SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 541

Raspoređivanje vozila u stvarnom vremenu uz pomoć genetskog programiranja

Petar Čolić

Zagreb, lipanj 2013.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod

Problem raspoređivanja (ili usmjeravanja) vozila je realan problem, vidljiv u svakodnevnom životu: taksi vozila, vozila za odvoz otpada, poštanska i druga transportna vozila. Raspoređivanje vozila svrstava se u kombinatorne probleme, i to NP-teške. To 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.

Popularan pristup problemu raspoređivanja vozila su evolucijski algoritmi. U ovome će se radu koristiti jedan od njih - genetsko programiranje. U genetskom programiranju svaka jedinka, odnosno rješenje, predstavlja zaseban računalni program pomoću kojega se vozila usmjeravaju.

Na početku rada opisani su evolucijski algoritmi, a posebno genetsko programiranje. Nakon toga detaljno je opisan problem usmjeravanja vozila. Opisano je programsko ostvarenje vlastite inačice problema. Na kraju su prikazani rezultati algoritma.

2. Evolucijski algoritmi

2.1. Općenito

Razvojem računarstva i povećanjem procesne moći računala, mnogi kompleksni problemi postali su rješivi. Rješavalo ih se uglavnom tehnikom grube sile (engl. *brute-force*), tj. pretraživanjem cjelokupnog prostora rješenja.

Inženjerski problemi današnjice su sve kompliciraniji i zahtjevniji. Rijetko su rješenja tih problema jednostavna i egzaktna. Na temeljima stohastike, svijet modernog računarstva se sve više okreće empirizmu, pa čak i u slučaju eksplicitno riješenih problema (najčešće zbog složenosti samih rješenja).

Evolucijski algoritmi su postupci optimiranja, učenja i modeliranja, koji se temelje na procesu evolucije u prirodi. Pojave iz biološke evolucije - mutacija, rekombinacija, križanje, selekcija itd. postaju temelj evolucijskih algoritama. Iako na prvi pogled ne postoji jasno vidljiva poveznica između spomenutih domena, nakon kraće analize ipak se dolazi do zaključka kako su principi evolucije – bolji opstaje, lošiji izumire – idealni za pronalaženje dovoljno dobrog rješenja zadanog problema.

Prve ideje o mogućoj primjeni ovakvih algoritama pojavile su se još pedesetih godina 20. stoljeća, ali zbog skromnih mogućnosti tadašnjih računala ostale su nerealizirane i nepoznate široj znanstvenoj javnosti. Šezdesetih godina neovisno su razvijena tri postupka zasnovana na načelima evolucije u prirodi: evolucijsko programiranje (L. J. Fogel), evolucijske strategije (Rechenberg, Schwefel) i genetski algoritmi (J. H. Holland). Nešto kasnije počinje i razvoj genetskog programiranja (J. Koza). Evolucijski algoritmi spadaju u šire područje znanosti o spoznaji (engl. *cognitive science*), a uže u područje inteligentnih algoritama (engl. *computational intelligence*). [1]

2.2. Genetsko programiranje

Iako se teorijski razvijao paralelno s ostalim evolucijskim metodama, koncept genetskog programiranja je svoju pravu vrijednost dosegao tek u devedesetima, kada ga je utvrdio i primijenio John R. Koza, doajen računarske znanosti sa sveučilišta Stanford. Od tada se uočava još veća progresija u razvoju, kako u istraživačkim timovima, tako i kod samog Koze.

Postoje brojne sličnosti, ali i neke bitne razlike od genetskih algoritama. Temeljna je ideja ista: stvoriti neku populaciju početnih rješenja. Djelovanjem genetskih operatora i ocjenjivanjem pojedinih rješenja kroz određeni broj generacija pronalazi se prihvatljivo rješenje.

2.2.1. Osnovni pojmovi i definicije

Genetsko programiranje je automatiziran optimizacijski postupak razvoja računalnih programa, čija je namjena rješavanje većinom složenih problema iz područja računarstva. Koncept je zasnovan na općim idejama iz teorije genetskih algoritama. Najjednostavnije rečeno, konačni cilj genetičkog programiranja je univerzalni računalni program koji nalazi optimalno rješenje određenog problema. [2]

2.2.2. Formalna predodžba računalnog programa

Bez obzira na činjenicu da se genetskom programiranju može pristupiti na više različitih načina i iz više različitih perspektiva, uglavnom se radi o univerzalnom pristupu računalnom programu kao formalnom stablu u kontekstu teorije grafova. Svaki računalni program može se prikazati kao stablo, gdje unutarnji čvorovi stabla imaju ulogu operatora, a listovi ulogu operanada. Pritom su skup operatora (engl. *function set F*) i operanada (engl. *terminal set T*) unaprijed definirani konačni skupovi.

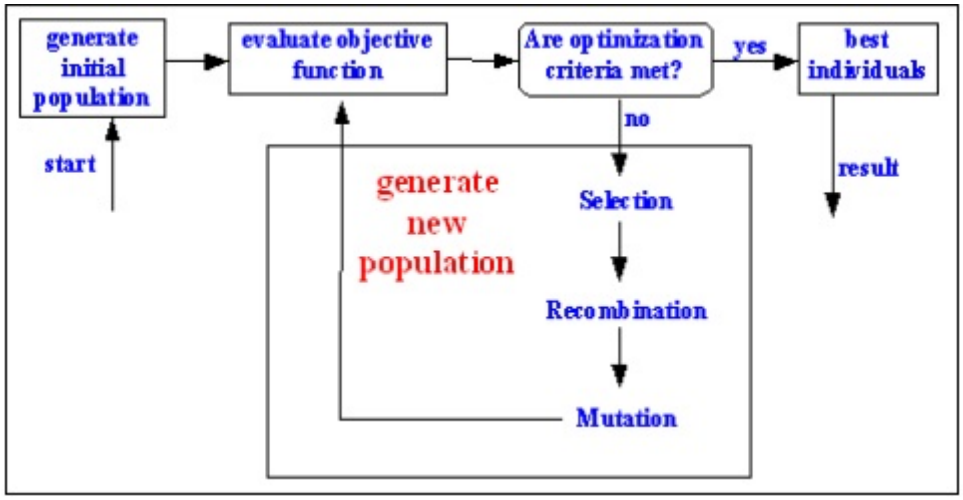
U skladu s općenitom teorijom evolucijskih algoritama, možemo reći da kod GP-a ulogu kromosoma imaju stabla. Upravo su svojstva stabala kao strogih matematičkih objekata, poput jednostavnog rekurzivnog obilaska, odredila razvojni put genetskog programiranja u naglašenom smjeru.

2.2.3. GP kao poseban slučaj evolucijskog algoritma

Iterativni postupak, na kojem se bazira genetsko programiranje, sadrži jasno vidljive elemente općenitog pseudokoda evolucijskih algoritama, a to su:

1. generacija inicijalne populacije rješenja,
2. analiza svake jedinke populacije i populacije općenito,
3. odabir genetskog operatora i provedba odgovarajućih akcija nad pojedinom jedinkom.

