SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6390

Primjena metaheurističkih algoritama na problem usmjeravanja vozila s vremenskim prozorima i mogućnošću preuzimanja tereta

Jelena Nemčić

Zagreb, lipanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 14. ožujka 2019.

ZAVRŠNI ZADATAK br. 6390

Pristupnik: **Jelena Nemčić (0036497921)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Primjena metaheurističkih algoritama na problem usmjeravanja vozila s vremenskim prozorima i mogućnošću preuzimanja tereta**

Opis zadatka:

Opisati problem usmjeravanja vozila i postojeće metode rješavanja toga problema. Posebnu pažnju posvetiti inačicama problema s ograničenjima u obliku vremenskih prozora i mogućnosti preuzimanja tereta. Ostvariti programski sustav za rješavanje problema usmjeravanja vozila uz pomoć operatora lokalne pretrage i evolucijskih algoritama. Ispitati učinkovitost različitih prikaza rješenja i operatora primijenjenih na navedeni problem. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 14. lipnja 2019.

Mentor:

Prof. dr. sc. Domagoj Jakobović

Djelovođa:

Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Doc. dr. sc. Marko Čupić

Sadržaj

1. Uvod

Problem usmjeravanja vozila (eng. *Vehicle Routing Problem (VRP)*) jedan je od najpoznatijih, najbitnijih i najzahtjevnijih problema kombinatorne optimizacije. Generalizira poznati problem trgovačkog putnika pokušavajući naći odgovor na pitanje *"Koje su rute optimalne za flotu vozila kako bi se obišli svi kupci?"*. Pri određivanju kvalitete rješenja u obzir se uzima ispunjenje određenih zahtjeva. Rješenje ovog problema je minimizacija ukupne cijene rute, koju mogu činiti razni parametri poput prijeđene udaljenosti, vremenskog trajanja, broja vozila i slično. Određivanje optimalne rute svih vozila predstavlja nepolinomijalno (NP) težak problem, te je zbog toga broj problema koji se mogu riješiti egzaktnim metodama ograničen.

Problem usmjeravanja vozila čini središte upravljanja distribucijom i svaki dan tisuće se tvrtki i organizacija susreću upravo s tim problemom. Zato VRP ima mnoge primjene u industriji te računalna optimizacija transporta dobara smanjuje tvrtkama trošak do 5%. S obzirom da transportni sektor čini 10% ukupnog BDP-a Europske Unije već i manja ušteda nastala zbog primjene VRP-a je značajna.

U ovom radu obrađuje se VRP problem s ograničenim kapacitetom vozila (eng. *Capacitated VRP (CVRP)*), CVRP u kombinaciji s VRP problemom s vremenskim prozorima (eng. *VRP with Time Windows (VRPTW)*) te CVRP i VRPTW u kombinaciji s mogućnošću preuzimanja tereta i njegove dostave (eng. *VRP with Pick-Up and Delivering (VRPPD)*). Objašnjena su tri načina implementacije rješenja i na svako od njih primjenjeni su metaheuristički algoritmi, te su pri tome provođene razne usporedbe uspješnosti.

2. Problem usmjeravanja vozila (VRP)

2.1. O nastanku problema

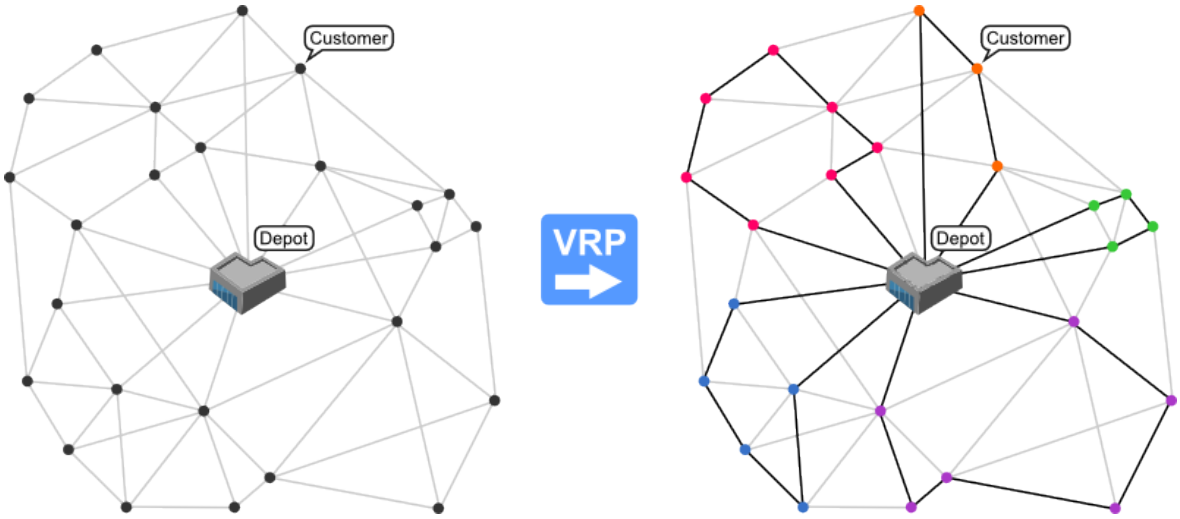
Problem usmjeravanja vozila uveli su 1959. godine G. B. Dantzig i J. H. Ramser. Problem su opisali kao generalizaciju problema trgovačkog putnika (eng. *Travelling Salesman Problem (TSP)*), jer imamo više vozila nego u tom problemu, gdje je jedino vozilo trgovac, i imamo određene restrikcije pri posjećivanju gradova, kao npr. ograničen kapacitet vozila. Oni su dali i prvo algoritamsko rješenje VRP-a koristeći stvaran problem - dostavu nafte benzinskim postajama. 1964. godine Clarke i Wright unaprijedili su algoritam koristeći pohlepnu heuristiku, koji se po tome zove algoritam uštede (eng. *Savings Algorithm*), koji je primjenjiv i na probleme s varijabilnim brojem vozila.

Od tada je VRP problem izrazito uznapredovao, napisani su mnogi radovi na tu temu i nastale su različite varijante samog problema. Osmišljene su mnoge metode nalaženja optimalnog ili dovoljno dobrog rješenja tog problema. Većina njih su heuristički i metaheuristički algoritimi jer zbog NP složenosti problema nijedan egzaktan algoritam ne može u razumnom vremenu naći optimalno rješenje kada je broj kupaca velik.

2.2. Formulacija problema

Problem usmjeravanja vozila je generičko ime dano cijeloj klasi problema koja se bavi određivanjem skupa ruta za flotu vozila, s bazom u jednom ili više skladišta, kako bi obišli geografski raspršene gradove ili kupce. Cilj VRP problema je obaviti dostavu svim kupcima, čiji su zahtjevi unaprijed poznati, s minimalnom cijenom svih ruta. Cijena pojedine rute ovisi o prijeđenoj udaljenosti i o postavljenim ograničenjima. Svaka ruta počinje i završava u skladištu i svaki kupac mora biti posjećen točno jednom. Prikaz jednog VRP problema i njegovog rješenja dan je na slici 2.1.

VRP je kombinatorijski problem koji je formalno definiran kao potpun neusmjeren



Slika 2.1: Prikaz VRP problema (lijevo) i njegovog rješenja (desno)

graf $G = (V, E)$ gdje je:

- $V = \{v_0, v_1, ..., v_n\}$ - set vrhova odnosno kupaca, pri čemu je v_0 skladište, a $V' = V \backslash \{v_0\}$ set od n kupaca
- $A = \{(v_i, v_j) | v_i, v_j \in V; i \neq j\}$ - set lukova odnosno puteva između kupaca
- C - matrica udaljenosti između kupaca (c_{ij} predstavlja udaljenost između kupaca v_i i v_j)
- d - vektor zahtjeva kupaca (d_i je zahtjev kupca v_i)
- R_i - ruta vozila i
- m - broj identičnih vozila na raspolaganju
- δ_i - vrijeme potrebno vozilu za istovar dobara kupcu v_i

Uglavnom se kao mjera uspješnosti pojedinog rješenja uzima njegovo ukupno trajanje odnosno trajanje svih njegovih ruta. Kao vrijeme putovanja između kupaca v_i i v_j najčešće se uzima njihova udaljenost c_{ij}. Tako je trajanje jedne rute $R_i = \{v_0, v_1, ..., v_{k+1}\}$, gdje je $v_0 = v_{k+1} = 0$ (skladište), dano s:

$$C(R_i) = \sum_{i=0}^{k} c_{i,i+1} + \sum_{i=1}^{k} \delta_i \tag{2.1}$$

Pri tome trajanje rute predstavlja i njenu cijenu.

Zahtjeva se da trajanje rute svakog vozila (suma vremena putovanja i vremena istovara) ne smije premašiti određenu granicu D ($C(R_i) \leq D$). Ako je to ispunjeno za svaku rutu i ako svako vozilo tijekom svoje rute posjećuje svakog kupca samo jednom, rješenje koje se sastoji od tih ruta je prihvatiljivo (eng. *feasible*). Ukupna cijena tog

rješenja iznosi:

$$F_{VRP} = \sum_{i=1}^{m} C(R_i) \tag{2.2}$$

Rješenje je to bolje što mu je cijena manja.

2.3. VRP s ograničenim kapacitetom (CVRP)

Jedan od najčešćih oblika VRP problema jest CVRP (eng. *Capacitated VRP*). U toj varijanti problema sva vozila imaju poznat jednak kapacitet C što nameće novi zahtjev da suma količine dobara pojedine rute ne smije premašiti kapacitet vozila. Za svaku rutu R_i koja obilazi kupce $v_1, ..., v_k$ mora vrijediti:

$$\sum_{i=1}^{k} d_i \leq C \tag{2.3}$$

U protivnom, rješenje prestaje biti prihvatljivo.

Uzimajući to u obzir pokušava se minimizirati broj potrebnih vozila i trajanje putovanja, pri čemu se cijena rute i ukupnog rješenja računaju kao što je navedno za VRP. U ovom radu obrađuje se kombinacija CVRP-a s VRPTW-om te s VRPPD-om koji su objašnjeni u daljnjim potpoglavljima.

2.4. VRP s vremenskim prozorima (VRPTW)

Problem usmjeravanja vozila s vremenskim prozorima (eng. *VRP with Time Windows (VRPTW)*) je VRP s dodatnom restrikcijom vremenskog okvira. Svaki kupac definira svoj vremenski interval $[e_0, l_0]$ u kojem treba biti poslužen. Vremenski interval skladišta predstavlja okvir dostave tijekom kojeg svaka dostava mora biti obavljena. Kako bi rješenje bilo prihvatljivo svaka ruta treba početi i završiti unutar vremenskog okvira skladišta i nijedan kupac ne smije biti poslužen nakon gornje granice svog vremenskog prozora. Dolazak do kupca prije njegove donje granice intervala uzrokuje čekanje, ali je ono dopustivo. Cilj je, kao i u VRP-u, minimizirati broj vozila i trajanje dostava te, dodatno, smanjiti i ukupno vrijeme čekanja.

Jedan od najčešće korištenih zapisa problema uveo je Marius M. Solomon i taj zapis korišten je i u ovom radu. Primjeri su podijeljeni u tri kategorije:

- tip C - kupci su položajno grupirani
- tip R - kupci su jednoliko distrubuirani
- tip RC - kombinacija prethodna dva tipa

U svakoj kategoriji postoje primjeri s 25, 50, 100, 200, 400, 600, 800 i 1000 kupaca.

