.2 Cycle Crossover (CX)

Predstavio ga je Oliver, on gradi dijete uzimajući dijelove obilaska iz oba roditelja. Križanje je implementirano metodom *CX* koja prima sve parametre kao i prije spomenuta metoda *PMX* i također vraća listu novih jedinki (i dio nepromijenjenih, ovisno o vrijednosti vjerojatnosti križanja).

Na početku se, također, iz reda obilaska izbacuju vozila, ali se pamti mjesto gdje su se vozila nalazila.

Primjer: U listi za križanje nalaze se dvije jedinke : 1 – 2 – **10** – 3 – 4 – 5 – **12** – 6 – **11** – 7 – 8 – 9 – **13** i jedinka 4 – **11** –1– 2- 8 – 7 – **12** – **13** – 6 – 9 - 3 – 5 – **10**.

Nakon što su iz reda obilaska maknuta sva vozila, na roditeljima se može primijeniti križanje.

Roditelj 1: 1 – 2 – 3 - 4 – 5 – 6 – 7 - 8 – 9

Roditelj 2: 4 - 1 - 2 - 8 - 7 - 6 – 9 - 3 - 5

Definira se “ciklička” relacija prema roditeljima: 1(r1) → 4(r2); 4(r1) → 8(r2); 8(r1) → 3(r2); 3(r1) → 2(r2); 2(r1) → 1(r2). Ovdje se završava, jer se od broja 1 i krenulo.

Dijete1: 1 – 2 – 3 – 4 – x – x – x – 8 – x

Dijete2: 4 – 1 – 2 – 8 – x – x – x – 3 - x

Dijelovi označeni s “x” nadopunjavaju se prema drugom roditelju:

Dijete1: 1 – 2 – 3 – 4 – 7 – 6 - 9 – 8 – 5

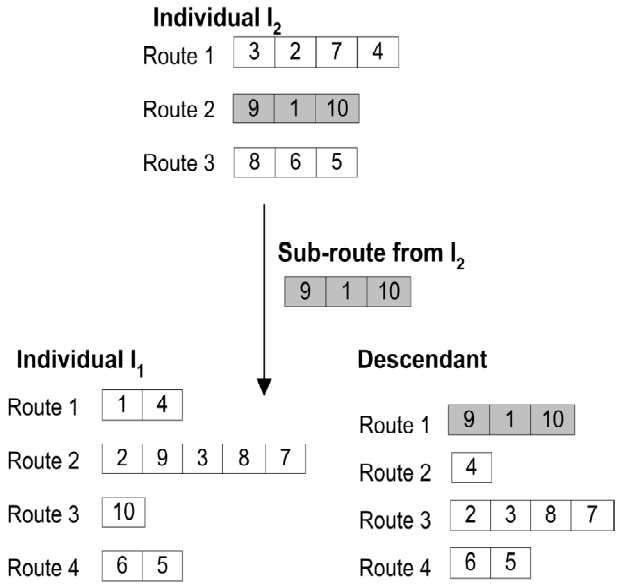
Analogno i za drugo dijete.

Dijete2: 4 – 1 – 2 – 8 – 5 – 6 – 7 – 3 - 9

Kad se djeci dodaju vozila, slijede dvije nove jedinke: 1 – 2 – **10** – 3 – 4 – 7 – **12** – 6 – **11** – 9 – 8 – 5 – **13** i 4 – **11** – 1 – 2 – 8 – 5 – **12** – **13** – 6 – 7 – 3 – 9 - **10**.

6.5.3 Generic Crossover (GVR)

Ovaj operator križanja uzima podrutu ili rutu iz prvog roditelja i stavlja je na početak djeteta, a ostatak djeteta puni se rutama drugog roditelja.



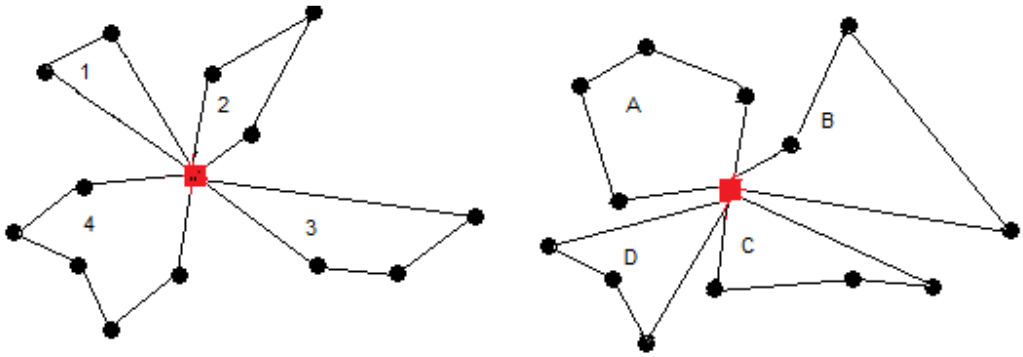
Slika 6.2 Primjer GVR – a

Na slici 6.2 prikazana su dva roditelja, svaki od njih ima tri rute. Uzima se druga ruta prvog roditelja i stavlja se na početak djeteta, ostatak djeteta popunjen je korisnicima drugog roditelja, ali tako da novonastala jedinka bude ispravna (ne smiju se ponavljati indeksi korisnika) pa su rute drugog roditelja koje se dodaju u dijete donekle izmijenjene.