Detaljniji pseudokod prikazan je na slici 2.1. [1]



Slika 2.1: Pseudokod evolucijskih algoritama (preuzeto s [1])

2.2.4. Populacija

U biološkom smislu, populacija je skup jedinki iste vrste koje žive u istom prostoru. Razmnožavanjem unutar populacije, ali i umiranjem, veličina populacije je konstantna kroz generativni proces. Što se tiče genetskog programiranja, jedinku predstavlja računalni program, a prostor egzistencije jedinki iz iste populacije je jedna iteracija u optimizacijskom algoritmu. Prema tome, govori se o populaciji računalnih programa jedne iteracije algoritma.

2.2.5. Funkcija dobrote

U cilju rješavanja problema evolucijskim algoritmom, jako je bitno da bolja rješenja ostaju, a ona lošija ne ostaju u daljnjem razmatranju. Upravo dobro definirana funkcija dobrote obavlja tu ulogu. Na temelju raznih parametara, ona određuje dobrotu (engl. *fitness*) pojedine jedinke.

Definiranje funkcije dobrote je jedan od ključnih problema genetskog programiranja. Budući da je njeno evaluiranje prisutno u svakom trenutku generativnog procesa, potrebno je da bude što "bolja", i što jednostavnija.

2.2.6. Genetski operatori

Evolucijski aspekt genetskog programiranja se očituje u načinu optimiranja promatranog programa, gdje su, u ulogama genetskih operatora, prisutne metode reprodukcije i križanja (engl. *crossover*). U nekim slučajevima se javljaju i mutacija, permutiranje i sl.

Reprodukcija

Reprodukcija je jednostavno kopiranje odabrane jedinke i njezino umetanje u novu populaciju. Parametri: vjerojatnost odabira reprodukcije kao genetskog operatora i vjerojatnost odabira pojedine jedinke.

Križanje

Križanje je analogno biološkoj spolnoj reprodukciji. Naime, u tom postupku dolazi do zamjene nekih podstabala odabranih dviju jedinki. Najučestaliji oblik je obavljanje zamjene na dva slučajno odabrana podstabla. Parametri: vjerojatnost odabira križanja kao genetskog operatora, vjerojatnost odabira pojedine jedinke i vjerojatnost odabira pojedinog čvora jedinke kao korijena podstabla.

2.2.7. Selekcija

Tijekom generiranja populacija odvija se selekcija, i to na temelju vjerojatnosti odabira genetskog operatora, te dobrote jedinki. Uobičajeno je da selekcija direktno ovisi o dobroti. Često se koriste proporcionalna (engl. *Roulette-wheel selection*), te turnirska selekcija.

2.2.8. Kriterij prekidanja generativnog postupka

Obzirom na činjenicu da koncept genetskog programiranja s dobro definiranom funkcijom dobrote nad određenim problemom, i adekvatnim skupovima F i T, zadovoljavajuće brzo dolazi do rješenja problema, u velikom broju slučajeva generativni postupak se prekida pri ispunjenju određenog logičkog predikata. Dodatno se uzima i konstanta

G, kao supremum broja iteracija, odnosno broja generacija populacije. Također, zadaje se i supremum M broja jedinki u populaciji, te supremum D dubine generiranih stabala.

2.2.9. Utvrđivanje i opisivanje rješenja

Jednom kada se ispuni kriterij prekida generativnog procesa, iz populacije rješenja se odabire ono najkvalitetnije. Moguće su i poželjne daljnje analize odabranog rješenja, kao i ponavljanje cijelog postupka zbog nezadovoljstva dobivenim rješenjem. Bolje rješenje se uglavnom pokušava naći redefiniranjem funkcije dobrote ili povećanjem supremuma broja iteracija. [6]

3. Problem usmjeravanja vozila

3.1. Općenito

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.

3.2. Opis problema

Mreža prometnica po kojoj se vozila kreću predstavljena je grafom. Bridovi grafa predstavljaju prometnice, a vrhovi korisnike 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 mu se količina robe mora dostaviti ili kod njega preuzeti. 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 koja su u njoj smještena 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 posjećeni 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 preuzeti i drugom dostaviti (engl. *pickup and delivery*).

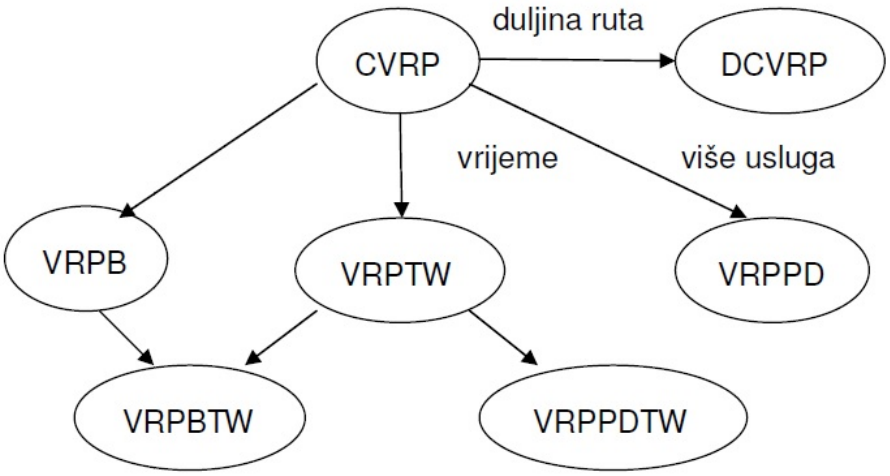
Potrebno je izračunati ukupnu duljinu puta ili vrijeme obilaska između svaka dva korisnika u grafu i između centrale i korisnika. Tada se, radi jednostavnosti, graf koji predstavlja mrežu prometnica pretvara u potpuni graf (engl. *complete graph*). Ako je dan graf G(V, E), gdje je V skup vrhova u grafu, a E skup bridova grafa, tada su određeni bridovi $e_{i,j}$ između svaka dva vrha grafa, a $c_{i,j}$ je cijena između vrha *i* i vrha *j*. Sve se cijene u grafu mogu izraziti uz pomoć matrice cijene. Ona može biti simetrična ili asimetrična, ovisno o tome je li graf usmjeren ili nije.