Cijena rute i računa se kao suma:

$$C_{VRPTW}(R_i) = \sum_{i=0}^{k} c_{i,i+1} + \sum_{i=1}^{k} \delta_i + \sum_{i=1}^{k} w_{v_i} \tag{2.4}$$

Pri čemu je w_{v_i} vrijeme čekanja vozila kod kupca v_i.

Na početku se pretpostavlja da sva vozila napuštaju skladište u najranijem mogućem trenutku e_0. Jednom kad se dobije zadovoljavajuće rješenje, vrijeme kretanja svakog vozila se prilagodi vremenu početka vremenskog okvira prvog kupca koji se poslužuje te se time eliminira nepotrebno čekanje.

2.5. VRP s mogućnošću preuzimanja tereta i dostave (VRPPD)

Problem usmjeravanja vozila s mogućnošću preuzimanja tereta i njegove dostava poseban je slučaj VRP-a koji uključuje i CVRP, a često i VRPTW. U toj varijanti razmatra se mogućnost da kupci vrate neku robu vozilu, koju ono tada dostavlja drugom kupcu ili vraća u skladište. Pri tom se mora uzeti u obzir kapacitet svakog vozila i paziti da se on ne premaši.

Moguće pojednostavljenje problema je uvođenje ograničenja da se sva roba odvozi u skladište i da nema razmjene među kupcima. Druga mogućnost je uklanjanje zahtjeva obilaska svakog kupca točno jednom. Također, moguće pojednostavljenje je dostava svih dobara iz skladišta prije preuzimanja robe od kupaca.

Cilj je minimizirati broj vozila i vrijeme putovanja, vodeći računa da vozilo mora imati kapacitet dovoljan za robu koju dostavlja i za onu koju će tek pokupiti kod kupaca. Rješenje je prihvatljivo ako količina dobara koje svako vozilo prevozi ne premašuje njegov kapacitet i ako vozilo u svakom trenutku rute ima dovoljno slobodnog mjesta za preuzimanje određene količine robe od kupaca. To se računa kao:

$$L(v_k) = C_p(v_k) + L(1) - C_d(v_k) \tag{2.5}$$

Pri čemu je:

- $L(v_k)$ - opterećenje vozila nakon obilaska kupca v_k
- $L(1)$ - teret s kojim vozilo kreće iz skladišta
- $P(1, v_k)$ - kupci posluženi u ruti koja kreće iz skladišta i završava s kupcem v_k, uključujući i kupca v_k
- $C_p(v_k) = \sum_{v_i \in P(1,v_k)} p_i$ - ukupna količina dobara pokupljena tijekom određene rute

– $C_d(v_k) = \sum_{v_i \in P(1,v_k)} d_i$ - ukupna količina dobara predana kupcima tijekom određene rute

– C - kapacitet pojedinog vozila

Da bi rješenje bilo valjano mora vrijediti:

$$L(v_k) \leq C, \forall v_k \in V \tag{2.6}$$

Za zapis VRPPD problema korišten je Li & Lim benchmark, koji kombinira VRPPD s CVRP-om i VRPTW-om i daje definicije problema za primjere s 100, 200, 400, 600, 800 i 1000 kupaca. Svaki kupac predstavlja jedan zadatak, koji može biti preuzimanje ili dostava tereta. U ovim primjerima svaki kupac ima svog para, pri čemu jedan u paru šalje teret, a drugi ga prima.

3. Metaheuristički algoritmi

3.1. O evolucijskim algoritmima

Evolucijski algoritmi (eng. *Evolutionary algorithm (EA)*) podskup su metaheurističkih algoritama optimizacije. Imitiraju proces evolucije i predstavljaju metodu slučajnog i usmjerenog traženja rješenja. Rade s populacijom jedinki, pri čemu svaka jedinka prestavlja jedno rješenje i ima zapis tog rješenja u svom genotipu. Početna se populacija najčešće sastoji od jedinki stvorenih slučajno, uz poštivanje ograničenja danog problema. Također, postoji funkcija dobrote koja svakoj jedinki pridružuje njenu dobrotu (eng. *fitness*), broj koji pokazuje koliko dobro ta jedinka rješava problem, odnosno njenu sposobnost i kvalitetu. Ovisno o problemu može se zahtjevati da dobrota jedinke bude maksimalna ili minimalna.

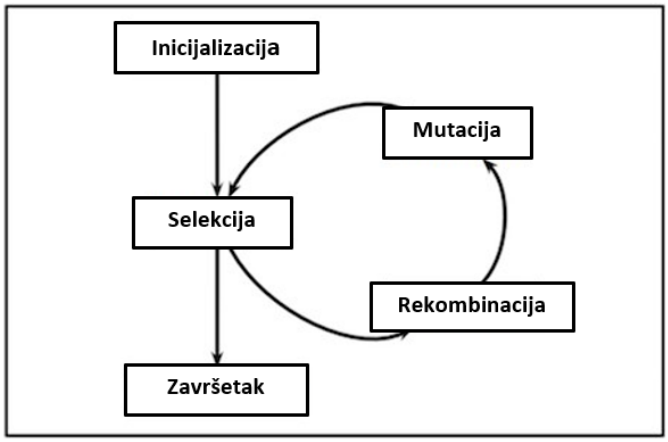
Algoritam zatim provodi evolucijske mehanizme, odabire najbolje jedinke na temelju njihove dobrote (selekcija) i nad njima obavlja operacije rekombinacije i mutacije. Jedinke s najlošijim rezultatom se u sljedećoj generaciji zamjenjuju novostvorenim jedinkama i proces se ponavlja do određenog broja generacija ili do pronalaska jedinke s traženim iznosom dobrote. Koraci evolucijskog algoritma prikazani su na slici 3.1, gdje jedna iteracija petlje predstavlja jednu generaciju.

U ovom radu korištena su dva algoritma, algoritam evolucijske strategije i turnirski eliminacijski genetski algoritam, koji su objašnjeni u nastavku.

3.2. Evolucijska strategija

Evolucijska strategija (eng. *Evolution strategy (ES)*) jedna je od tehnika optimizacije iz područja evolucijskih algoritama. Algoritam se temelji na provođenju operacija mutacije i rekombinacije i zadržavanja određenog broja najboljih jedinki. Selekcija se provodi deterministički i bazira se samo na rangu jedinke na temelju dobrote, a ne na stvarnim vrijednostima dobrote. Zbog toga je algoritam neosjetljiv na monotone promjene funkcije dobrote.

Pojedini algoritam evolucijske strategije formalno se zapisuje kao: $(\mu/\rho +/, \lambda) - ES$,



Slika 3.1: Koraci evolucijskog algoritma

gdje je:

- μ - broj jedinki u generaciji roditelja
- ρ - broj jedinki koje se koriste za stvaranje potomstva (moguće vrijednosti: 1 (mutacija), 2 (rekombinacija))
- λ - broj jedinki stvorenih iz generacije roditelja
- selekcija
  - plus "+" - jedinke roditelja i potomaka se, temeljem svoje dobrote, skupa natječu za poziciju u sljedećoj generaciji
  - zarez "," - samo se jedinke potomaka natječu za prijelaz u sljedeću generaciju

U svakom slučaju, u populaciji se zadrži μ najboljih jedinki, a ako se koristi selekcija označena s "+" uvjet je da vrijedi $\lambda > \mu$.

Pseudokod 3.1 prikazuje pseudokod navedenog algoritma. U radu je korišten algoritam Evolucijske strategije $(1/1 + 4)$, što znači da se od operacija provodi samo mutacija.

Pseudokod 3.1: Pseudokod algoritma evolucijska strategija

```
1 single generation {
2  repeat(for every subpopulation) {
3     add μ individuals to the parent pool;
4     create λ offspring using random ρ parents for each;
5     if comma selection
6         create new parent pool with μ best from offspring pool;
7     else
```

```
        create new parent pool with μ best from offspring and parents pool;
 }
}
```

3.3. Genetski algoritam

Turnirski eliminacijski genetski algoritam (eng. *Steady-State Algorithm with Tournament Selection*) spada u genetske algoritme, koji su podskupina evolucijskih algoritama. Predstavlja eliminacijski algoritam s turnirskom metodom selekcije.

Pojam eliminacijski označava da nema klasičnih generacija u kojima nove jedinke nastanu mutacijom i rekombinacijom starih, već se u pojednoj iteraciji populaciji doda na taj način dobiveno potomstvo. Slučajnim odabirom makne se određen broj jedinki iz početne generacije kako bi veličina populacije bila konstantna. Za micanje moguće je odabrati i samo najlošije jedinke, ali time može doći do smanjenja raznolikosti u populaciji.

Turnirska metoda selekcije jest metoda odabira jedinki za rekombinaciju i uključuje provođenje više "turnira" između k slučajno odabranih jedinki. Jedinka s najboljom dobrotom postaje pobjednik turnira i koristi se za rekombinaciju i/ili prenosi u sljedeću generaciju. Korištena implementacija algoritma zamjenjuje najlošiju jedinku u turniru s onom dobivenom rekombinacijom dviju slučajno odabranih jedinki koje su preostale u turniru. Algoritam je prikazan pseudokodom 3.2, a u radu su korišteni turniri veličine tri jedinke.

Pseudokod 3.2: Pseudokod turnirskog eliminacijskog genetskog algoritma

```
single generation {
 repeat(deme size times) {
  randomly add n individuals to the tournament;
  select the worst one in the tournament;
  randomly select two parents from remaining ones in the tournament;
  replace the worst with crossover child;
  perform mutation on child;
 }
}
```

4. Implementacija

4.1. Oblici zapisa rješenja

Kako bi se nad problemom usmjeravanja vozila mogli provoditi algoritmi optimizacije i različiti operatori nužno je prvo odrediti oblik zapisa jednog rješenja. U ovom radu korištena su tri načina zapisa rješenja, nad sva tri primjenjena je optimizacija i uspoređena je kvaliteta dobivenih rješenja. Budući da su korišteni evolucijski algoritmi pojedino rješenje zapravo predstavlja genom jedinke.

Za svaki problem unaprijed je zadan broj vozila N_v, kapacitet pojedinog vozila C, broj kupaca N_c i za svakog kupca njegove koordinate, narudžba, vremenski okvir u kojemu je dostupan i trajanje samog istovara, te za VRPPD dodatno i kojem kupcu se šalje ili od kojeg se kupca očekuje roba. U svakom rješenju vozila i kupci prikazuju se pomoću cijelih brojeva, na način da brojevi u intervalu $[1, N_v]$ predstavljaju oznake kupaca, a interval $[N_c + 1, N_c + N_v]$ oznake vozila. Svaki oblik ukratko je prikazan primjerom s 15 kupaca i 5 vozila. U tom slučaju oznake kupaca su $[1, 15]$, a oznake vozila $[16, 20]$.

Prvi oblik zapisa je prikaz rješenja pomoću vektora permutacije. Genom je u tom slučaju vektor cijelih brojeva, gdje se svaki broj u intervalu $[1, N_v + N_c]$ pojavljuje točno jednom. Vektor započinje oznakom vozila i nakon toga slijede slučajnim redoslijedom oznake kupaca i ostalih vozila. Kupci čija se oznaka nalazi u vektoru desno od oznake prvog vozila, a prije oznake drugog vozila, pripadaju prvom vozilu i obilazi ih se redoslijedom kojim su navedeni u vektoru. Na slici 4.1 dan je primjer takvog zapisa, gdje vozilo 16 obilazi kupce 3, 6, 11, 2, 9 i 13, vozilo 18 kupce 14 i 1, vozilo 17 kupce 5, 10, 4, 12 i vozilo 20 kupce 7, 8 i 15, a vozilo 19 se ne koristi.