Križanje je implementirano u metodi *GVR* koja ima iste parametre kao i metode za prethodna križanja. Također, vraća listu djece.

6.5.4 Uniformno križanje

Uniformno se križanje (engl. *Uniform Crossover, UC*) vrlo često spominje u literaturi [12]. Kod problema usmjeravanja vozila uniformno križanje bira između ruta roditelja (ovisno o njihovoj kvaliteti) i sve dok jedne s drugima nisu u konfliktu. Na slici 6.3 prikazan je jednostavan primjer ovog križanja.



Slika 6.3 Prikaz dviju jedinki

Kod ovog križanja računa se kvocijent $R_{s,r}$ prema jednadžbi

$$R_{s,r} = \text{ukupnaCijena}_{s,r} / \text{brojKorisnika}_{s,r} \quad (6.3)$$

$\text{ukupnaCijena}_{s,r}$ je ukupna cijena rute *r* u rješenju *s*, a $\text{brojKorisnika}_{s,r}$ je broj korisnika u ruti *r* u rješenju *s*. Za primjer na slici 6.3 prikazane su odgovarajuće vrijednosti u tablici 6.1.

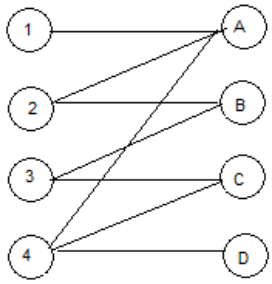
Križanje je implementirano u metodi *UniformnoKrizanje* u razredu *GA*.

Tablica 6.1 Vrijednosti kvocijenta $R_{s,r}$ za gornji primjer

P1	P2
$R_1 = 8.0$	$R_A = 5.0$
$R_2 = 6.0$	$R_B = 7.0$
$R_3 = 4.7$	$R_C = 4.7$
$R_4 = 3.2$	$R_D = 5.3$

Cilj je prenijeti u dijete što više očuvanih ruta roditelja, ali ne mogu se prenijeti u izvornom obliku one rute koje kolidiraju kod oba roditelja. Zato se i računa koeficijent i prema njemu se biraju rute. Na početku se uzima ruta iz prvog roditelja (P1) koja ima najmanji koeficijent R. Tada se iz drugog roditelja (P2) izuzimaju sve rute koje s njom

kolidiraju, tj. one koje u P2 prolaze kroz iste korisnike kroz koje prolazi i izabrana ruta iz P1. U implementaciji su definirana dva asocijativna polja (*hashset*); u jednog se spremaju odabrane rute, a u drugog rute iz P2 koje su izuzete. Nakon toga se iz P2 biraju preostale rute (ako ih ima više, bira se opet ona s najmanjim koeficijentom), a iz P1 izuzimaju se one koje s odabranom kolidiraju. Postupak se ponavlja dok ili sve rute nisu odabrane ili sve preostale nisu izuzete. Dijete će činiti samo odabrane rute. Ako na kraju ostane neiskorištenih korisnika, oni se ubacuju na bilo koje mjesto bilo koje rute u dijete.



Slika 6.4 Povezanost ruta u oba roditelja

U ovom primjeru uzima se prvo ruta 4 iz P1, tada se iz P2 izuzimaju one koje s njom kolidiraju, a to su rute A, C i D, što se vidi iz slike 6.4. Nakon toga se iz P2 odabire ruta B (jedina koja je i preostala) i iz P1 se izuzimaju one koje s njom kolidiraju, a to su rute 2 i 3. Nakon toga se iz P1 odabire jedna od onih koja je ostala, u ovom slučaju ruta 1. Iz P2 se izuzimaju one koje kolidiraju s rutom 1, a to je ruta A. Postupak tada prestaje, jer u P2 nije preostala nijedna ruta. Odabrane su rute 1, B i 4 i one će činiti dijete.

6.6 Operatori mutacije

U razredu *GA* implementirano je i 5 različitih operatora mutacije. Definirana je i vjerojatnost mutacije. Odabire se slučajan broj između 0 i 1 i ako je taj broj manji od vjerojatnosti mutacije, tada će se jedinka iz liste dobivene nakon križanja mutirati, a ako je veći ostat će nepromijenjena.

6.6.1 Invertirajuća mutacija

Uzima se podruta i obrće se redoslijed korisnika te podrute.

6.6.2 Mutacija odabirom slučajnog čvora

Odabire se slučajan čvor u nekoj ruti i on se ubacuje na neko slučajno mjesto u drugu rutu.

6.6.3 Mutacija odabirom slučajne rute

Odabire se podruta slučajno odabrane rute i ubacuje se na slučajno mjesto u drugu slučajno izabranu rutu.