3.3. Inačice problema

Postoji mnogo različitih vrsta problema usmjeravanja vozila, ovisno o ograničenjima koja su uzeta u obzir. [7]

Na slici 3.1 prikazane su pojedine vrste problema:

- CVRP - kapacitativni VRP (engl. *capacitated VRP*)
- VRPTW - VRP s vremenskim prozorima (engl. *VRP with time windows*)
- VRPB - VRP s povratom (engl. *VRP with Backhauls*)
- VRPBTW - VRPB s vremenskim prozorima (engl. *VRPB with time windows*)
- VRPPD - VRP s podijeljenom isporukom (engl. *VRP with Pickup and Delivering*)
- DCVRP - CVRP s ograničenjem udaljenosti (engl. *Distance – Constrained CVRP*)
- VRPPDTW - VRPPD s vremenskim prozorima (engl. *VRPD with time windows*)



Slika 3.1: Vrste problema usmjeravanja vozila

3.3.1. Kapacitativni VRP

Kapacitativni problem usmjeravanja vozila 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 $e_{i,j}$ između vrhova *i* i *j*. Centrali može odgovarati vrh 0 ili vrh n+1. Svakom bridu grafa pridijeljena je nenegativna cijena $c_{i,j}$. 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 zadovoljava tzv. nejednakost trokuta prikazanu u jednadžbi 3.1.

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

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 centrala ima zahtjev $d_0 = 0$. U centrali je smješteno K jednakih vozila, svako vozilo može prevoziti najveću količinu tereta C (što odgovara njegovom kapacitetu). Pretpostavlja se da za i-ti vrh grafa 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 čvor posjećen najviše jednom
3. Zbroj vrijednosti svih zahtjeva za isporukom robe svih korisnika koji čine neku rutu ne smije prijeći kapacitet vozila koje obilazi tu rutu

Postoji još nekoliko različitih vrsta CVRPa. 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. Također, rute koje sadrže samo jednog korisnika mogu u nekim inačicama CRVPa 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 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 DCVRPu. [7]

3.3.2. Slični problemi

Problem trgovačkog putnika (engl. *travelling salesman problem, TSP*) je blizak problemu usmjeravanja vozila, no nešto jednostavniji. Također postoji početna lokacija (centrala) i lista čvorova, samo što je problem osmišljen za jedno vozilo koje u jednom ciklusu mora obići sve čvorove i vratiti se na početak. Cilj optimizacije je pronaći što kraći ciklus. [4]

4. Programsko ostvarenje

Slijedi opis vlastite implementacije genetskog programiranja za VRP. Projekt je napravljen u programskom jeziku Java, u programerskom okruženju Eclipse.

4.1. Optimizacijski problem

Zadatak je naći što bolje rješenje za vlastitu inačicu VRPa. VRP je postavljen tako da je zadana samo jedna centrala, lista korisnika (čvorova) koje treba obići i broj vozila koja su na raspolaganju. Radno vrijeme vozila traje osam sati dnevno. Na početku i na kraju svakog radnog dana, sva vozila moraju biti u centrali. Svaki čvor, uz koordinate, ima zadano i trajanje, tj. vrijeme potrebno da se korisniku pruži usluga. Za to vrijeme vozilo ostaje kod korisnika. Ako vozilo nema dovoljno vremena (zbog ograničenja od osam sati dnevno), može obaviti dio posla, i sljedeći radni dan nastaviti (i pritom se vratiti u centralu, te opet otići do korisnika). Koristi se euklidska udaljenost između čvorova i pretpostavka je da vozila putuju konstantnom brzinom od 80km/h.

Cilj je pružiti uslugu svim korisnicima u što kraćem vremenu (iskoristiti svih osam sati dnevno) i pritom minimizirati troškove putovanja. Potrebno je razviti takav algoritam koji može u stvarnom vremenu (brzo) odgovoriti na promjene postojećih ili na dodavanje novih čvorova.

4.2. Rješenje

VRP je optimiziran genetskim programiranjem. Postupak se može podijeliti na četiri glavna koraka:

1. Stvori inicijalnu populaciju jedinki (rješenja).
2. Evaluiraj populaciju.
3. Čuvaj dobre jedinke i nad njima obavi operatore križanja i mutacije.

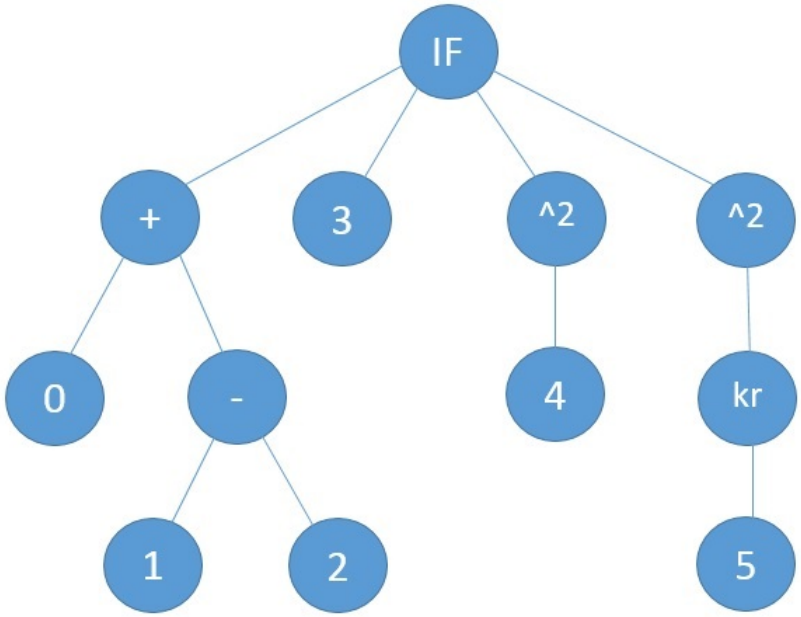
4. Ponavljaj 2-3.

4.2.1. Prikaz jedinke

Svaka jedinka odnosno rješenje predstavljeno je stablom. Stablo predstavlja zaseban računalni program koji može za bilo koju lokaciju proizvesti neku vrijednost (prioritet). Pomoću tih vrijednosti odabire se sljedeća destinacija za pojedino vozilo. Listovi stabla su operandi odnosno akcije, a ostali čvorovi operatori odnosno funkcije. Skup svih mogućih čvorova odnosno akcija i funkcija čine:

1. zbrajanje,
2. oduzimanje,
3. množenje,
4. kvadriranje,
5. korjenovanje,
6. "IF" => funkcija grananja, ima četvero djece, ako je d1>d2 vrati d3 inače d4,
7. udaljenost od centrale,
8. trajanje usluge,
9. cijena usluge,
10. 50 (konstanta),
11. 0.2 (konstanta),
12. x koordinata lokacije,
13. y koordinata lokacije,
14. udaljenost od trenutne lokacije vozila,
15. x koordinata trenutne lokacije vozila,
16. y koordinata trenutne lokacije vozila.

Skup funkcija čine čvorovi 1-6, a skup akcija čvorovi 7-16. U čvoru 6, u slučaju da je uvjet (prvo dijete ima veću vrijednost od drugog) ispunjen, prelazi se na treće (s lijeva), a u suprotnom, na desno dijete (podstablo). Zbrajanje, oduzimanje i množenje imaju dvoje djece, a kvadriranje i korjenovanje jedno. Cijena usluge nema utjecaj na ukupan profit jer će se uvijek sve lokacije obići. Imat će utjecaja samo ako se cijene mijenjaju s vremenom.