Slika 4.1: Primjer zapisa rješenja pomoću vektora permutacije

Sljedeći oblik rješenja jest zapis genoma kao dvije liste cijelih brojeva. Prva lista sadrži oznake kupaca, a druga oznake vozila i obje su duljine broja kupaca N_c. U listi kupaca oznaka svakog kupca pojavljuje se točno jednom, a u listi vozila se oznaka pojedinog vozila pojavljuje slučajan broj puta (od 0 do N_c puta). Kupac koji se u listi kupaca nalazi na mjestu i pripada onom vozilu koje se nalazi na i-tom mjestu u listi vozila i vozilo obilazi kupce koji su mu dodijeljeni onim redom kojim se oni pojavljuju u listi kupaca. Na slici 4.2 zapisan je prethodni primjer u obliku dvije liste.

3	6	14	11	5	1	10	4	7	2	8	15	9	12	13
16	16	18	16	17	18	17	17	20	16	20	20	16	17	16

Slika 4.2: Primjer zapisa rješenja u obliku dvije liste

Zadnji implementirani oblik rješenja je prikaz genoma pomoću mape. Ključ mape jest oznaka vozila, a vrijednost mape vektor cijelih brojeva koji sadrži oznake onih kupaca koji pripadaju tom vozilu. Vozilo kupce obilazi redom kojim su navedeni u pripadnom vektoru. Prikaz takvog zapisa rješenja danog primjera dan je na slici 4.3.

Ključ	Vrijednost
16	3, 6, 11, 2, 9, 13
17	5, 10, 4, 12
18	14, 1
19	
20	7, 8, 15

Slika 4.3: Primjer zapisa rješenja pomoću mape

4.2. Korišteni operatori

U radu su implementirani po jedan operator mutacije i rekombinacije za svaki oblik zapisa genoma, operator evaluacije te dva lokalna operatora, operator lokalne pretrage i operator lokalnog spajanja.

Operator mutacije radi jednu od dvije moguće zamjene s jednakom vjerojatnošću. Jedna je mogućnost zamjena mjesta dvaju kupaca unutar istog slučajno odabranog vozila, a druga je zamjena dva kupca iz različitih vozila. Primjer mutacije dan je na slici 4.4 za oblik zapisa rješenja pomoću mape.

Ključ	Vrijednost
16	3, 6, 2, 11, 9, 13
17	5, 10, 4, 12
18	14, 1
19	
20	7, 8, 15

Ključ	Vrijednost
16	3, 6, 11, 2, 9, 13
17	5, 8, 4, 12
18	14, 1
19	
20	7, 10, 15

Slika 4.4: Prikaz mutacije unutar istog vozila (lijevo) i između različitih vozila (desno)

Operacija rekombinacije ostvarena je tako da se slučajno odaberu dva broja, indeks početka (i_1) i indeks kraja (i_2), pri čemu je indeks kraja strogo veći od indeksa početka. Dio genoma prvog roditelja od i_1 do i_2 postaje početak genoma djeteta. Zatim se na taj genom dodaju sve oznake vozila i kupaca koji nedostaju i to onim redom kojim se pojavljuju u genomu drugog roditelja. U slučaju zapisa vektorom permutacije na kraju se još osigura da je na prvom mjestu oznaka vozila tako što se, ako se na prvom mjestu nalazi kupac, on zamijeni s prvom oznakom vozila koja se pojavljuje u vektoru. Slika 4.5 prikazuje primjer rekombinacije na obliku zapisa genoma vektorom permutacije.

Prvi roditelj	16 3 6 11 2 9 13 18 14 1 19 17 5 10 4 12 20 7 8 15
Drugi roditelj	18 5 2 14 1 16 3 20 15 11 12 8 19 7 4 13 17 6 9 10
Dijete	18 14 1 19 17 5 10 4 2 16 3 20 15 11 12 8 7 13 6 9

Slika 4.5: Prikaze rekombinacije za slučaj $i_1 = 7$, $i_2 = 14$

Operator evaluacije pretvara sva tri načina zapisa u isti oblik i odabrano je da je to oblik vektora permutacije. Nad dobivenim oblikom zatim poziva funkciju za određivanje dobrote jedinke. Za svaku jedinku pohranjuje se njezina dobrota i rješenje koje ona predstavlja je to bolje što je iznos dobivene dobrote manji. Tijekom generacija teži se minimizaciji dobrote jedinki i pronalasku rješenja s minimalnim iznosom dobrote.

4.3. Operator lokalne pretrage

Operator lokalne pretrage je operator koji pokušava iterativnim pozivanjem odgovarajuće operacije mutacije poboljšati početno rješenje. Ovisi o trima definiranima parametrima: frekvenciji (eng. *frequency* - f), budžetu (eng. *budget* - b) i broju pokušaja (eng. *attempts* - a). Poziva se svakih f generacija i tada nad najboljom jedinkom iterativno poziva operaciju mutacije. Prilikom svakog poboljšanja dobrote jedinke, sprema

mutiranu jedinku na mjesto stare. Staje kada je izvršio mutaciju b puta ili kada je iskoristio sve pokušaje, odnosno kada a puta za redom jedinka nakon mutacije ima lošije rezultate nego prije mutacije. U radu je testirana efikasnost tog operatora usporedbom rješenja dobivenih sa i bez njegova korištenja.

4.4. Operator lokalnog spajanja

Drugi korišteni lokalni operator jest operator lokalnog spajanja ruta. Kada je pozvan, spaja dvije rute s najmanjim brojem vozila u jednu tako da kupce iz najkraće rute prebaci na kraj druge po redu najkraće rute. Ovisi samo o parametru frekvenciji f kojom se definira koliko će se često operator pozivati. Ako jedinka dobivena nakon spajanja ruta ima bolje rezultate nego prije sprema se na mjesto stare jedinke, a u protivnom se promjene ruta odbacuju. Provedene su usporedbe dobivenih rezultata prije korištenja operatora i s njegovim korištenjem.

4.5. Funkcija dobrote

Kao što je već rečeno, implementirana je jedna funkcija dobrote (eng. *Fitness function*) koja računa dobrotu rješenja zapisanog u obliku vektora permutacije.

Ovisno o vrsti VRP problema koji se rješava, funkcija dobrote uzima u obzir različite parametre. U svakom slučaju dobrota ovisi o prijeđenoj udaljenosti, a da bi rješenje bilo prihvatljivo udaljenost bi trebala predstavljati cjelokupni iznos dobrote. Za slučaj CVRP-a tu se dodaje i kazna za prekoračenje kapaciteta, koja iznosi prekoračenje uvećano tisuću puta. Pri računanju dobrote VRPTW problema, uz prekoračenje kapaciteta, dodaje se i suma ukupnog čekanja te kazna za dolazak iza vremenskog okvira kupca, koja množi trenutnu dobrotu s ukupnim brojem vozila. Za VRPPD problem vrijede sve do sad navedene kazne te se očuvanje kapaciteta provjerava na svakom koraku rute. Pseudokod 4.1 prikazuje računanje funkcije dobrote.

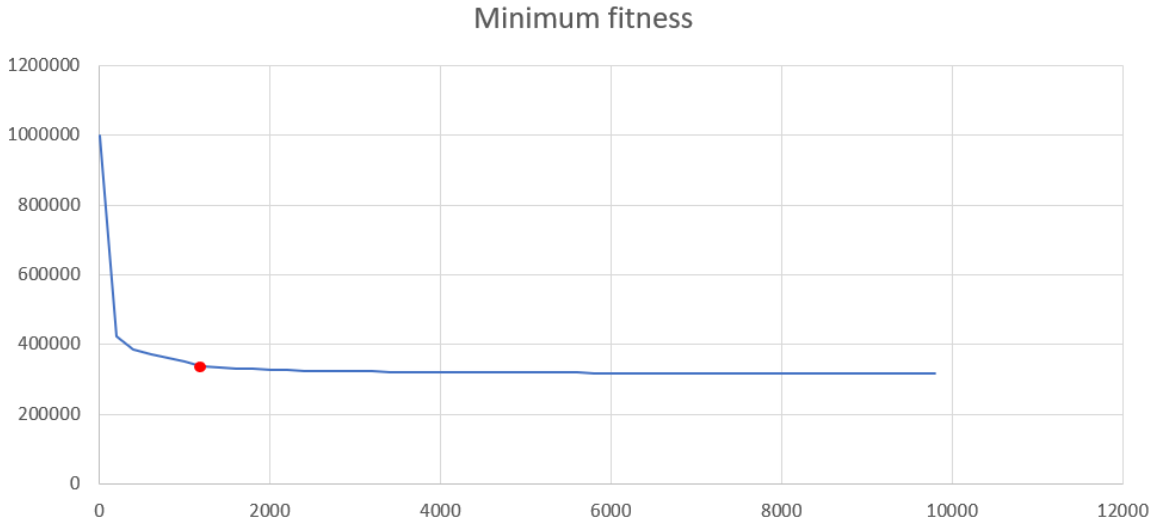
Pseudokod 4.1: Pseudokod funkcije dobrote

```
1 if no time windows
2         fitness = totalDistance + 1000 * overCapacity;
3 else
4         fitness = totalDistance + 1000 * overCapacity + overTime;
5         if  was late
6                 fitness *= numberOfVehicles;
```

5. Rezultati

5.1. Kriterij zaustavljanja

U ovom poglavlju prikazane su razne usporedbe kvaliteta rješenja. Kako bi takve usporedbe bile moguće, nužno je odrediti kriterij zaustavljanja programa odnosno do kad će program pokušavati optimizirati rješenje. Uzeto je da je taj kriterij broj evaluacija jedinki potreban da funkcija minimalne dobrote počne konvergirati. Za svaku instancu problema program je pokrenut 10 puta i zaustavljen je tek nakon milijun evaluacija. Zatim je za svaku generaciju izračunat prosjek minimalnog iznosa dobrote u toj generaciji. Na slici 5.1 prikazano je određivanje kriterija zaustavljanja na primjeru VRPTW problema s 100 kupaca koristeći zapis rješenja u obliku vektora permutacije, turnirski genetski algoritam i 100 jedinki po populaciji. Na x osi je broj generacija, a na y osi prosjek minimalne dobrote te generacije. Za ovaj primjer kao kriterij zaustavljanja uzima se 1200 generacija odnosno 120 000 iteracija.



Slika 5.1: Određivanje kriterija zaustavljanja za VRPTW za oblik permutacije

5.2. Usporedba različitih načina zapisa rješenja

U ovom potpoglavlju provedena je usporedba efikasnosti tri navedena načina zapisa rješenja (vektora permutacije, liste i mape) za svaki oblik VRP problema. Budući da nije uspoređivana brzina pronalaženja najboljeg rješenja, već samo kvaliteta tog rješenja, svaki oblik zapisa imao je svoj kriterij zaustavljanja odnosno imao je onoliko iteracija koliko mu je potrebno do konvergencije minimalnog iznosa dobrote. Za sva tri zapisa korišten je turnirski genetski algoritam i veličina populacije od 100 jedinki, a program je pokretan 20 puta.