6.6.4 Mutacija zamjenom čvorova

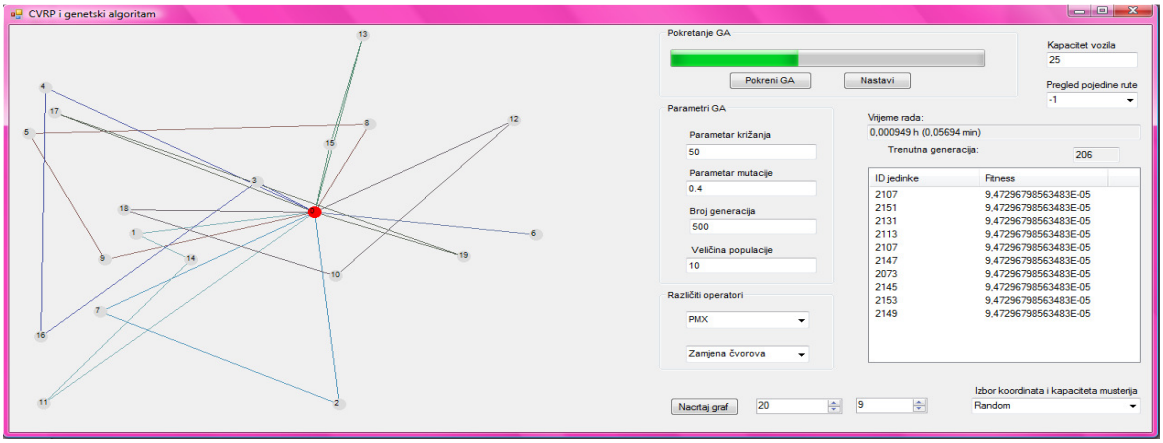
Slučajno se izaberu dva čvora (u istoj ili različitim rutama) i zamjenjuju im se mjesta.

6.6.5 Miješana mutacija

Svakom je od prethodno opisanih operatora pridijeljen broj od 1 – 4. Miješana mutacija u svakoj generaciji slučajno izabire koji će se od četiri operatora primijeniti.

7. Grafičko sučelje

Na grafičkom se sučelju moraju unijeti parametri GA: vjerojatnost križanja, vjerojatnost mutacije, broj generacija, veličina populacije. Može se izabrati i broj korisnika i broj vozila. Ako se u izborniku (*Izbor koordinata i kapaciteta korisnika*) odabere "Random", tada će se graf (raspored korisnika) generirati slučajno, a ako se odabere neka od ponuđenih datoteka, tada se korisnici u grafu crtaju prema koordinatama zadanim u datoteci (za broj vozila mora se odabrati broj kakav je naveden u naslovu datoteke, npr. " A-n65-k9.vrp", broj vozila je 9 i takav se mora odabrati, također, mora se za kapacitet vozila unijeti broj kakav piše u datoteci (za datoteke koje se mogu izabrati, kapacitet je 100). Klikom na gumb *Nacrtaj graf*, crta se graf. Klikom na gumb *Pokreni GA*, pokreće se genetski algoritam. U tijeku rada (ali i ako je korisnik pauzirao rad), može se u izborniku (*Pregled pojedine rute*), vidjeti svaka pojedina ruta ("-1" za sve rute, a ostalo su indeksi ruta). U svakoj se generaciji iscrtava najbolja jedinka. Algoritam ne može početi s radom ako nisu unesene sve vrijednosti potrebne za njegov rad (u tom se slučaju ispisuje upozorenje o tome koja vrijednost još nije unesena). Izgled grafičkog sučelja prikazan je na slici 7.1. U *listview* kontroli se ispisuju ID i dobrota svih jedinki u populaciji trenutne generacije (jedinke su poredane od najbolje prema lošijima). Također, može se vidjeti i broj trenutne generacije i vrijeme izvršavanja algoritma.



Slika 7.1 Izgled grafičkog sučelja

8. Rezultati mjerenja

Eksperimentiranjem se nastojao utvrditi utjecaj različitih parametara genetskog algoritma i različitih operatora križanja i mutacije na razvoj rješenja.

8.1 Rješenja u ovisnosti o veličini populacije

Mjerenja su izvršena u 5000 generacija, s vjerojatnošću mutacije 0.01 i vjerojatnošću križanja 0.5 na problemu od 34 korisnika i 5 vozila. Za križanje je izabran operator PMX i za mutaciju inverzna mutacija. Za svaku tisućitu generaciju ispisivala se vrijednost funkcije dobrote najbolje jedinke i na kraju je izračunata srednja vrijednost. U tablici 8.1 prikazane su srednje vrijednosti funkcije dobrote najboljih jedinki za 10 pokretanja algoritma za svakih tisuću generacija.

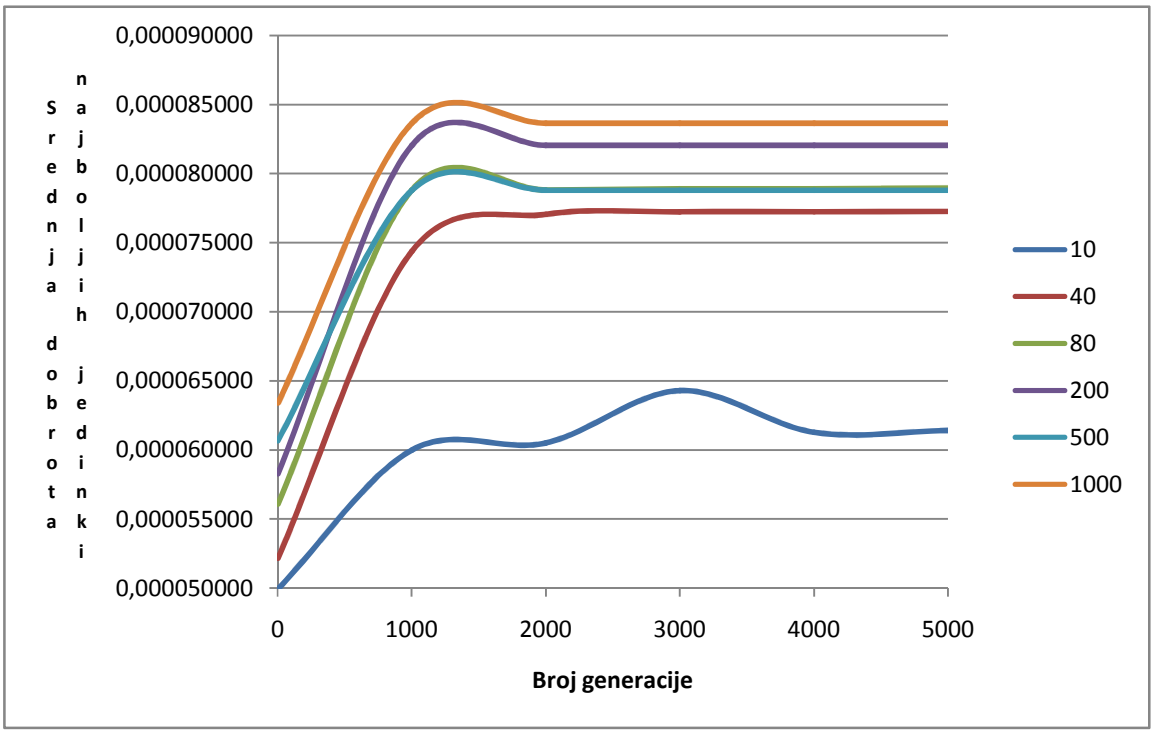
Tablica 8.1 Prikaz srednjih vrijednosti dobrote najboljih jedinki za svaku generaciju u ovisnosti o veličini populacije