Slika 4.1: Primjer stabla

Zapis stabla u računalu ostvaren je kao niz znakova. Znak može biti slovo ili broj. Brojevi označavaju listove, a slova funkcije. Slovo A odgovara navedenoj funkciji 1, slovo B čvoru 2 itd. Broj 0 odgovara čvoru 7, broj 1 čvoru 8 itd. Djeca nekog čvora zapisana su, s lijeva na desno, direktno poslije samog čvora, tj. zapis ide "u dubinu". Prvi znak u nizu je ujedno i korijen stabla.

Obilazak stabla, tj. čitanje niza znakova obavlja se s lijeva na desno. Svaki znak odnosno čvor ima svoju težinu. Težina je određena brojem djece. Tako čvorovi 1-3 imaju težinu 2, dok čvorovi 7-16 (listovi) imaju težinu 0. Ukupna težina nekog čvora je težina podstabla kojemu je taj čvor korijen. Obilazak počinje s korjenom koji mora biti funkcija, u obzir dolaze čvorovi 1-6. Aktivira se korijen. Ako treba "skočiti" na drugo dijete, računa se ukupna težina prvog djeteta, pa se za taj iznos pomaknemo u nizu, tako da sljedeći čvor bude upravo drugo dijete.

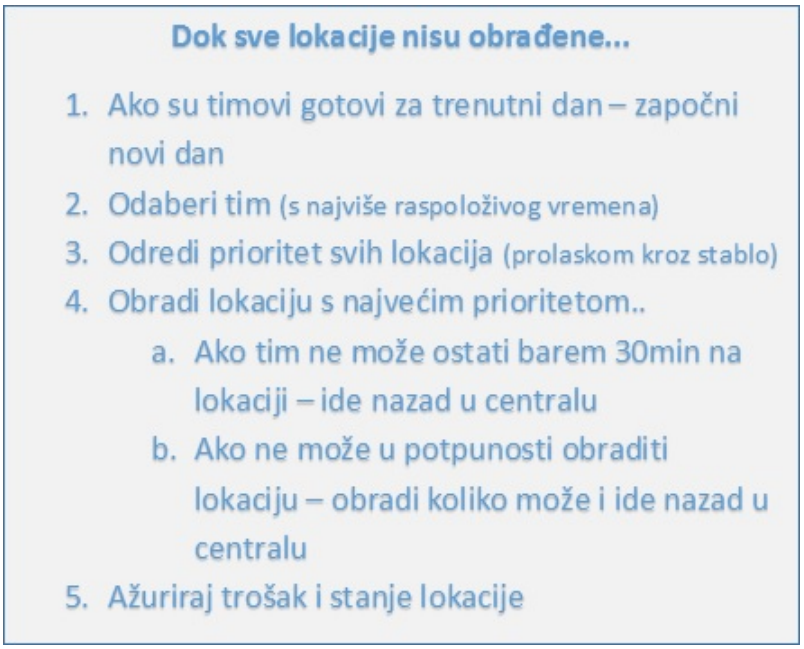
Generiranje početne populacije jedinki odnosno stabala, odvija se uz ograničenja. Maksimalan broj čvorova u stablu je 35, a minimalan 7. Odabir čvorova u početnoj

populaciji je slučajan. Pazi se samo da je broj čvorova unutar granica i da nema "praznih" grananja. Slika 4.1 pokazuje stablo "EA0B123D4DF5". Ovakav zapis rješenja omogućuje njegovo korištenje u stvarnom vremenu. Stablo se može iskoristiti za bilo koju lokaciju, pa tako i za neku promijenjenu ili novonastalu.

4.2.2. Evaluiranje populacije

Svaka jedinka se zasebno evaluira. Za svako vozilo jedinka odabire destinaciju, odnosno uzima se lokacija koja daje najveću vrijednost (prolaskom kroz stablo). Ako tim (vozilo) zbog nedostatka vremena ne može u potpunosti obraditi lokaciju, obradi koliko može i vraća se u centralu. Trajanje lokacije se ažurira ovisno o tome koliko je još vremena potrebno da se u potpunosti obradi. Slika 4.2 prikazuje postupak evaluacije jedinke.

Funkcija dobrote je ukupan profit nakon što su sve lokacije obrađene. Cilj je, naravno, maksimizirati profit. Ako se zanemare cijene (postave na nulu), profit će uvijek biti negativan (samo troškovi putovanja). Preživljavanje jedinke, tj. ulazak u novu generaciju ovisi o njezinoj dobroti. Što je veća dobrota, veća je i šansa da će biti odabrana za sljedeću generaciju.



Slika 4.2: Postupak evaluacije jedinke

4.2.3. Genetski operatori

Nad novom generacijom jedinki obavljaju se operatori križanja i mutacije.

U procesu križanja sudjeluju dvije jedinke. Slučajno se odabire podstablo jedne i zamjenjuje sa slučajno odabranim podstablom druge jedinke. Pritom se mora paziti da nova stabla budu u dozvoljenim granicama veličine. Broj križanja nije fiksan, nego ovisi o parametru - vjerojatnosti križanja. Odabir jedinki za križanje je slučajan, ali pazi se da se ne odaberu iste jedinke više puta.

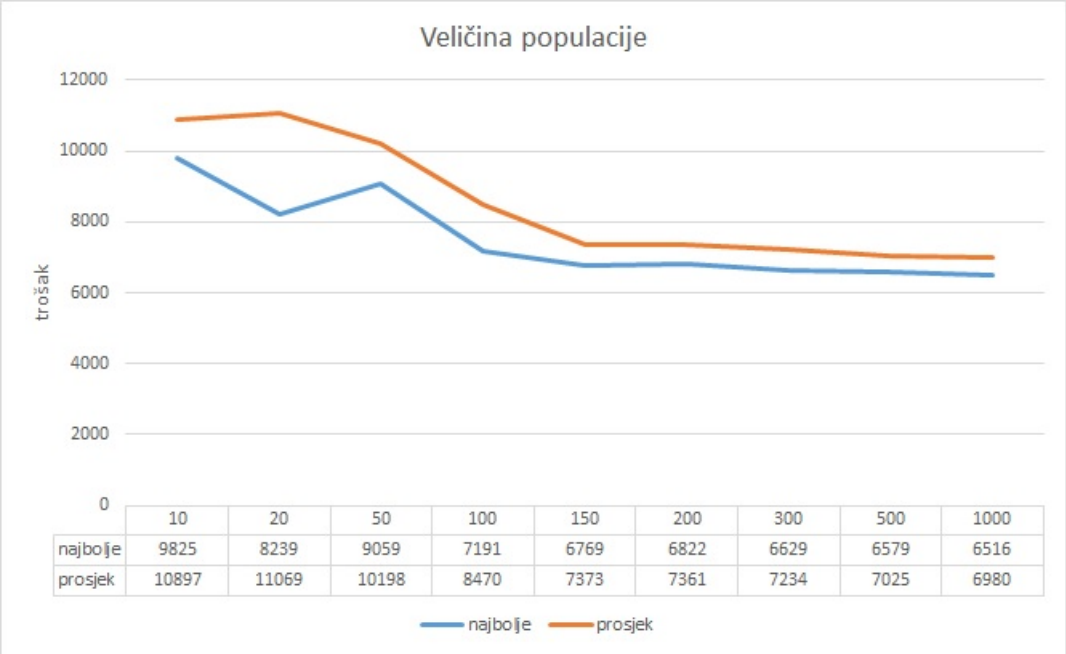
Mutacija stabla je zamjena jednog čvora, skupa s pripadajućim podstablom, s novim slučajno generiranim podstablom. Također se pazi da novo stablo bude dozvoljene veličine. Broj mutacija nije fiksan. Vjerojatnost mutacije svake jedinke također ovisi o parametru.