Na slici 5.2 prikazana je uspješnost svih zapisa pri rješavanju CVRP, VRPTW i VRPPD problema. Iz grafova se vidi da mapa predstavlja najlošiji način zapisa rješenja i vidno zaostaje za permutacijom i listom, dok od ta dva zapisa permutacija daje nešto bolje rezultate.

5.3. Usporedba algoritama

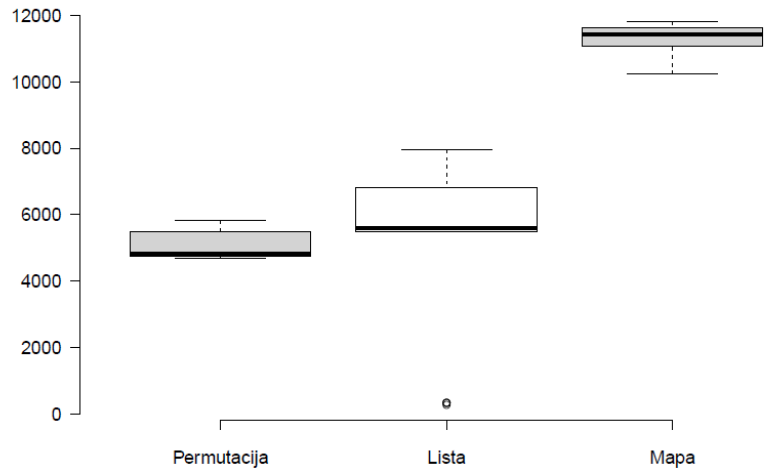
Uspoređeni su i navedeni algoritmi kako bi se vidjelo koji od njih dolazi do boljih rješenja. Kao zajednički kriterij zaustavljanja oba algoritma uzet je veći od brojeva potrebnih evaluacija pojedinog algoritma. Na svakom grafu y os predstavlja iznos dobrote, a broj jedinki u populaciji je 100. Za svaki primjer program je pokretan 20 puta.

Za CVRP problem za 25 kupaca usporedba algoritama prikazana je na slici 5.3. Za zapis u obliku permutacije turnirski algoritam dolazi do boljih rješenja, međutim u jednom od pokretanja evolucijska strategija dobiva rješenje bolje od svih dobivenih od genetskog algoritma. U slučaju liste evolucijska strategija u prosjeku dolazi do boljih rješenja, ali ima i mnogo veća i češća odstupanja od njih.

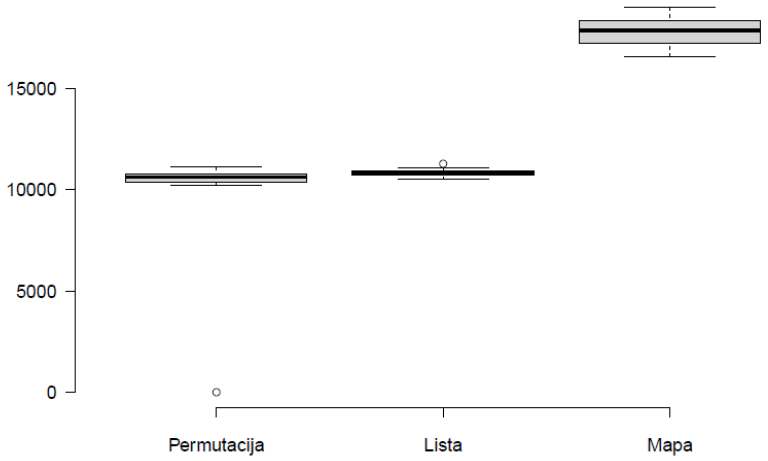
U slučaju VRPTW problema s 25 kupaca turnirski genetski algoritam nadmašuje evolucijsku strategiju u slučaju zapisa listom, međutim pri korištenju zapisa rješenja u obliku mape algoritam evolucijske strategije dolazi do puno boljih rezultata. Grafički prikaz dan je na slici 5.4.

Usporedba za VRPPD problem sa 100 kupaca prikazana je na slici 5.5. Vidi se da u slučaju zapisa vektorom permutacije bolje rezultate ostvaruje turnirski genetski algoritam, dok je u slučaju zapisa mapom to algoritam evolucijske strategije.

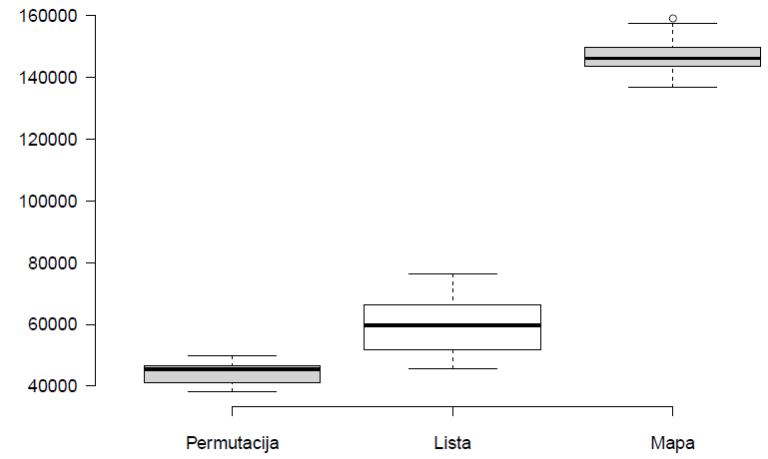
Iz prikazanih usporedbi može se zaključiti da zapisi rješenja u obliku vektora permutacije i liste dolaze do boljih rezultata koristeći turnirski genetski algoritam, a zapis