			A-n34-k5.vrp		
		Veličina populacije			
Broj generacija	10	40	80	200	500
2	0,000049931	0,000052170	0,000056092	0,000058270	0,000060673
1000	0,000060002	0,000074376	0,000078803	0,000082027	0,000078787
2000	0,000060507	0,000077058	0,000078803	0,000082027	0,000078788
3000	0,000064299	0,000077231	0,000078895	0,000082027	0,000078788
4000	0,000061276	0,000077231	0,000078895	0,000082027	0,000078788
5000	0,000061415	0,000077248	0,000078943	0,000082027	0,000078788

Vrijednosti iz tablice 8.1 unesene su u graf na slici 8.1. Ispitivao se utjecaj na rješenja za veličine populacije 10, 40, 80, 200, 500 i 1000 jedinki.

Veličina populacije trebala bi biti dovoljno velika, ali opet ne prevelika, jer bi tada algoritam bio neučinkovit. Iz grafa se može isčitati da veličina populacije od 10 jedinki ne daje zadovoljavajuća rješenja, čak ni nakon 5000 generacija. Za ostale veličine populacije algoritam konvergira prema boljem rješenju brzo (već nakon otprilike 1000 generacija), ali brzo i zaglavi u lokalnom optimumu. Za veličine populacije 200 i 500

jedinki srednja vrijednost dobrote najboljih jedinki se ne mijenja, dok se kod veličina populacije od 40 i 80 jedinki mijenja; promjene su male, ali algoritam ipak pronalazi bolja rješenja).



Slika 8.1 Graf srednjih vrijednosti dobrote najboljih jedinki u ovisnosti o veličini populacije

8.2 Rješenja u ovisnosti o vjerojatnosti mutacije

Kako bi se utvdio utjecaj vjerojatnosti mutacije na kvalitetu rješenja, algoritam je za problem od 34 korisnika i 5 vozila pokrenut 10 puta uz parametre 5000 generacija, vjerojatnost križanja 0.5 i veličinu populacije 50. Za vrijednosti vjerojatnosti mutacije uzete su 0.1, 0.4, 0.8, 1 i 5 %. U tablicu 8.2 unesene su srednje vrijednosti najboljih rješenja iz 10 pokretanja za svakih tisuću generacija.