4.3. Rezultati

Rezultati su podijeljeni u tri kategorije: optimizacija parametara, usporedba s već poznatim problemima (i rezultatima), te ispitivanje naučenih rješenja na neviđenim primjerima.

4.3.1. Optimizacija parametara

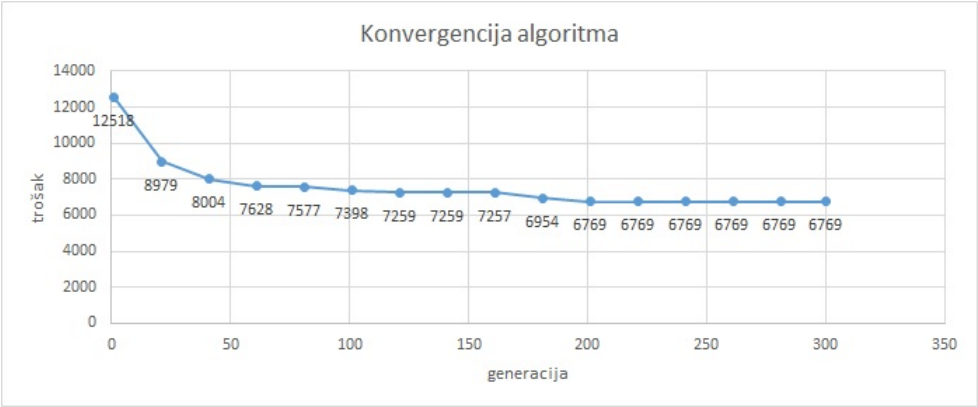
Ispitani parametri vezani uz genetsko programiranje su veličina populacije, vjerojatnost mutacije, te skup dostupnih funkcijskih čvorova. Također treba odrediti kriterij zaustavljanja. Ispitivanja su provedena na primjeru s dvjesto lokacija i četiri raspoloživa vozila.



	10	20	50	100	150	200	300	500	1000
najbolje	9825	8239	9059	7191	6769	6822	6629	6579	6516
prosjek	10897	11069	10198	8470	7373	7361	7234	7025	6980

Slika 4.3: Utjecaj veličine populacije

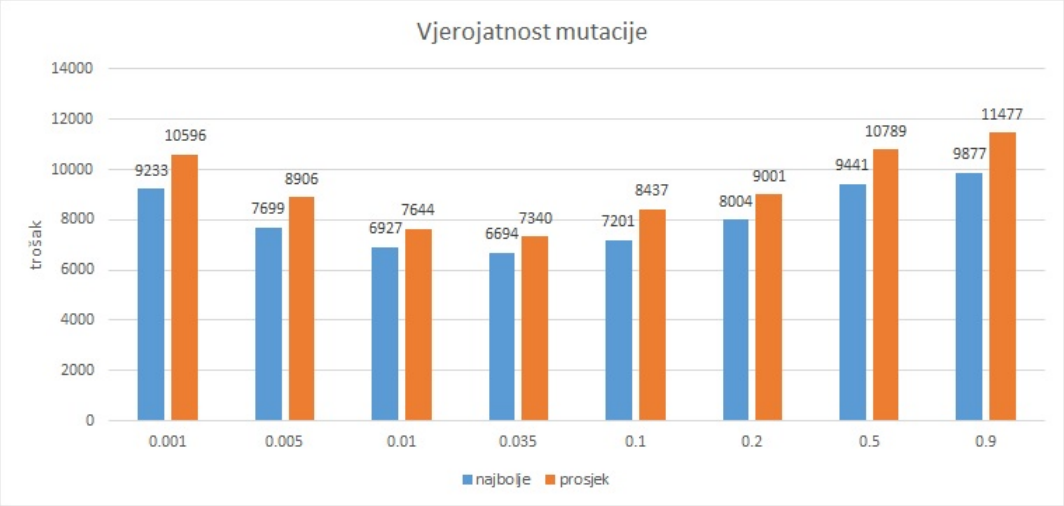
Slika 4.3 prikazuje utjecaj veličine populacije na prosječan i najbolji rezultat algoritma. Korišten je fiksan broj generacija (300) u svakoj iteraciji algoritma. Za svaku vrijednost veličine populacije, algoritam je pokrenut petnaest puta. Zabilježeni su najbolji i prosječni rezultati (od petnaest ponavljanja). Najbolji rezultat i prosjek postignut je uz veličinu populacije 1000. Uzimajući u obzir broj evaluacija jedinki (raste proporcionalno veličini populacije), najbolje je uzeti populaciju veličine 150.