(a) CVRP problem s 25 kupaca

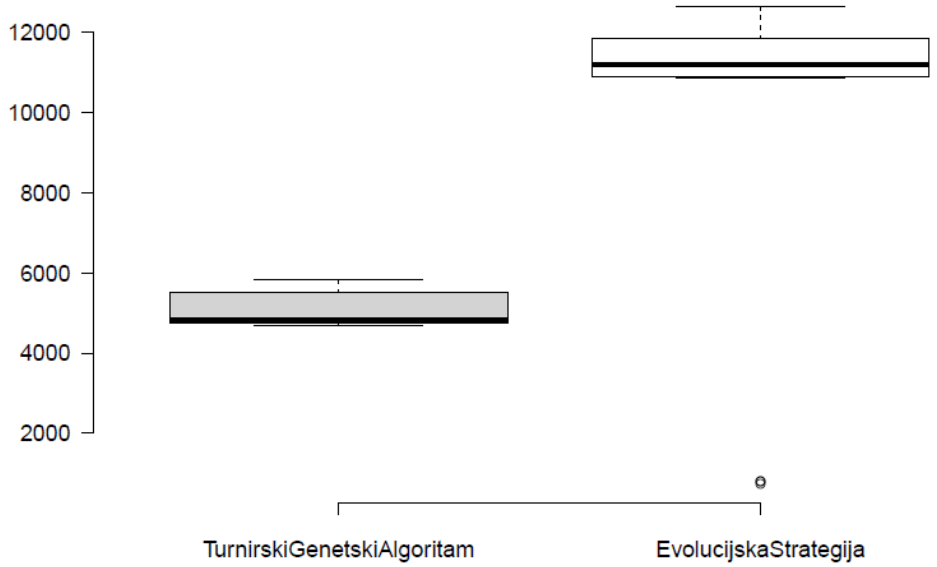


(b) VRPTW problem s 25 kupaca

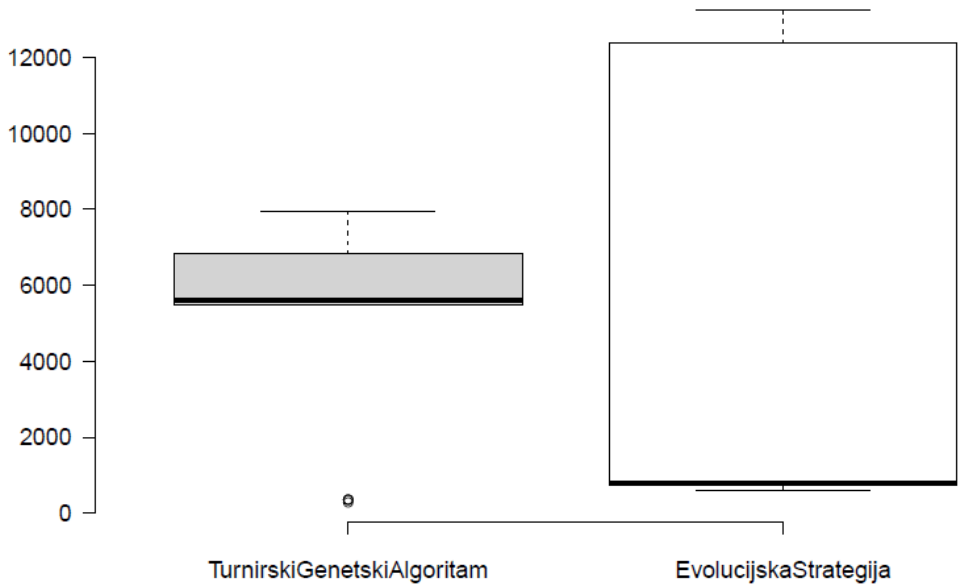


(c) VRPPD problem sa 100 kupaca

Slika 5.2: Usporedba efikasnosti različitih načina zapisa

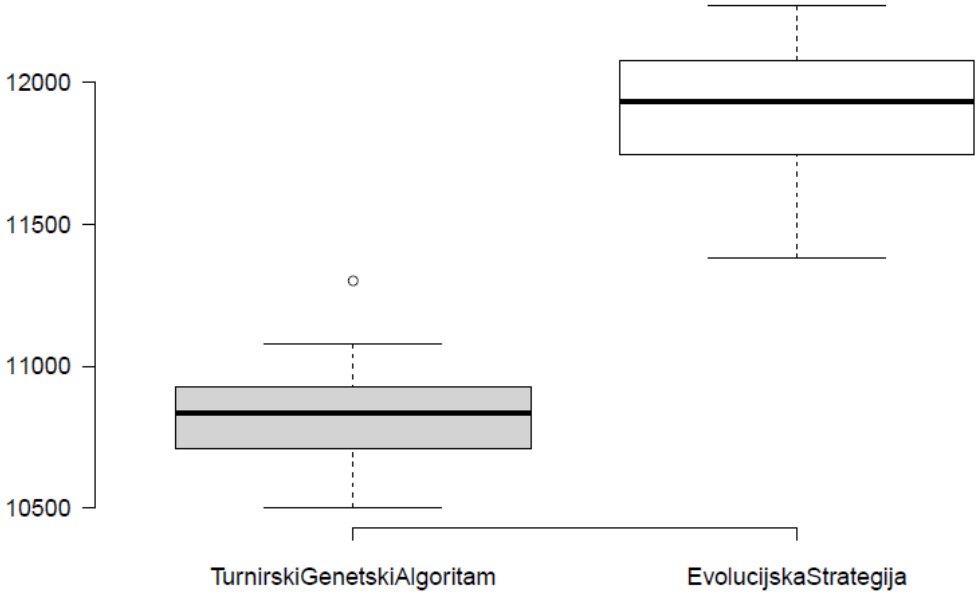


(a) Zapis u obliku permutacije

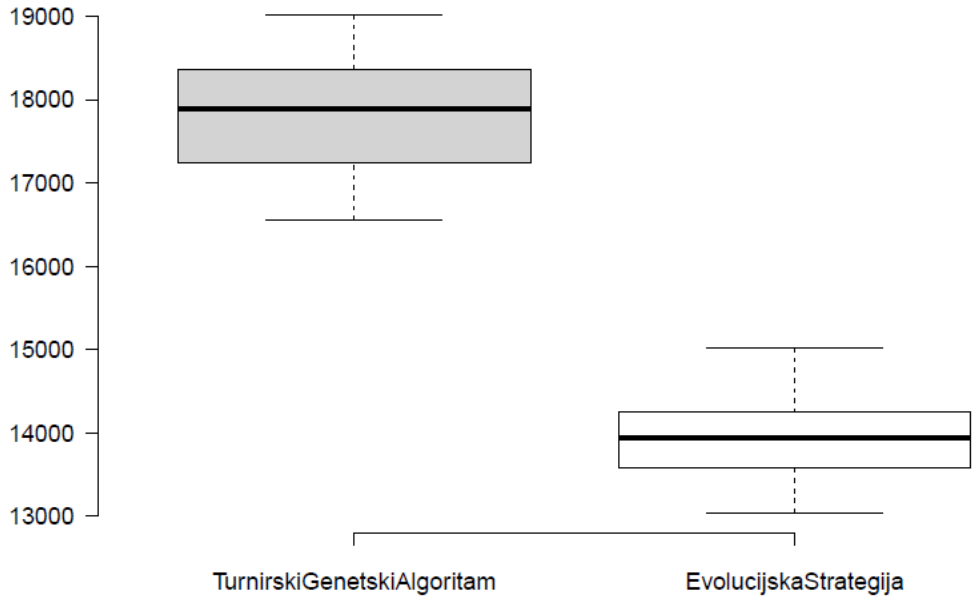


(b) Zapis u obliku liste

Slika 5.3: Usporedba algoritama za CVRP problem s 25 kupaca

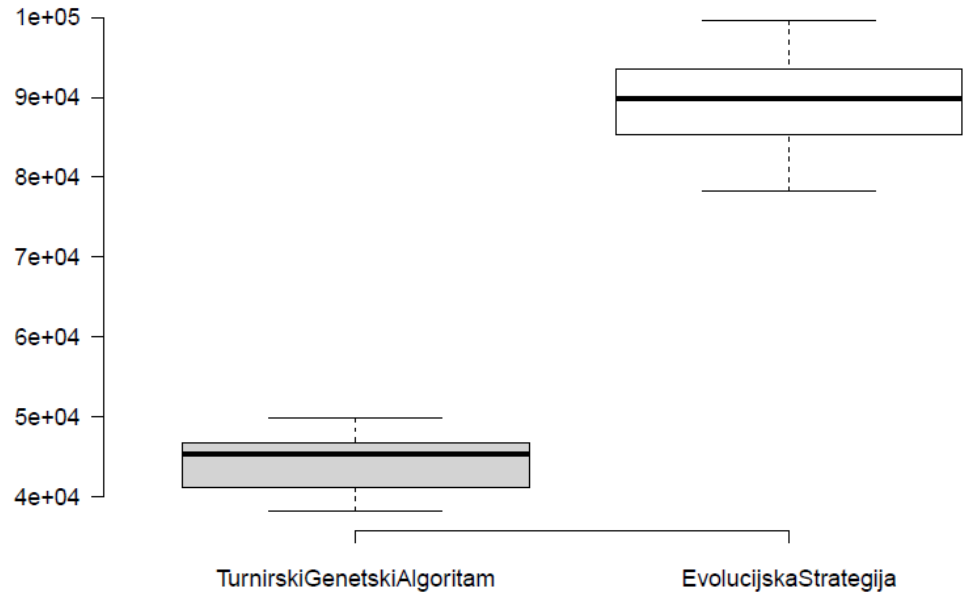


(a) Zapis u obliku liste

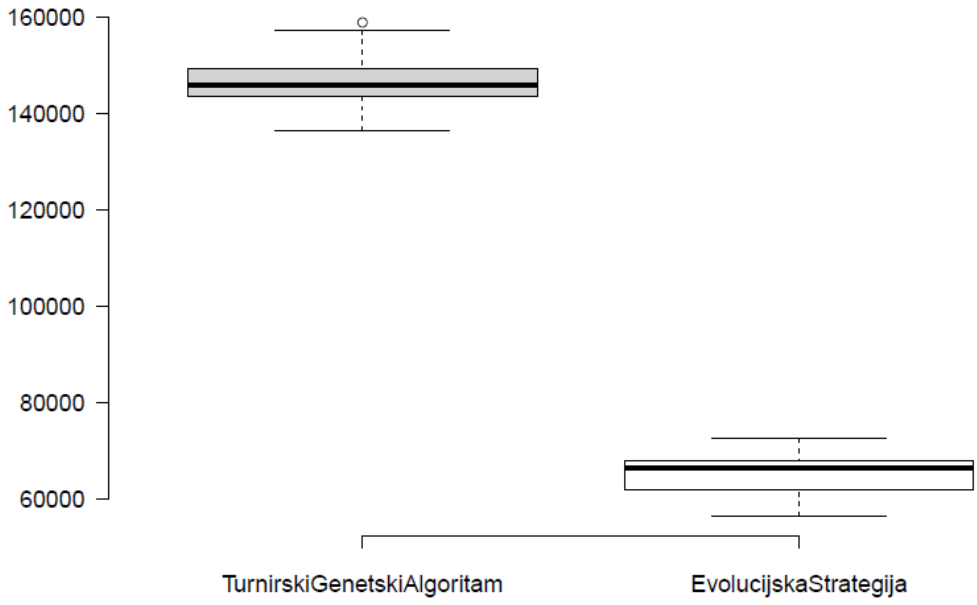


(b) Zapis u obliku mape

Slika 5.4: Usporedba algoritama za VRPTW problem s 25 kupaca



(a) Zapis u obliku permutacije



(b) Zapis u obliku mape

Slika 5.5: Usporedba algoritama za VRPPD problem sa 100 kupaca

u obliku mape koristeći algoritam evolucijske strategije. Najbolja rješenja ostvarila su se kombinacijom zapisa u obliku permutacije i genetskog algoritma.

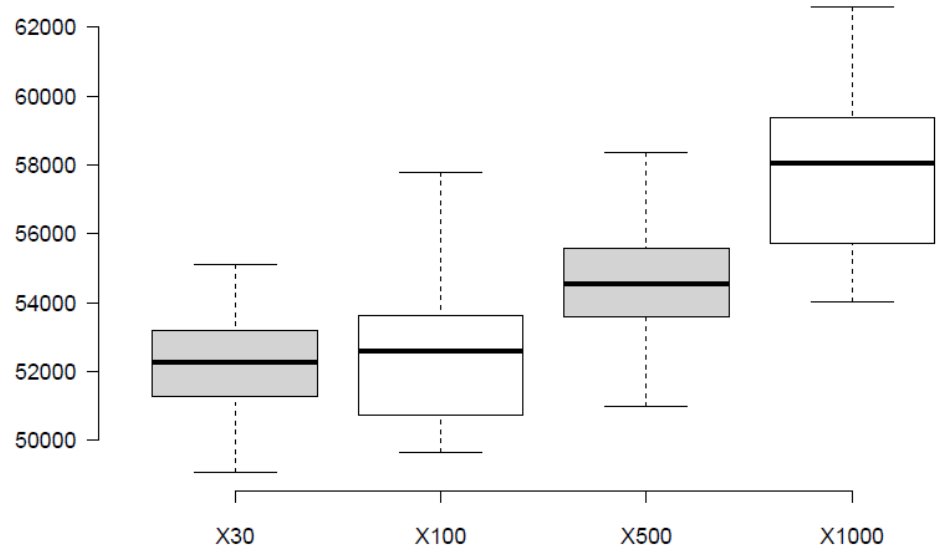
5.4. Utjecaj veličine populacije

Kako bi se pokazao utjecaj veličine populacija provedena je usporedba rezultata populacija različitih veličina. Broj evaluacija, vjerojatnost mutacije i ostali parametri isti su za svaku veličinu populacije. Dobiveni su rezultati za 30, 100, 500 i 1000 jedinki po populaciji. Na svakom grafu x os predstavlja broj jedinki, a y os iznos minimalnih dobrota. Svi rezultati dobiveni su iz 20 pokretanja programa.

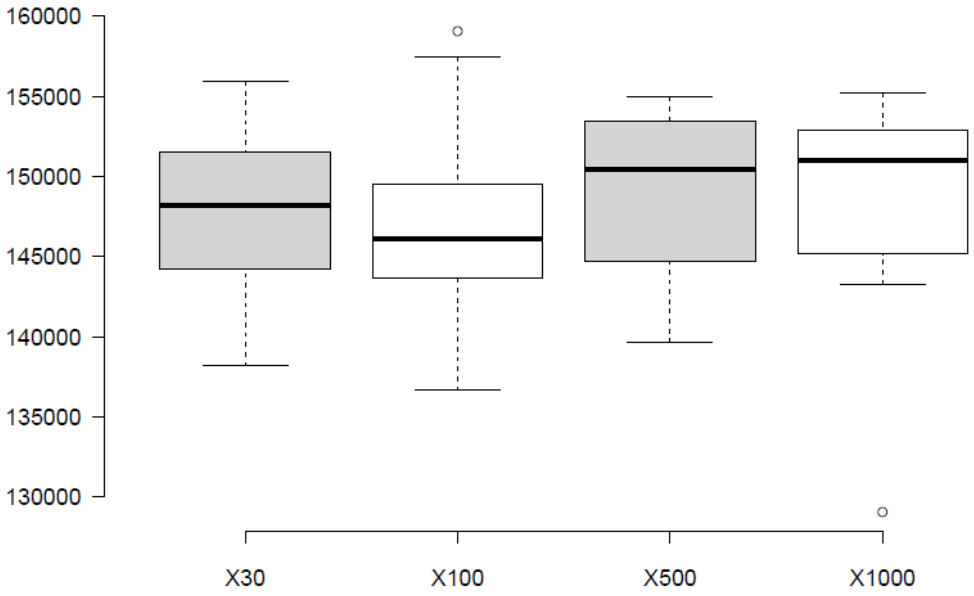
Na slici 5.6 dani su prikazi ovisnosti minimalne dobrote o veličini populacije za VRPTW problem i zapis liste te za VRPPD problem i zapis mape. Oboje je dobiveno primjenom turnirskog genetskog algoritma. Može se vidjeti da s porastom veličine populacije kvaliteta rješenja opada i da se dobri rezultati mogu dobiti za populaciju do veličine 100 jedinki. Gotovo identični rezultati dobiveni su i za ostale primjere, pri čemu za zapise oblika permutacije i liste prosječna minimalna dobrota raste brže s povećanjem populacije, a za zapise oblika mape nešto sporije.

Slična situacija dobiva se koristeći algoritam evolucijske strategije kao što je prikazano na slici 5.7. U ovom slučaju dobrota porastom veličine populacije opada još brže, gotovo linearno.

Budući da je kriterij zaustavljanja broj evaluacija, primjeri s većim brojem jedinki u populaciji imaju manje generacija od onih s manje jedinki. Na temelju toga se iz dobivenih rezultata može zaključiti kako je za efikasnost rješenja bitnije imati više generacija nego više jedinki u populaciji. Time se vidi važnost operatora mutacije i rekombinacije koji omogućavaju to poboljšanje kroz generacije.

