SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6392

Rješavanje problema usmjeravanja vozila metaheuristikama u statičkim i dinamičkim uvjetima

Jakov Vidulić

Zagreb, srpanj 2019.

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

Zagreb, 14. ožujka 2019.

ZAVRŠNI ZADATAK br. 6392

Pristupnik: **Jakov Vidulić (0036495953)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Rješavanje problema usmjeravanja vozila metaheuristikama u statičkim i dinamičkim uvjetima**

Opis zadatka:

Opisati problem usmjeravanja vozila i postojeće metode rješavanja toga problema. Navesti inačice problema s obzirom na zadana ograničenja i dati pregled uobičajenih pristupa rješavanju. Ostvariti programski sustav za rješavanje problema usmjeravanja vozila u statičkim uvjetima. Ispitati mogućnost uporabe metaheurističkih postupaka u dinamičkim uvjetima rada. Ispitati učinkovitost različitih postupaka primijenjenih na navedeni problem, s obzirom na kvalitetu rješenja i vremensku složenost algoritma. 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

Predsjednik odbora za završni rad modula:

Doc. dr. sc. Marko Čupić

Djelovođa:

Izv. prof. dr. sc. Tomislav Hrkać

Hvala svima koji su bili dijelom mojeg dosadašnjeg obrazovanja.

SADRŽAJ

Popis slika **vi**

Popis tablica **viii**

1. Uvod **1**

2. Problem usmjeravanja vozila **3**

2.1. CVRP 5

2.2. VRPTW 6

2.3. VRPPD 6

3. Najčešći pristupi rješavanja **7**

3.1. Egzaktni pristup 7

3.1.1. Branch and bound 7

3.2. Heuristički pristup 8

3.2.1. Clarke and Wright 8

3.3. Metaheuristički pristup 10

3.3.1. Algoritam kolonije mrava 10

3.3.2. Evolucijski algoritmi 11

3.4. Genetski algoritam 12

3.4.1. Početak genetskog algoritma 12

3.4.2. Funkcija dobrote 13

3.4.3. Selekcija 13

3.4.4. Mutacija 13

3.4.5. Križanje 14

4. Programsko ostvarenje **15**

4.1. Prikazi rješenja 15

4.1.1. Prikaz rješenja permutacijom 15

4.1.2. Prikaz rješenja permutacijskim nizom i listom vozila 16
4.1.3. Prikaz rješenja mapom 17
4.2. Evaluacija rješenja 19

5. Analiza rezultata **21**
5.1. Korišteni algoritmi 21
5.1.1. SteadyStateTournament 21
5.1.2. Evolution Strategy 22
5.2. Kriterij zaustavljanja 23
5.3. Rezultati u statičkim uvjetima 24
5.3.1. Veličina populacije 24
5.3.2. Rezultati dobiveni algoritmom SteadyStateTournament 24
5.3.3. Rezultati dobiveni algoritmom Evolution Strategy 30
5.3.4. Najbolje kombinacije parametara 33
5.4. Rezultati u dinamičkim uvjetima 33
5.4.1. Vrijeme do pronalaska heuristički dobivenih rješenja 33
5.4.2. Usporedba vremena ovisno o prikazu 35

6. Zaključak **37**

Literatura **38**

POPIS SLIKA

2.1. VRP (*www.neo.lcc.uma.es/vrp*) 4
2.2. Inačice VRP-a (*www.wikipedia.org*) 5

3.1. Algoritam uštede 9
3.2. Algoritam kolonije mrava (*www.sciencedirect.com*) 11
3.3. Primjer mutiranja jedinke 13
3.4. Križanje dviju jedinki 14

4.1. Oblik zapisa rješenja permutacijskim nizom 15
4.2. Genom nakon djelovanja operatora mutacije 16
4.3. Rješenje prikazano listom klijenata i vozila 16
4.4. Operator križanja za prikaz rješenja mapom 19