Slika 4.4: Konvergencija algoritma

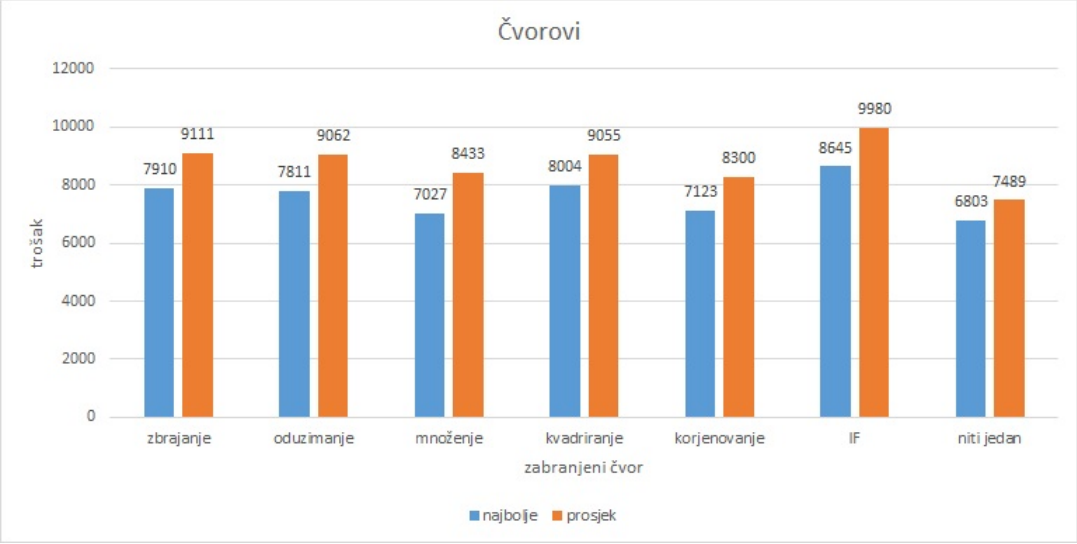
Pomoću ovih rezultata određuje se kriterij zaustavljanja algoritma. Slika 4.4 prikazuje konvergenciju algoritma (na veličini populacije 150). Iz grafa se vidi da algoritam brzo konvergira, te da bi dobar kriterij zaustavljanja bio stagniranje rezultata kroz zad-

njih 80 generacija. Taj kriterij zaustavljanja će se koristiti u daljnjim istraživanjima.



Slika 4.5: Utjecaj vjerojatnosti mutacije

Slika 4.5 prikazuje utjecaj vjerojatnosti mutacije na prosječan i najbolji rezultat algoritma. Najbolji rezultat, a ujedno i najbolji prosjek, postignut je uz vjerojatnost 0.035% (vjerojatnost da će jedinka mutirati). Odabir takve vrijednosti daje algoritmu mogućnost fokusa na lokalne optimume, ali ujedno i na cijeli prostor pretraživanja.

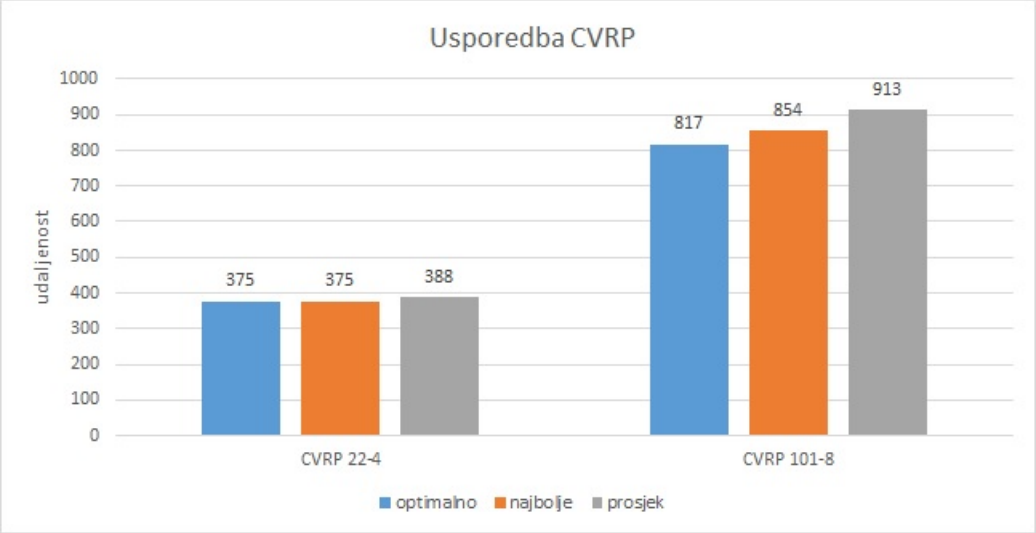


Slika 4.6: Utjecaj odabira mogućih čvorova

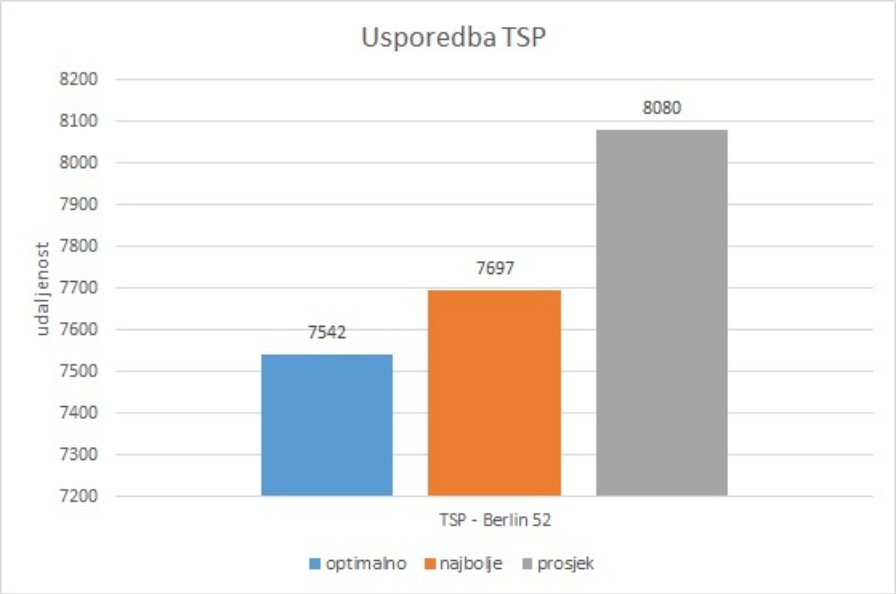
Na slici 4.6 dana je usporedba rezultata kada se izbaci jedan čvor iz skupa mogućih funkcijskih čvorova. Korišteni su optimalni parametri. Najbolji rezultati se i dalje dobivaju kada su svi čvorovi dostupni, no najmanje utjecajni su se pokazali čvorovi ‘korjenovanje‘ i ‘množenje‘. Proizlazi da je najvažniji čvor ‘IF‘.

4.3.2. Poznati problemi

Valja usporediti algoritam s nekim već poznatim rezultatima. Budući da ne postoji verzija VRPa identična vlastitoj, korišteni su simetrični CVRP i TSP. Konkretno, korišteni su CVRPi s 22 lokacije i 4 raspoloživa vozila odnosno sa 101 lokacijom i 8 raspoloživih vozila preuzeti s [5], te TSP s 52 lokacije u Berlinu preuzeto s [3]. Da bi se ovi pokusi mogli izvesti, potrebno je modificirati evaluaciju jedinke. Za TSP evaluacija je pojednostavljena, koristi se samo jedan tim koji u svakom koraku odlazi na lokaciju s najvećim prioritetom. Dostupni čvorovi ostaju isti (nekorišteni terminalni čvorovi postavljaju se na nulu). U CVRPu timovi umjesto slobodnog vremena imaju svoj kapacitet. Čvor 'trajanje usluge' postaje 'potražnja na lokaciji'. U oba slučaja nije potrebno svakodnevno vraćanje timova u centralu nego samo nakon obilaska svih lokacija. Slika 4.7 prikazuje rezultate CVRPa. Rezultati su dobri; u lakšem primjeru postignuto je optimalno rješenje, a u težem je rješenje blizu optimalnog. Kod TSPa su slični rezultati, što prikazuje slika 4.8. Dobiveno je dovoljno dobro rješenje.