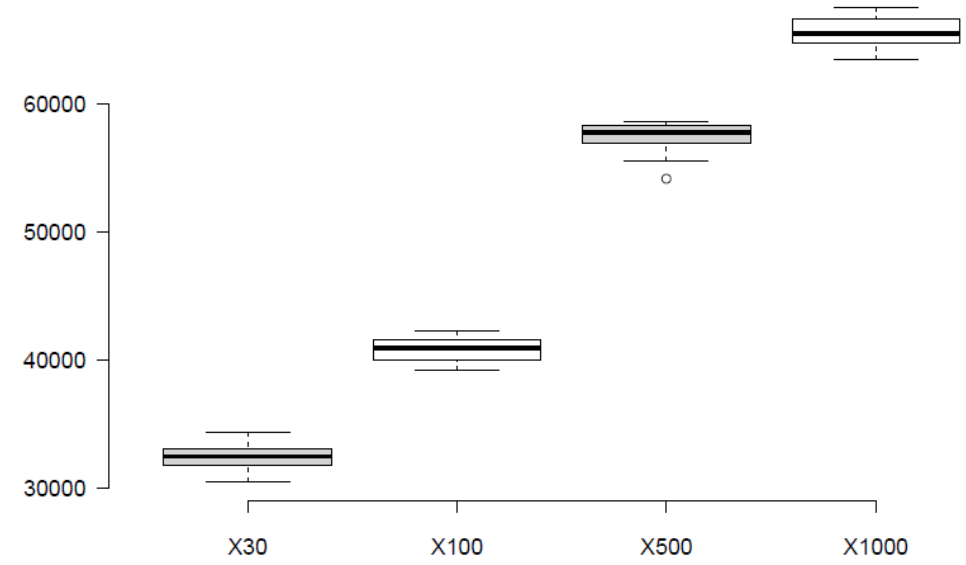


(a) VRPTW problem s 50 kupaca i rješenjem zapisanim u obliku liste

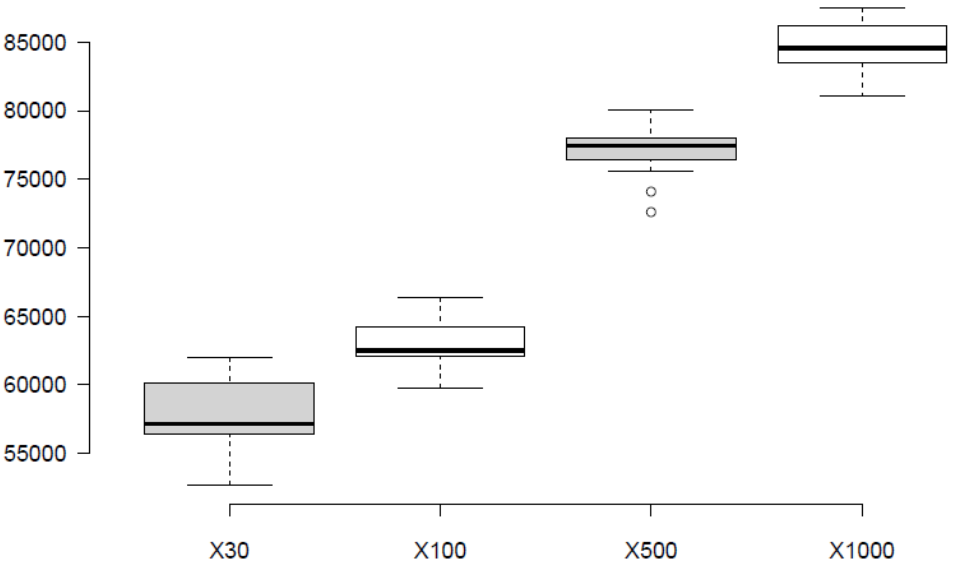


(b) VRPPD problem sa 100 kupaca i rješenjem zapisanim u obliku mape

Slika 5.6: Usporedba uspješnosti populacija različitih veličina (turnirski genetski algoritam)



(a) CVRP problem s 50 kupaca i rješenjem zapisanim u obliku liste



(b) VRPTW problem s 50 kupaca i rješenjem zapisanim u obliku permutacije

Slika 5.7: Usporedba uspješnosti populacija različitih veličina (evolucijska strategija)

5.5. Testiranje operatora lokalne pretrage

Kao što je već spomenuto, operator lokalne pretrage poziva uzastopno određen broj operacija mutacije i na taj način pokušava poboljšati dano rješenje. Ima tri parametra: frekvenciju pozivanja, budžet i dozvoljeni broj pokušaja, te se uvijek poziva nad najboljom jedinkom u generaciji.

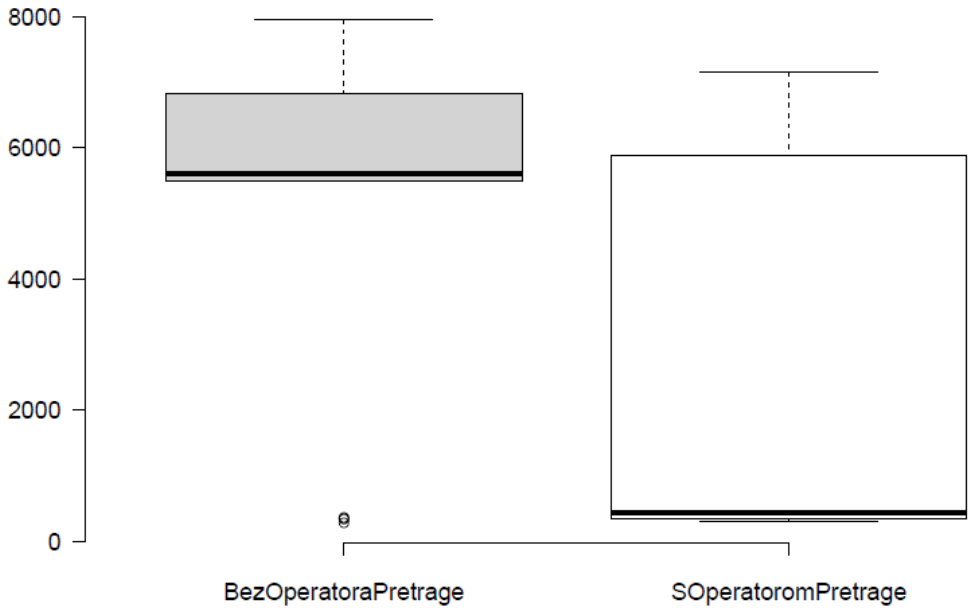
Provedena je usporedba kvalitete rješenja sa i bez korištenja lokalne pretrage. U tablici 5.1 prikazani su rezultati dobiveni za CVRP problem s 25 kupaca, VRPTW problem s također 25 kupaca te VRPPD problem sa 100 kupaca, koristeći svaki oblik zapisa. Treći i četvrti stupac tablice predstavljaju prosječan minimalni iznos dobrote u 20 pokretanja. Za svaku instancu problema broj jedinki u populaciji je 100 i korišten je turnirski genetski algoritam, a za testiranje operatora korišten je isti broj evaluacija kao i bez njega. Parametri operatora lokalne pretrage su: $f = 3$, $b = 10$ i $a = 3$, odnosno poziva se svake tri generacije, može provesti mutaciju maksimalno 10 puta te smije dobiti lošiji rezultat najviše tri puta za redom. Za svaki od slučajeva program je pokrenut 20 puta.

Tablica 5.1: Usporedba rješenja s korištenjem operatora lokalne pretrage i bez njega.

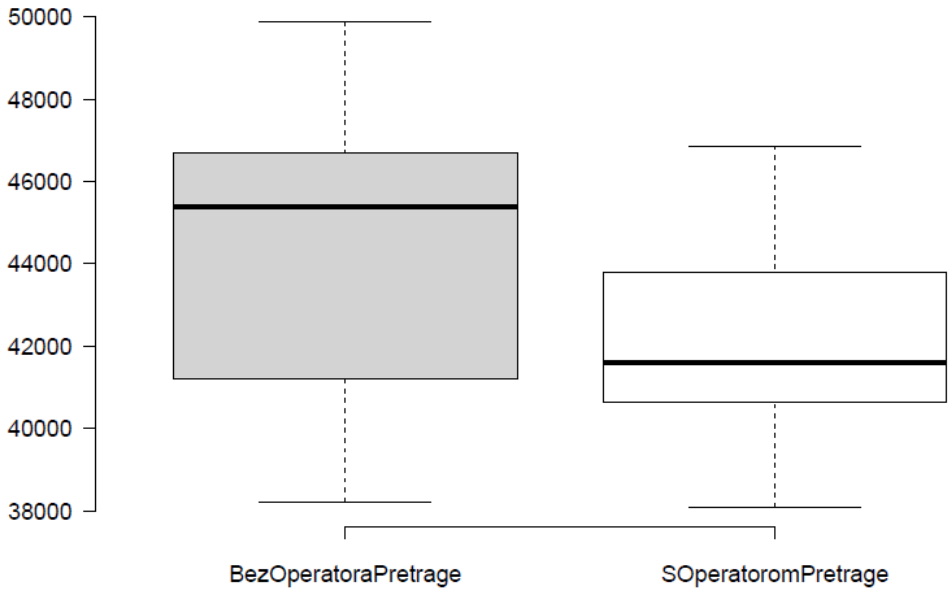
Problem	**Oblik zapisa**	**Bez lokalne pretrage**	**S lokalnom pretragom**
CVRP	permutacija	5106.62	4799.94
CVRP	lista	5119.50	2438.08
CVRP	mapa	11318.54	10924.13
VRPTW	permutacija	10649.86	10508.54
VRPTW	lista	10833.55	10542.69
VRPTW	mapa	17797.79	17738.19
VRPPD	permutacija	44610.54	41970.56
VRPPD	lista	59441.17	59370.45
VRPPD	mapa	147063.65	146528.60

Iz tablice vidimo da program koji koristi operator lokalne pretrage u prosjeku dolazi do boljeg rješenja nego onaj koji ga ne koristi, pri čemu imaju jednak broj dopuštenih evaluacija. U nekim slučajevima, poput CVRP problema sa zapisom u obliku liste, korištenje lokalne pretrage značajno poboljšava rezultate.

Na slici 5.8 prikazana je grafička usporedba uspješnosti nekih rješenja sa i bez korištenja operatora lokalne pretrage.