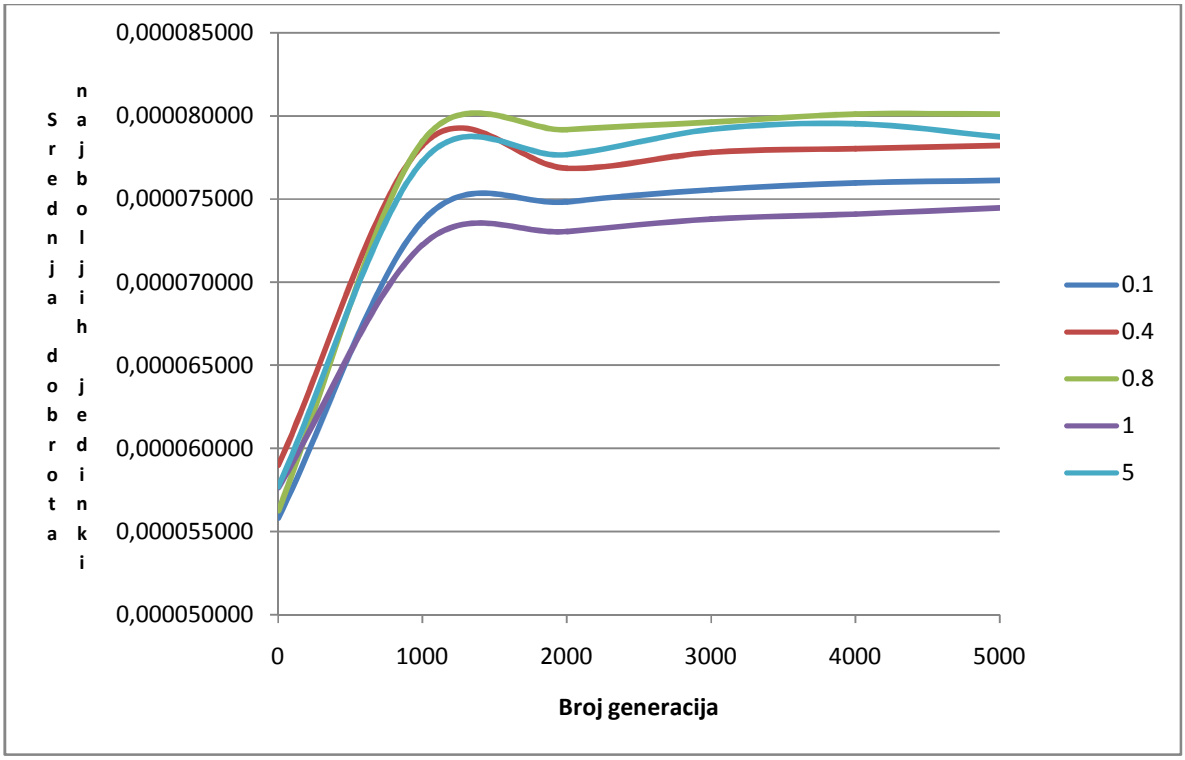
Iz grafa na slici 8.2 može se vidjeti da su rješenja kvalitetnija za veće vjerojatnosti mutacije. Iznimka je na ovom primjeru jedino vjerojatnost 1%. Što je veća vjerojatnost mutacije, to će više jedinki sudjelovati u mutaciji. Vjerojatnosti 0.008 i 0.5 daju

najbolja rješenja; za vjerojatnost 0.5 i 0.008 ista se kvaliteta rješenja dobija za 1000, odnosno oko 3500 generacija.

Tablica 8.2 Prikaz srednjih vrijednosti dobrote najboljih jedinki za svaku generaciju u ovisnosti o vjerojatnosti mutacije

			A-n34-k5.vrp		
		Vjerojatnost mutacije			
Broj generacija	0.1	0.4	0.8	1	5
2	0,000055810	0,000058997	0,000056268	0,000057643	0,000057711
1000	0,000073652	0,000078216	0,000078464	0,000072234	0,000077265
2000	0,000074821	0,000076840	0,000079167	0,000073042	0,000077668
3000	0,000075545	0,000077807	0,000079618	0,000073797	0,000079178
4000	0,000075966	0,000078027	0,000080106	0,000074095	0,000079522
5000	0,000076119	0,000078210	0,000080106	0,000074462	0,000078743

Vrijednosti iz tablice unesene su u graf na slici 8.2.



Slika 8.2 Graf srednjih vrijednosti dobrote najboljih jedinki u ovisnosti o vjerojatnosti mutacije (u legendi označeno u %)

8.3 Rješenja u ovisnosti o vjerojatnosti križanja

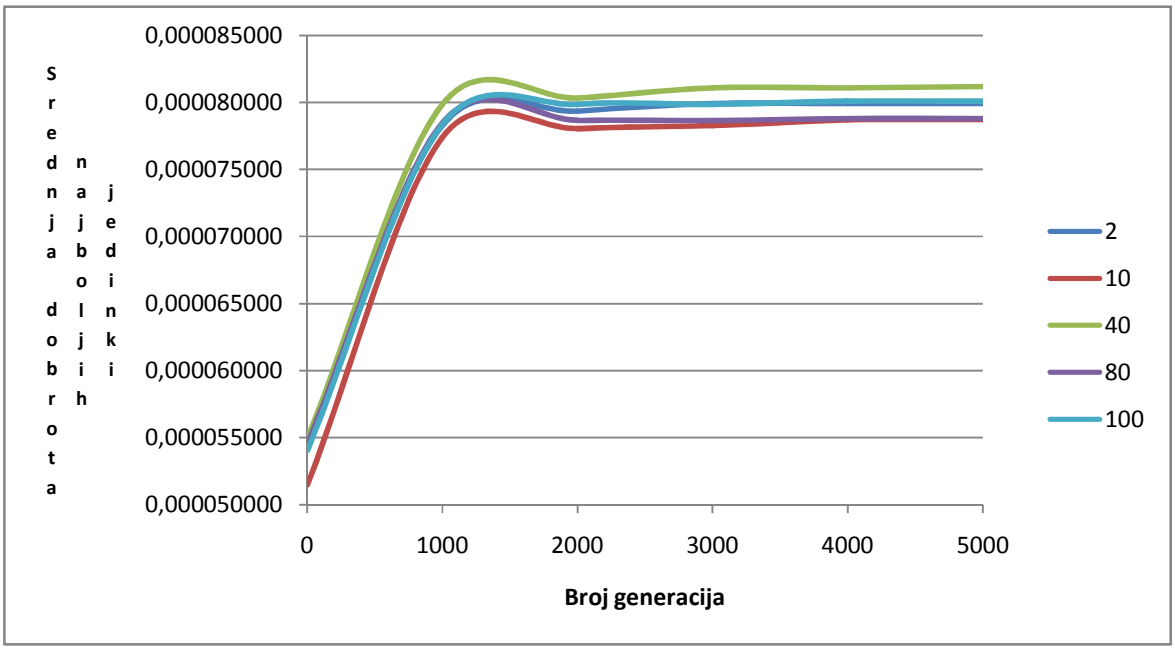
Za ovo su mjerenje korišteni isti parametri kao i u prethodnom slučaju. Vjerojatnost mutacije bila je 0.8%, budući da su rezultati u prethodnom mjerenju pokazali kako se velika vrijednost dobrote unutar 5000 generacija dobiva upravo za tu vjerojatnost mutacije.