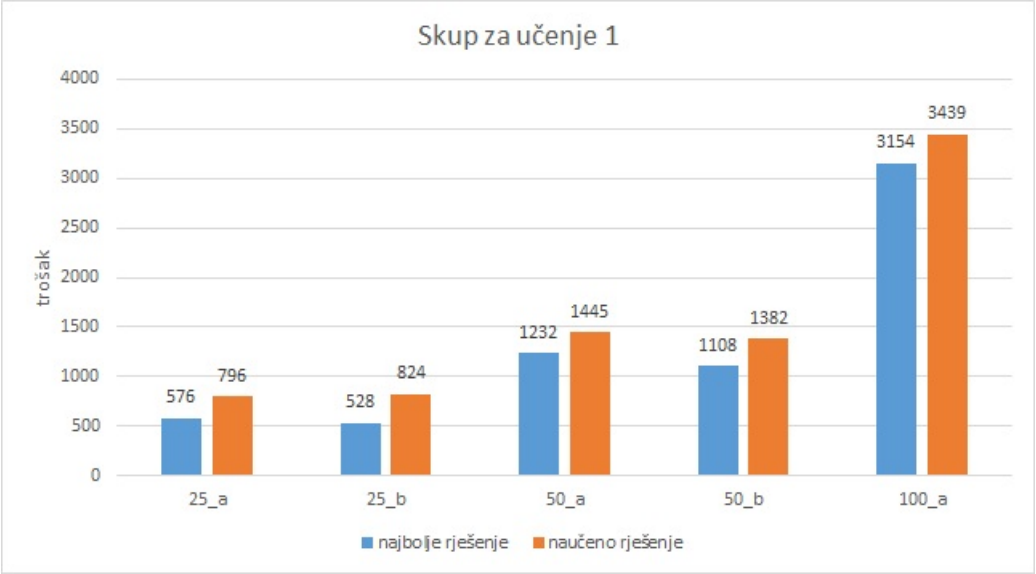
Slika 4.7: Rezultati CVRP



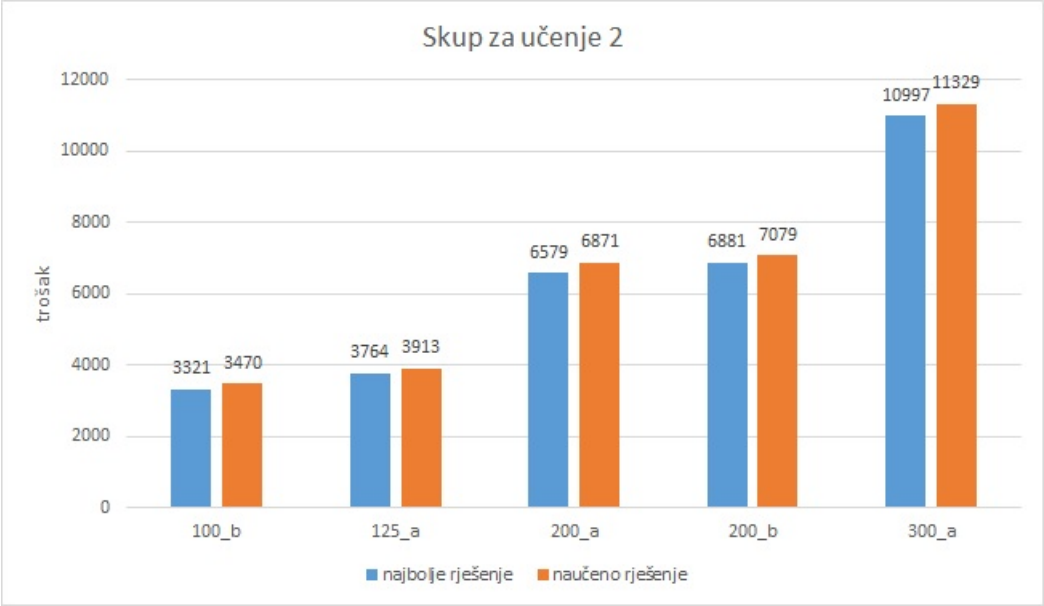
Slika 4.8: Rezultati TSP

4.3.3. Generalizacija

Ispituje se svojstvo generalizacije, tj. koliko je dobiveno rješenje dobro na nekom neviđenom skupu problema. Testni primjeri su podijeljeni u dva skupa: skup za učenje i ispitni skup. Oba skupa sastoje se od vlastitih primjera. Za usporedbu koristi se najbolje rješenje dobiveno pokretanjem algoritma petnaest puta s optimalnim parametrima. Kao naučeno rješenje, uzeta je najbolja jedinka dobivena pokretanjem algoritma (nad skupom za učenje) pet puta (nakon 500 generacija). Slike 4.9 i 4.10 prikazuju rezultate naučenog rješenja na skupu za učenje. Rezultati su dovoljno dobri, tj relativno blizu najboljim pronađenim rješenjima. Veća odstupanja su na primjerima s manjim brojem lokacija, što je razumljivo jer ti primjeri nose mali postotak ukupnog troška.