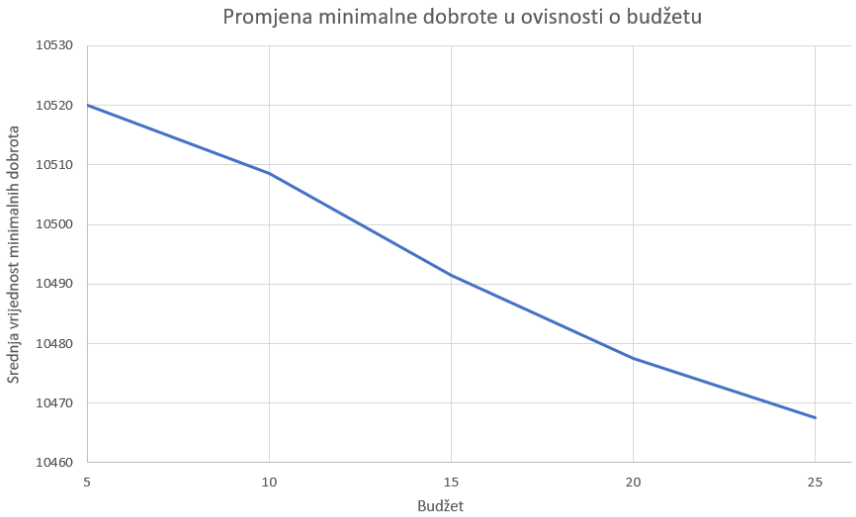


(a) CVRP problem s 50 kupaca i rješenjem zapisanim u obliku liste

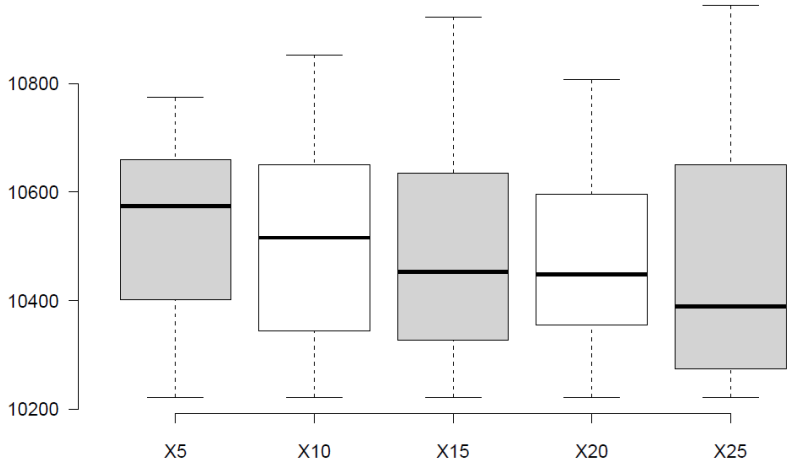


(b) VRPPD problem sa 100 kupaca i rješenjem zapisanim u obliku permutacije

Slika 5.8: Usporedba rezultata s korištenjem operatora lokalne pretrage i bez njega



(a) Graf ovisnosti srednjih vrijednosti minimalnih dobrota o budžetu



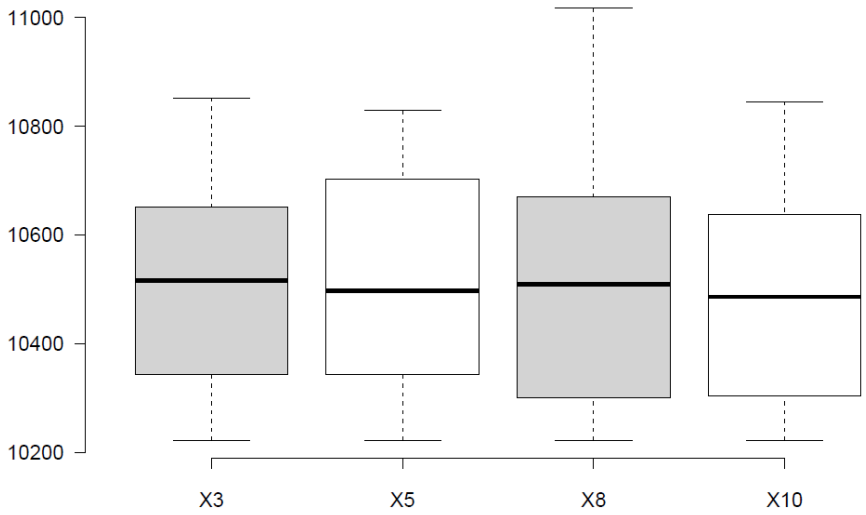
(b) Pravokutni dijagram ovisnosti dobrote o budžetu

Slika 5.9: Prikaz ovisnosti kvalitete rješenja o budžetu

Također provedena je i usporedba kvalitete rješenja u ovisnosti o veličini parametra budžet. Testiranje je provedeno na VRPTW problemu s 25 kupaca i 100 jedinki po populaciji korištenjem genetskog turnirskog algoritma i zapisa rješenja u obliku vektora permutacije. Ostali su parametri nepromijenjeni, operator se pozivao svake tri generacije s maksimalno tri neuspješne mutacije za redom. Dobiveni rezultati prikazani su na slici 5.9. Iz grafova je vidljivo kako se povećanjem dopuštenog broja pozivanja operacije mutacije dobivaju sve bolji i bolji rezultati, odnosno dolazi do izražaja efikasnost operatora lokalne pretrage.

Za razliku od ovisnosti o budžetu, mijenjanje broja dopuštenih neuspješnih mutacija za redom ne utječe toliko na kvalitetu rješenja. Na slici 5.10 dan je prikaz rezultata

za dozvoljenih 3, 5, 8 i 10 pokušaja, pri čemu je budžet bio 10, a ostali parametri nepromijenjeni. Iz dobivenih rezultata može se zaključiti kako će program nakon pogoršanja rješenja mutacijom uglavnom nastaviti u lošem smjeru te, koliko god imao dozvoljenih pokušaja, neće uspjeti doći do poboljšanja.



Slika 5.10: Prikaz ovisnosti kvalitete rješenja o broju dozvoljenih krivih pokušaja

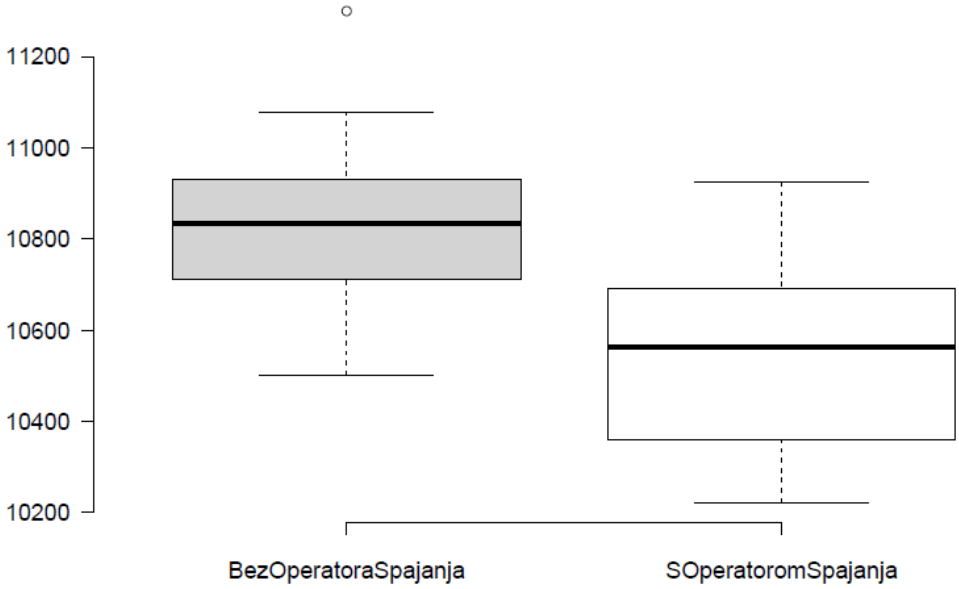
5.6. Testiranje operatora lokalnog spajanja

Operator lokalnog spajanja spaja dvije rute s najmanje kupaca u jednu te je u ovom potpoglavlju ispitivana efikasnost te operacije.

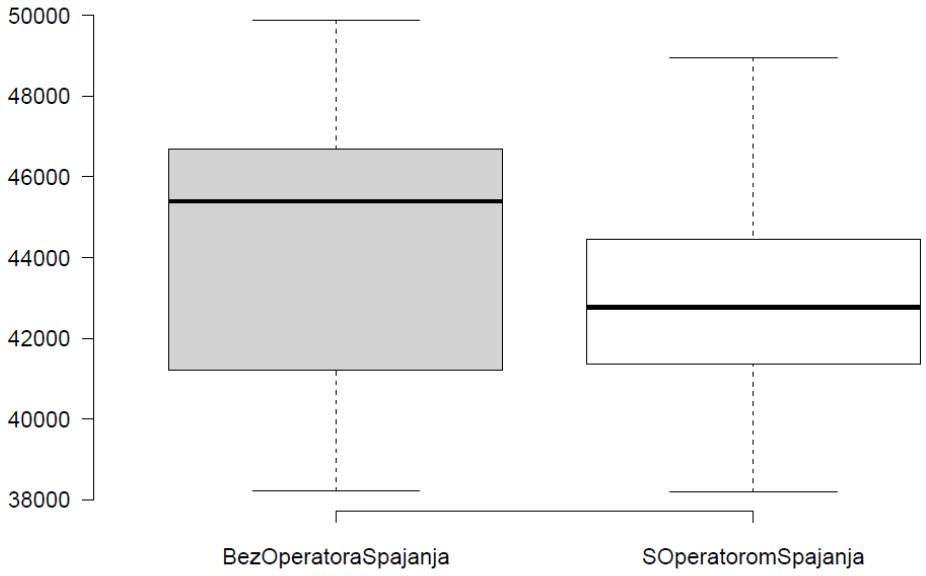
U tablici 5.2 prikazani su rezultati dobiveni za CVRP problem s 25 kupaca, VRPTW problem s također 25 kupaca te VRPPD problem sa 100 kupaca, koristeći svaki oblik zapisa. Kao i u prethodnoj tablici, treći i četvrti stupac su prosječni minimalni iznosi dobrote u 20 pokretanja, broj jedinki u populaciji je 100 i korišten je turnirski genetski algoritam, te isti broj evaluacija za pokretanje sa i bez operatora. Parametar operatora lokalnog spajanja jest $f = 10$, odnosno pozivan je svakih 10 generacija.

Vidi se da u nekim slučajevima operator lokalnog spajanja poboljšava prosječne rezultate, no u nekima nema poboljšanja ili je ono zanemarivo malo. Iz tog možemo zaključiti da je efikasnost operatora lokalnog spajanja ipak manja od one operatora lokalnog pretraživanja.

Na slici 5.11 prikazana su neka od rješenja dobivena korištenjem operatora lokalnog spajanja ruta i bez njega.