Rezultati su opet pokazali da se bolja rješenja dobivaju za veću vjerojatnost križanja. Za ovu veličinu populacije najbolje je za parametar uzeti vjerojatnost križanja 40%. Za najmanju ispitivanu vjerojatnost (10%) može se utvrditi da ne daje baš dobra rješenja, iako se nakon 5000 generacija u jednom trenutku dobije otprilike jednaka kvaliteta rješenja kao i za vjerojatnost 80%.

Tablica 8.3 Prikaz srednjih vrijednosti dobrote najboljih jedinki za svaku generaciju u ovisnosti o vjerojatnosti križanja

			A-n34-k5.vrp		
		Vjerojatnost križanja			
Broj generacija	2	10	40	80	100
2	0,000054631	0,000051499	0,000054963	0,000054565	0,000054075
1000	0,000078266	0,000077388	0,000079849	0,000078451	0,000078373
2000	0,000079344	0,000078047	0,000080312	0,000078657	0,000079855
3000	0,000079909	0,000078278	0,000081081	0,000078657	0,000079858
4000	0,000079915	0,000078695	0,000081081	0,000078794	0,000080095
5000	0,000079915	0,000078735	0,000081184	0,000078794	0,000080095

Iz srednjih vrijednosti najboljih jedinki vidljivo je da se za pojedinu vjerojatnost iz generacije u generaciju pronalazi sve bolje rješenje, dok se za neke vrijednost dobrote ne mijenja kroz više generacija.

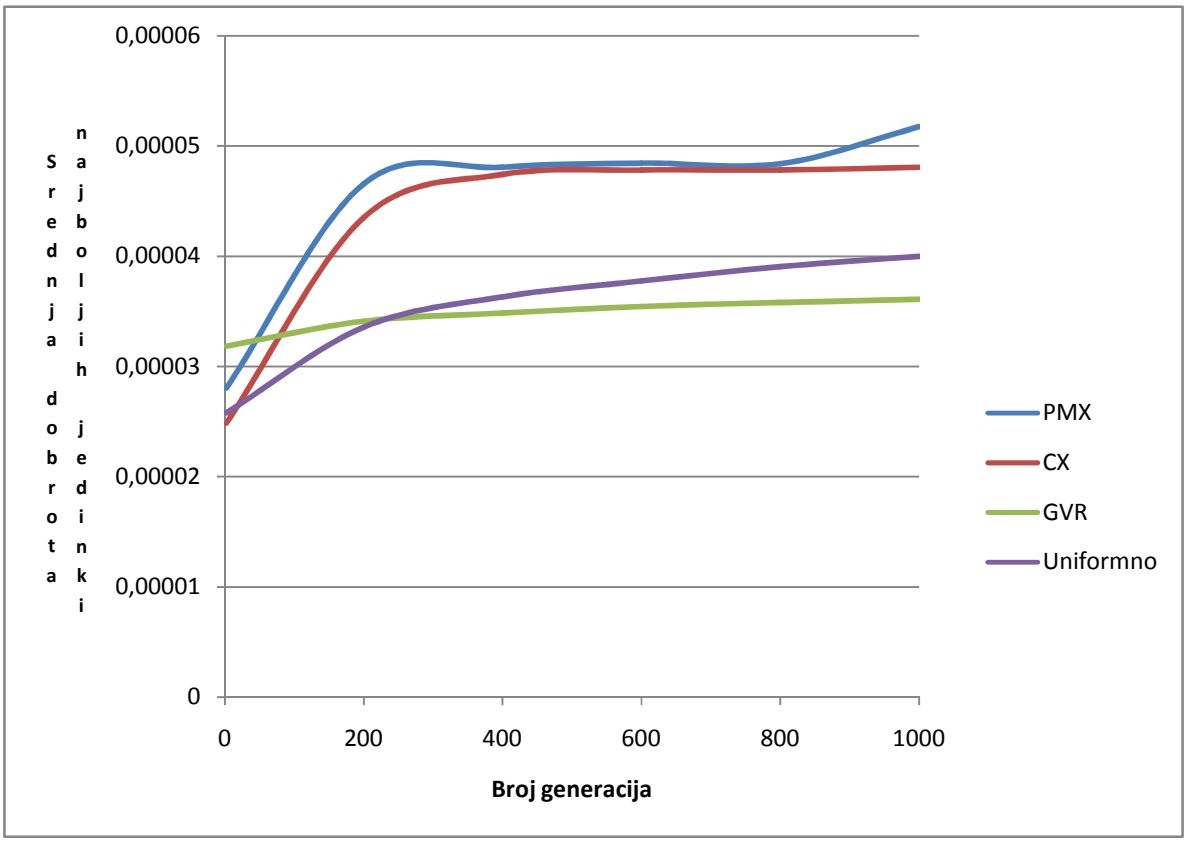


Slika 8.3 Graf srednjih vrijednosti dobrote najboljih jedinki u ovisnosti o vjerojatnosti križanja (u legendi označeno u %)