5.1. SteadyStateTournament pseudokod *[3]* 22
5.2. Evolution Strategy pseudokod *[3]* 23
5.3. Poboljšanje dobrote rješenja kroz evaluacije 23
5.4. CVRP 25 klijenata - rješenje permutacijskim nizom 25
5.5. CVRP 25 klijenata - rješenje dvjema listama 25
5.6. CVRP 25 klijenata -rješenje mapom 25
5.7. CVRP 50 klijenata - rješenje permutacijskim nizom 26
5.8. CVRP 50 klijenata - rješenje dvjema listama 26
5.9. CVRP 50 klijenata - rješenje mapom 26
5.10. VRPTW 50 klijenata - rješenje permutacijskim nizom 27
5.11. VRPPD 100 klijenata - rješenje permutacijskim nizom 27
5.12. CVRP 50 klijenata - rješenje permutacijskim nizom uz vjerojatnost mutacije 0.6 28
5.13. CVRP 50 klijenata - rješenje dvjema listama uz vjerojatnost mutacije 0.6 28
5.14. CVRP 50 klijenata - rješenje mapom uz vjerojatnost mutacije 0.6 . . . 28

5.15. VRPTW 50 klijenata - rješenje permutacijskim nizom uz vjerojatnost mutacije 0.6 29
5.16. VRPPD 100 klijenata - rješenje permutacijskim nizom uz vjerojatnost mutacije 0.6 29
5.17. CVRP 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30 30
5.18. VRPTW 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30 31
5.19. VRPPD 100 klijenata - usporedba prikaza rješenja uz veličinu populacije 30 31
5.20. CVRP 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30 32
5.21. VRPTW 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30 32
5.22. VRPPD 100 klijenata - usporedba prikaza rješenja uz veličinu populacije 30 32
5.23. Iznos dobrote kroz evaluacije 36
5.24. Iznos dobrote kroz evaluacije 36

POPIS TABLICA

4.1. Rješenje sa slike 4.1 prikazano mapom 18
4.2. Drugi roditeljski genotip 18
4.3. Genotip strukture mape dobiven operatorom križanja 19

5.1. Tablica najboljih parametara 33
5.2. Instance problema sa zadanim rješenjima dobivenim heuristikama . . 33
5.3. Vremena konvergencije ka rješenju za VRPTW problem 25 klijenata . 34
5.4. Vremena konvergencije ka rješenju za VRPTW problem 50 klijenata . 34
5.5. Vremena konvergencije ka rješenju za VRPPD problem 100 klijenata 35
5.6. Vremena generiranja rješenja za VRPTW problem 1000 klijenata . . . 35

1. Uvod

Otkad se trgovina pojavila na našoj planeti, ljudi neprestano pokušavaju pronaći način kako najbrže dopremiti robu na odredišta uz najmanji trošak i najveću zaradu. U samom početku su ljudi koji su trgovali dobrima neizbježno ovisili o prirodi, primjerice životinjama, vremenskim prilikama i vjetru, te time ograničeni nailazili na poteškoće pri razmjeni dobara, no nakon parne i industrijske revolucije trgovanje je postalo iznimno jednostavnije i granice trgovine su se širile sve do danas.

Cestovni promet je najzastupljeniji oblik prometa na svijetu te je od krucijalne važnosti osobito u današnje vrijeme, u doba sve veće potražnje, dostupnosti trgovine na internetu i globalizacije. Kao posljedica globalizacije trgovina putem interneta isporučuje sve manje pakete, a ujedno i sve raznolikiju robu te je uslijed toga potreba za čim efikasnijim rješenjima prijevoza robe izraženija.

Svaka organizacija u čije je djelovanje uključen prijevoz dobara suočava se s raznim problemima vezanim uz logistiku i distribuciju dobara. U nekim tvrtkama udio troška prijevoza čini velik udio u sveukupnim troškovima logistike. Zato tvrtke u svrhu vlastitog profita traže optimalna rješenja u kontekstu vremena jer, kako i poznata narodna krilatica kaže, *vrijeme je novac*.
Razvojem računarske znanosti kroz godine uspješno se izlazi na kraj s takvom vrstom problema koji se općenito naziva *problem usmjeravanja vozila*.

Problem usmjeravanja vozila ima važnu ulogu u prijevozu, logistici i distribuciji dobara. Primjenjivost i kompleksnost problema je očita i za očekivati je proširenje i dodatno specificiranje problema ovisno o potrebama raznih organizacija. U ovom završnom radu obrađene su tri inačice problema usmjeravanja vozila — problem s ograničenim kapacitetom, problem s vremenskim prozorom i problem dostavljanja i prikupljanja tereta. Rješenje svih triju inačica problema implementirano je genetskim algoritmom.