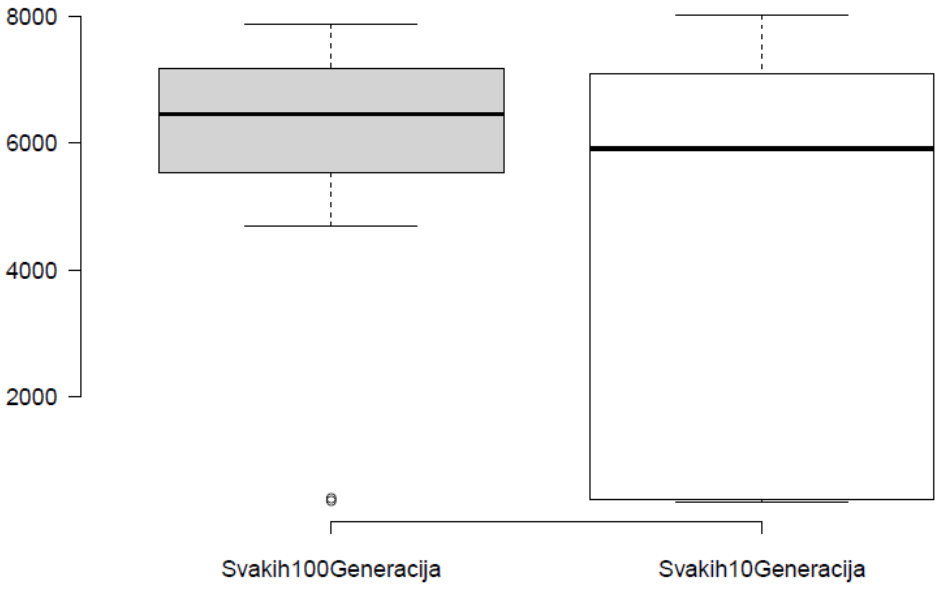


(a) VRPTW problem s 25 kupaca i rješenjem zapisanom u obliku liste

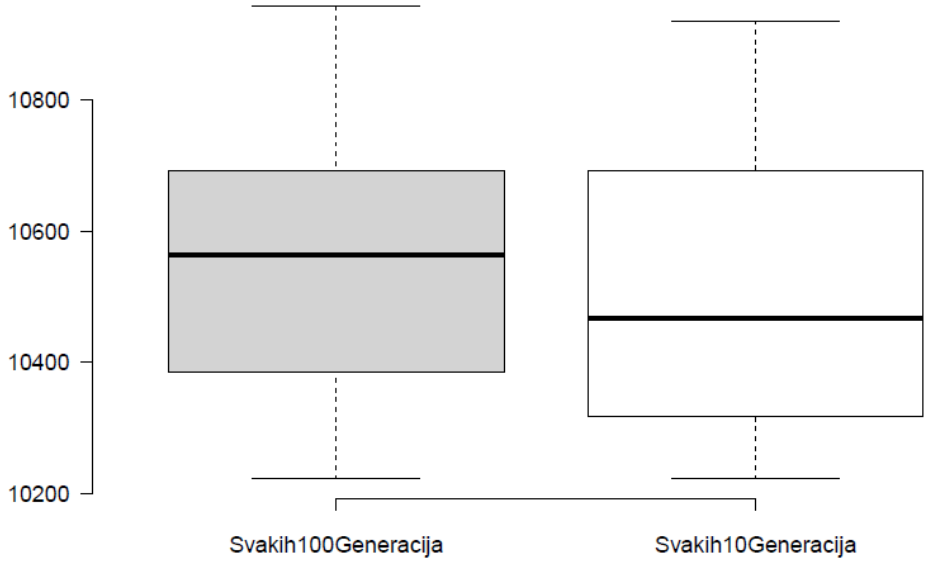


(b) VRPPD problem sa 100 kupaca i rješenjem zapisanim u obliku permutacije

Slika 5.11: Usporedba rezultata s korištenjem operatora lokalnog spajanja i bez njega



(a) CVRP problem s 25 kupaca i rješenjem zapisanom u obliku liste



(b) VRPTW problem s 25 kupaca i rješenjem zapisanom u obliku liste

Slika 5.12: Prikaz ovisnosti kvalitete rješenja o učestalosti pozivanja operatora

Tablica 5.2: Usporedba rješenja s korištenjem operatora lokalnog spajanja i bez njega.

Problem	Oblik zapisa	Bez lokalnog spajanja	S lokalnim spajanjem
CVRP	permutacija	5106.62	5196.58
CVRP	lista	5119.50	4192.40
CVRP	mapa	11318.54	11288.35
VRPTW	permutacija	10649.86	10531.76
VRPTW	lista	10833.55	10548.49
VRPTW	mapa	17797.79	18056.62
VRPPD	permutacija	44610.54	43077.13
VRPPD	lista	59441.17	60246.71
VRPPD	mapa	147063.65	147371.10

Provedena je i usporedba kvalitete rješenja ovisno o frekvenciji, odnosno učestalosti poziva operatora lokalnog spajanja. Na slici 5.12 prikazani su rezultati dobiveni za CVRP problem i VRPTW problem s 25 kupaca i zapis u obliku liste, ako se operator pozivao svakih 100 te svakih 10 generacija. Iz grafova vidimo da se ipak nešto bolji rezultati dobiju češćim pozivanjem operatora lokalnog spajanja ruta, ukoliko se koristi primjer za koji se dobije poboljšanje uporabom tog operatora.

5.7. Optimalne kombinacije

S obzirom na sva provedena testiranja i usporedbe, u tablici 5.3 dane su najbolje kombinacije parametara za svaki problem. Uzeti su problemi CVRP i VRPTW s 25 kupaca te VRPPD sa 100 kupaca.

Tablica 5.3: Optimalne kombinacije parametara za svaki problem.

	CVRP	VRPTW	VRPPD
Oblik zapisa	lista	permutacija	permutacija
Algoritam	genetski algoritam	genetski algoritam	genetski algoritam
Broj iteracija	120 000	20 000	60 000
Veličina populacije	30	100	30
Najbolji rezultat	269.086	10222.2	33846

6. Zaključak

U ovom radu obrađen je problem usmjeravanja vozila čiji je cilj pronalaženje optimalnog puta dostave robe kupcima, s obzirom na određena ograničenja. Ispitane su instance ovog problema s ograničenim kapacitetom (CVRP), vremenskim prozorima (VRPTW) i mogućnošću preuzimanja tereta i dostave (VRPPD). Potrebno je minimizirati cijenu rute koja ovisi o prijeđenoj udaljenosti, poštivanju kapaciteta vozila te, u slučaju VRPTW-a, i o čekanju i kašnjenju vozila.

Ovaj problem spada u NP teške probleme i zbog njegove kompleksnosti rijetko kada se može riješiti egzaktnim metodama. Zbog toga su u ovom radu na probleme usmjeravanja vozila primjenjeni metaheuristički algoritmi koji, iako ubrzavaju rješavanje, ne garantiraju pronalazak optimalnog rješenja. Korištena su dva metaheuristička algoritma, genetski algoritam turnirske eliminacije i algoritam evolucijske strategije.

Obrađena su i uspoređena tri načina zapisa rješenja VRP problema: u obliku vektora permutacije, liste i mape. Za svaki od njih implementiran je jedan operator mutacije i jedan operator rekombinacije, te zajednički operator evaluacije. Uz korištenje turnirskog genetskog algoritma oblik permutacije dalje najbolje, dok zapis u obliku mape daje daleko najgore rezultate. Uz evolucijsku strategiju dobivaju se u prosjeku lošiji rezultati nego uz genetski algoritam, međutim tada zapis u obliku mape dolazi do puno boljih rješenja nego ostali zapisi.

Testiranjem uspješnosti ovisno o broju jedinki u populaciji dolazi se do zaključka kako povećanjem broja jedinki, i time smanjenjem ukupnog broja generacija, rezultati postaju sve lošiji. Za genetski turnirski algoritam dolazi se do dobrih rješenja uzimajući do sto jedinki u populaciji, dok se kod evolucijske strategije dobivaju to bolja rješenja što je broj jedinki po generaciji manji. Pri tome se vidi važnost evolucije kroz generacije odnosno važnost operatora mutacije i rekombinacije.

Također, implementirana su i dva lokalna operatora: operator lokalne pretrage i operator lokalnog spajanja. Operator lokalne pretrage provodi uzastopno određen broj mutacija najbolje jedinke i na taj način pokušava poboljšati rješenje. Nakon

primjene tog operatora vidljivo je značajno poboljšanje rezultata u odnosu na prije njegovog korištenja. Prikazana je i ovisnost kvalitete rješenja o dozvoljenom broju poziva operacije mutacije; što je on veći, rješenja su bolja. Operator lokalnog spajanja spaja dvije najkraće rute u jednu i za neke primjere su se dobivala bolja rješenja nakon korištenja operatora, međutim za dosta njih nije bilo značajnih poboljšanja u rezultatima. Ako pak poboljšanja ima, ono je to veće što se češće poziva operator.

Iz svega navedenoga se može zaključiti da ovom problemu najviše odgovara zapis rješenja u obliku vektora permutacije te genetski algoritam s turnirskom selekcijom. Uz korištenje operatora lokalne pretrage dobit će se još bolji rezultati te je moguće i manje poboljšanje korištenjem operatora spajanja ruta.

Literatura

[1] Jason Brownlee. *Clever Algorithms: Nature-Inspired Programming Recipes*, poglavlje Evolution Strategies. 2015. URL http://www.cleveralgorithms.com/nature-inspired/evolution/evolution_strategies.html.

[2] Tonci Caric i Hrvoje Gold (I-Tech). *Vehicle Routing Problem*. In-Teh, 2008. URL https://bib.irb.hr/datoteka/433524.Vehnicle_Routing_Problem.pdf.

[3] Jean-François Cordeau, Gilbert Laporte, Martin W.P. Savelsberg, i Daniele Vigo. *Vehicle Routing*. C. Barnhart and G. Laporte (Eds.), Handbook in OR & MS, Vol. 14, 2007. URL http://dis.unal.edu.co/~gjhernandezp/TOS/ROUTING/VRP1.pdf.

[4] G. B. Dantzig i J. H. Ramser. The truck dispatching problem. *Management Science, Vol. 6, No. 1 (Oct., 1959), pp. 80-91*, 1959. URL https://andresjaquep.files.wordpress.com/2008/10/2627477-clasico-dantzig.pdf.

[5] Juraj Fosin, Tonci Caric, i Edouard Ivanjko. Vehicle routing optimization using multiple local search improvements. *Automatika : časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije, Vol. 55 No. 2*, 2014. URL https://doi.org/10.7305/automatika.2014.01.580.

[6] Bhawna Minocha i Saswati Tripathi. Solving time constrained vehicle routing problem using hybrid genetic algorithm. Technical report, Amity School of Computer Sciences, Noida, India; Indian Institute of Foreign Trade, Kolkata, India, 2011. URL https://www.researchgate.net/publication/326630494_Solving_Time_Constrained_Vehicle_Routing_Problem_Using_Hybrid_Genetic_Algorithm.

[7] Networking i Emerging Optimization (NEO). Vehicle routing problem, 2013. URL http://neo.lcc.uma.es/vrp.

[8] Beatrice Ombuki, Brian J. Ross, i Franklin Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. Technical report,

Brock University. Dept. of Computer Science, St. Catharines, ON, Canada L2S 3A1, 2004. URL https://www.researchgate.net/publication/227309327_Multi-Objective_Genetic_Algorithms_for_Vehicle_Routing_Problem_with_Time_Windows.

[9] Devin Soni. Introduction to evolutionary algorithms. 2018. URL https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac.

Primjena metaheurističkih algoritama na problem usmjeravanja vozila s vremenskim prozorima i mogućnošću preuzimanja tereta

Sažetak

Ovaj rad obrađuje problem usmjeravanja vozila, te njegove instance s ograničenim kapacitetom, vremenskim prozorima i mogućnošću preuzimanja tereta i dostave. Dan je detaljan opis problema i korištenih metaheurističkih algoritama: genetskog algoritma turnirske eliminacije i algoritma evolucijske strategije. Implementirana su tri oblika zapisa rješenja, operatori mutacije i križanja i dva lokalna operatora. Istraživana je ovisnost kvalitete rješenja o obliku zapisa, korištenom algoritmu i korištenju lokalnih operatora. Dobiveni rezultati prikazani su grafički i analizirani.

Ključne riječi: vrp, cvrp, vrptw, vrppd, metaheuristika, genetski algoritam, evolucijska strategija, lokalno pretraživanje, optimizacija

Metaheuristic algorithms for vehicle routing problems with time windows and pick-up and delivery

Abstract

This paper deals with vehicle routing problem and its instances with limited capacity, time windows and pick-up and delivery. Detailed description of the problem is given, as well as of the two metaheuristic algorithms that were used: genetic algorithm with tournament selection and evolution strategy algorithm. Three forms of results, mutation and crossover operators and two local operators were implemented. The dependence of the quality of the solution on the form of result, used algorithm and the use of local operators was explored. The obtained results are shown graphically and analyzed.

Keywords: vrp, cvrp, vrptw, vrppd, metaheuristics, genetic algorithm, evolution strategy, local search, optimization