8.4 Rješenja u ovisnosti o različitim operatorima križanja

Tablica 8.4 Prikaz srednjih vrijednosti dobrote najboljih jedinki za svaku generaciju u ovisnosti o različitim operatorima križanja

		A-n65-k9.vrp		
Broj generacije	PMX	CX	GVR	Uniformno
2	0,000028043	0,000024875	0,000031835	0,0000258
200	0,000046577	0,000043523	0,000034103	0,000033566
400	0,000048072	0,000047414	0,000034835	0,000036318
600	0,000048436	0,000062021	0,000035436	0,000037747
800	0,000048378	0,000047807	0,000035809	0,000039053



Slika 8.4 Graf srednjih vrijednosti najboljih jedinki u ovisnosti o odabiru različitih operatora križanja

Vrijednosti parametara koji su se koristili pri ovom mjerenju su: 1000 generacija, veličina populacije 50, vjerojatnost križanja 0.8 i vjerojatnost mutacije 0.2 i miješana mutacija. Algoritam je za svaki operator ktižanja pokrenut 30 puta (osim za uniformno križanje 20 puta), svakih dvjesto generacija ispisivala se vrijednost funkcije dobrote najbolje jedinke i računala se srednja vrijednost. U tablici 8.4 prikazane su srednje vrijednosti dobrote najboljih jedinki svakih dvjesto generacija

Iz grafa na slici 8.4 je vidljivo da (unutar 1000 generacija) PMX i CX postižu kvalitetna rješenja, dok su rješenja dobivena GVR i uniformnim križanjem nešto lošija. Iz grafa se vidi i da su srednje vrijednosti u generacijama do tisuću također nešto manje (a samim time i lošije) za GVR i uniformno križanje. PMX i CX su uobičajena križanja za ovakve vrste problema i rade samo s redom obilaska (ne cijepaju pa su samim time i

nešto brža od GVR i uniformnog križanja (a to su križanja isključivo za problem usmjeravanja vozila i rad s rutama). Uniformno križanje je sporije, jer za svaku pojedinu rutu jednog roditelja ispituje s kojim se rutama ta ruta preklapa u drugom roditelju i još se za svaku pojedinu rutu računaju cijene koje se dijele s brojem vrhova koje ta ruta sadrži. Također, uniformno križanje i GVR ne rade s redovima obilaska, već red obilaska cijepaju na rute. Prema grafu, u prvih 200 generacija korištenjem uniformnog križanja dobivaju se lošija rješenja od križanja GVR, a nakon toga uniformno križanje naglo konvergira prema kvalitetnijim rješenjima.

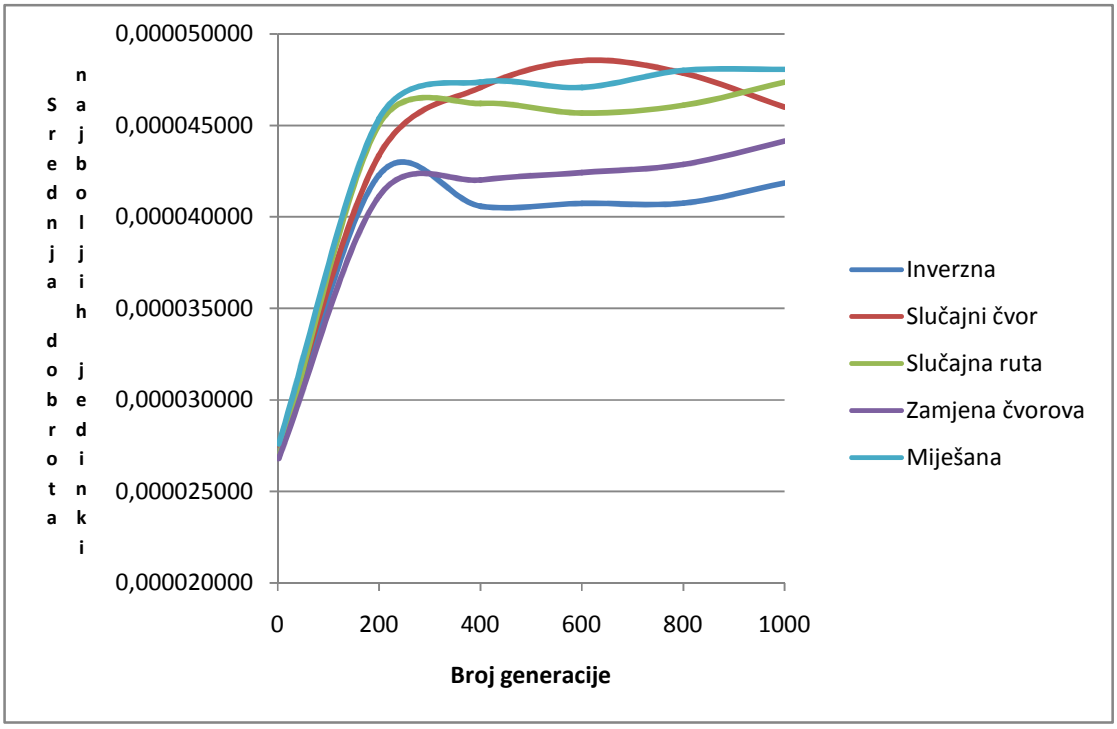
Rješenja u ovisnosti o različitim operatorima mutacije

Tablica 8.5 Prikaz srednjih vrijednosti dobrote najboljih jedinki za svaku generaciju u ovisnosti o različitim operatorima mutacije