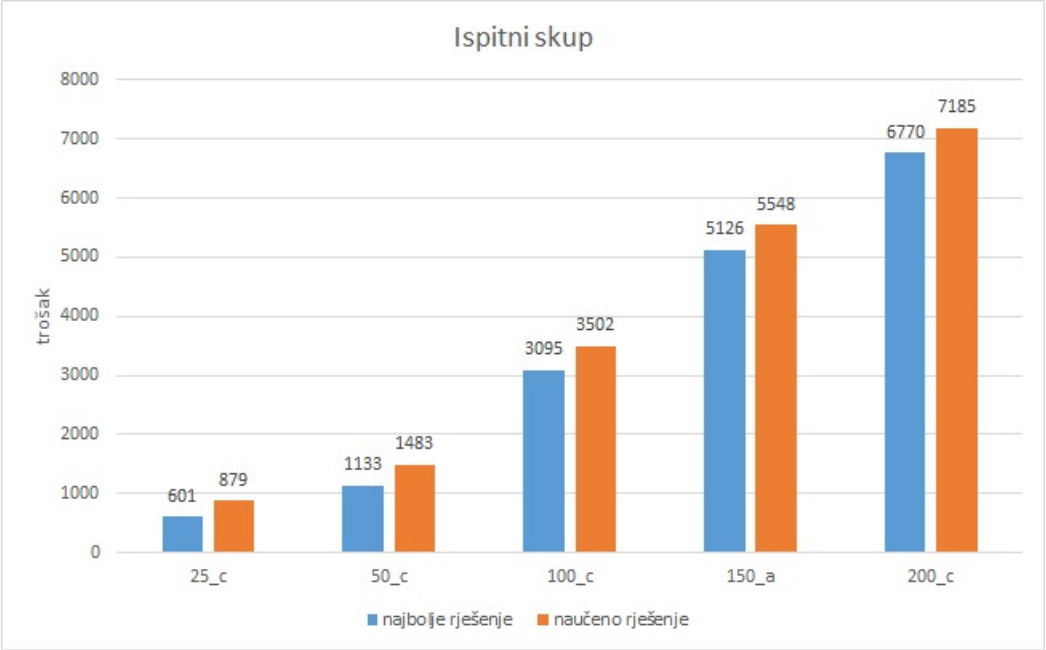


Slika 4.9: Skup za učenje, prvi dio

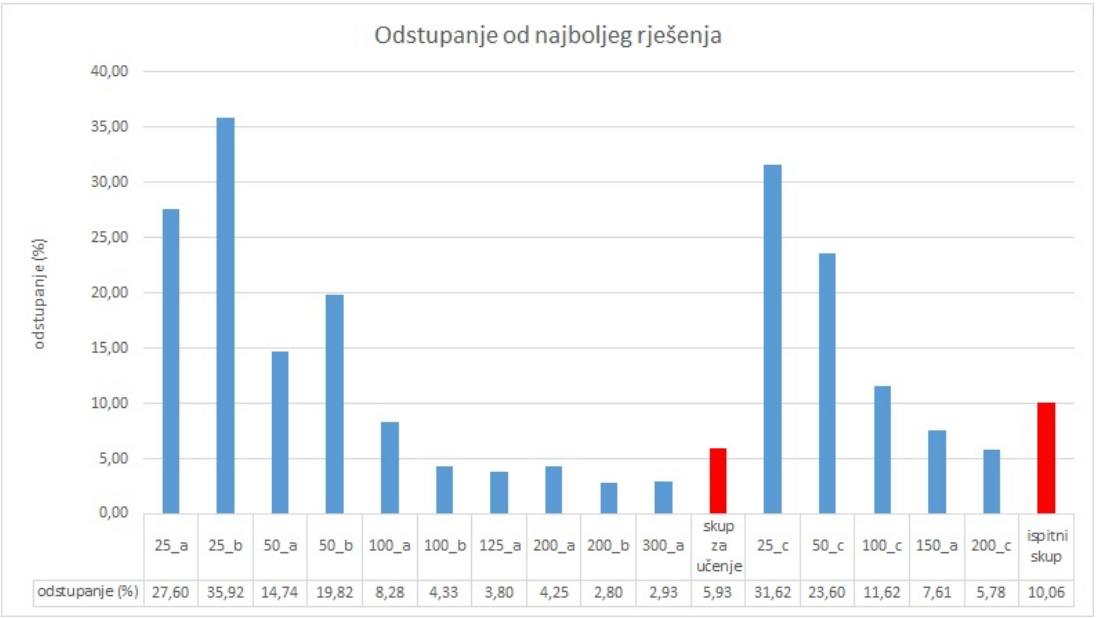


Slika 4.10: Skup za učenje, drugi dio

Slika 4.11 prikazuje rezultate naučenog rješenja nad ispitnim skupom. Odstupanje je veće, što je i očekivano, ali rezultati su dovoljno dobri, odnosno dovoljno blizu najboljima. Na slici 4.12 mogu se vidjeti odstupanja od najbolje dobivenih rješenja u postocima. Kada se gledaju skupovi kao cjeline, ispitni skup ima dvostruko veće odstupanje od skupa za učenje.



Slika 4.11: Ispitni skup



Slika 4.12: Odstupanje naučenog od najboljeg rješenja u oba skupa

5. Zaključak

Postupak optimizacije genetskim programiranjem, u kratkom vremenu, postiže zadovoljavajuće rezultate. Algoritam brzo konvergira prema dobrim rješenjima, a nakon dužeg vremena, pomaci su minimalni. Genetsko programiranje može se mjeriti s alternativnim algoritmima što dokazuju usporedbe s problemima s unaprijed poznatim optimalnim rješenjima. Jednom dobiveno rješenje se može odmah prilagoditi promjenama u stvarnom vremenu, što je jedna od prednosti nad ostalim algoritmima. Rješenja dobivena genetskim programiranjem pokazuju se dovoljno dobra na neviđenim primjerima, što znači da se jednom naučena rješenja mogu koristiti u različite svrhe i u raznim inačicama problema usmjeravanja vozila, ali i sličnim problemima.

LITERATURA

[1] Evolutionary algorithms. http://www.scholarpedia.org/article/Evolutionary_algorithms, 2008. URL http://www.scholarpedia.org/article/Evolutionary_algorithms.

[2] Genetic programming – wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Genetic_programming, 2011. URL http://en.wikipedia.org/wiki/Genetic_programming.

[3] Tsplib. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/, 2013. URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[4] Travelling salesman problem. http://en.wikipedia.org/wiki/Travelling_salesman_problem, 2013. URL http://en.wikipedia.org/wiki/Travelling_salesman_problem.

[5] Vrp web. http://neo.lcc.uma.es/radi-aeb/WebVRP/, 2013. URL http://neo.lcc.uma.es/radi-aeb/WebVRP/.

[6] R. J. Koza. *What is Genetic Programming (GP)?, How Genetic Programming Works*, 2007. URL http://www.genetic-programming.com/.

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

Raspoređivanje vozila u stvarnom vremenu uz pomoć genetskog programiranja

Sažetak

Evolucijski algoritmi su postupci optimiranja koji se temelje na mehanizmu evolucije u prirodi. Kod genetskog programiranja kromosom predstavlja program koji je rješenje zadanog problema. Tipična struktura podataka koja se kod genetskog programiranja koristi za prikaz kromosoma jest stablo. Problem usmjeravanja vozila spada pod NP-teške probleme, pa je idealan za isprobavanje evolucijskih algoritama. Genetsko programiranje se pokazalo kao dobar algoritam s brzom konvergencijom i mogućnošću korištenja naučenih rješenja na drugim testnim primjerima.

Ključne riječi: genetsko, programiranje, algoritmi, evolucija, VRP.

Vehicle routing in real time with genetic programming

Abstract

Evolutionary algorithms are optimization routines and are based on the mechanism of evolution in nature. In genetic programming, a chromosome is a computer program which represents a solution to a given problem. Typical data structure used in genetic programming to display a chromosome is a tree. The vehicle routing problem falls under the NP-hard problems, and is ideal for testing evolutionary algorithms. Genetic programming proved to be a good algorithm with rapid convergence, and has the ability to use learned solutions in other test cases.

Keywords: genetic, algorithm, programming, evolution, VRP.