			A-n65-k9.vrp		
Broj generacije	Inverzna	Slučajni čvor	Slučajna ruta	Zamjena čvorova	Miješana
2	0,000026892	0,000027534	0,000026854	0,000026782	0,000027591
200	0,000042280	0,000043354	0,000045045	0,000041123	0,000045361
400	0,000040585	0,000047059	0,000046198	0,000042014	0,000047365
600	0,000040736	0,000048532	0,000045675	0,000042415	0,000047071
800	0,000040746	0,000047848	0,000088070	0,000042853	0,000047995
1000	0,000041863	0,000045998	0,000047345	0,000044143	0,000048056

Parametri koji su korišteni pri ovom mjerenju: vjerojatnost križanja 0.5, vjerojatnost mutacije 0.004, veličina populacije 100 jedinki. Odabran je operator križanja PMX, budući da su rezultati pokazali da se, u usporedbi s ostalim operatorima križanja, tako dobivaju bolja rješenja.

Iz grafa na slici 8.5 vidi se da se u početku (tek prvih 200 generacija) kvalitetnija rješenja mogu dobiti pri korištenju miješane mutacije, a najlošija pri korištenju mutacije zamjenom čvorova. Korištenjem inverzne mutacije nakon dvjesto generacija dobivaju se dosta dobra rješenja. Za šesto generacija dobivaju se najbolja rješenja uz mutaciju koja odabire slučajni čvor i premješta ga u neku drugu rutu. Nakon tisuću generacija najbolja je miješana mutacija koja u svakoj generaciji slučajno koristi neku od ostalih mutacija, a najlošija inverzna mutacija.



Slika 8.5 Graf srednjih vrijednosti najboljih jedinki u ovisnosti o odabiru različitih operatora mutacije

9. Zaključak

Problem usmjeravanja vozila je NP težak problem, a ujedno i *real – life* problem. Razvijeni su i prilagođeni mnogi algoritmi koji taj problem mogu riješiti, jedan od njih je i genetski algoritam. Kako bi genetski algoritam bio učinkovitiji, potrebno je odabrati dobre parametre algoritma. Rezultati mjerenja pokazali su da se bolja rješenja dobivaju ako se, za veću veličinu populacije, koriste relativno velika vjerojatnost mutacije (oko 1%) i relativno velika vjerojatnost križanja (80 – 90%). Implementirano je i nekoliko različitih operatora križanja i mutacije. Prema rezultatima mjerenja zaključeno je da PMX i CX daju bolja rješenja od GVR i uniformnog križanja. Također, rezultati su pokazali da se miješanom mutacijom i mutacijom koja odabire slučajan čvor i ubacuje ga u drugu slučajno odabranu rutu dobivaju bolja rješenja.

10. Literatura

[1] Golub, M., Genetski algoritam, prvi dio, Zagreb, 2004.

[2] Toth, P., Vigo, D., The vehicle routing problem, SIAM, Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[3] Golub, M., Genetski algoritam, drugi dio, Zagreb, 2004.

[4] Tan, K. C., Lee, L. H., Zhu, K., Q., Heuristics for Vehicle Routing Problem With Time Windows, 1999.

[5] Ralphs, T., Hartman, J., Galati, M., Capacitated Vehicle Routing problem and some related problems, Rutgers University, 2001.

[6] Laporte, G.: "What You Should Know about the Vehicle Routing Problem", Technical Report, GERAD and Canada Research Chair in Distribution Management, 2007.

[7] Xu, Q., Introduction to Job Shop Scheduling Problem, 2001.

[8] *The VRP Web* , URL: http://neo.lcc.uma.es/radi-aeb/WebVRP/, 28. 5. 2009.

[9] *Genetic Algorithms*, URL: http://www.obitko.com/tutorials/geneticalgorithms/index.php, 28. 5. 2009.

[10] Roland, F., Tseng, S., *Traveling Salesman Problem by Genetic Algorithm and Simulated Annealing*, URL: http://wayne.cs.nthu.edu.tw/~roland/fuzzy/index.html#sec#1, 30. 5. 2009.

[11] Tavares, J., Pereira, F.B., Machado P., Costa, E., Crossover and Diversity: A Study avout GVR, Coimbra, Portugal

[12] Bjarnadóttir, S. A., Solving The Vehicle Routing Problem with Genetic Algorithms, Technical University of Denmark, DTU, 2001.

[13] Kovács, Á., Solving The Vehicle Routing Problem with Genetic Algorithm and Simulated Annealing, Högskolan Dalarna, Švedska, 2008

11. Sažetak

Predstavljen je i implementiran jednostavan genetski algoritam za rješavanje kapacitativnog problema usmjeravanja vozila. Implementirana su i četiri različita operatora križanja i pet operatora mutacije. Rezultatima mjerenja pokušala se odrediti međuovisnost pojedinih parametara i operatora genetskog algoritma te njihov utjecaj na kvalitetu rješenja.

11.1 Abstract

A simple genetic algorithm was presented and implemented for solving the capacitated vehicle routing problem. Four different operators of crossover and five operators of mutation have also been implemented. Results of measurements attempted to determine the interdependence of certain parameters and operators of genetic algorithm and their impact on the quality of solutions.

11.2 Ključne riječi

Genetski algoritam (engl. *Genetic Algorithm, GA),* problem usmjeravanja vozila (engl. *Vehicle Routing Problem, VRP).*