U nastavku rada u drugom poglavlju detaljno je opisan problem usmjeravanja vozila kao i promatrane inačice problema. Nakon toga u trećem poglavlju navedeni su najčešći pristupi problemu. Detaljnije je opisana ideja pronalaženja rješenja pomoću genetskog algoritma. U četvrtom poglavlju opisana je implementacija rješenja kroz tri različita prikaza. U petom poglavlju navedeni su rezultati prilikom ispitivanja funkcionalnosti implementacije ovisno o statičkoj i dinamičkoj komponenti problema. Na samom kraju rada, u šestom poglavlju, doneseni su zaključci temeljem dobivenih rezultata.

2. Problem usmjeravanja vozila

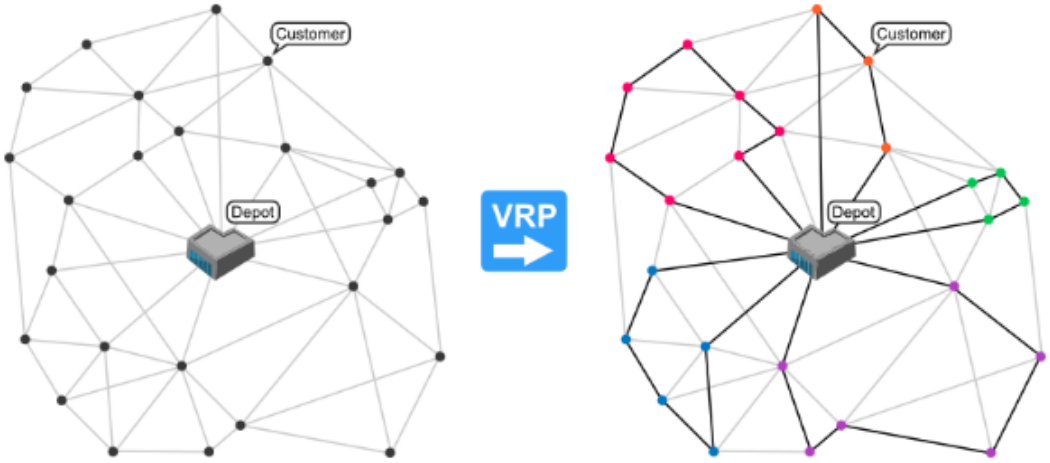
Problem usmjeravanja vozila, poznatiji pod akronimom VRP (engl. *Vehicle Routing Problem*), je općeniti naziv za skup problema u kojima je potrebno skupinom vozila koja se nalaze u skladištu obići sve klijente koji su u potražnji za nekom uslugom uz uvjet da svakog klijenta posluži isključivo jedno vozilo. VRP pripada kategoriji *NP teških problema*[1] za koje je teško naći optimalno rješenje. Jedan je od najčešće proučavanih problema kombinatorne optimizacije čiji je zadatak pronaći optimalne rute vozila takve da minimiziraju trošak pri posjećivanju klijenata uz zadovoljavanje svih zadanih ograničenja.

Prije šezdeset godina Dantzig i Ramser [2] su predstavili i formirali matematičko programski model problema. Opisali su VRP na konkretnom primjeru dostave goriva benzinskim stanicama.
Problem usmjeravanja vozila je problem koji se temelji na teoriji grafova te se na osnovu toga koristi sljedeća uobičajena notacija:

- $G(V, E)$ predstavlja graf
- $V = \{v_0, v_1, \ldots, v_n\}$ je skup vrhova grafa gdje:
  - skladište se nalazi u v_0.
  - $V' = V \setminus \{v_0\}$ predstavlja skup n klijenata.
- $A = \{(v_i, v_j) | v_i, v_j \in V; i \neq j\}$ predstavlja skup lukova grafa.
- d je vektor potražnje klijenata.
- R_i je ruta vozila i.
- m je broj vozila. Jedna je ruta dodijeljena svakom vozilu.

Ruta vozila R_i je prihvatljiva ako vozilo posjeti svakog klijenta samo jedanput i ako zadovoljava ostale kriterije koji ovise o specifikaciji problema.

[1] nepolinomijalno teški problemi



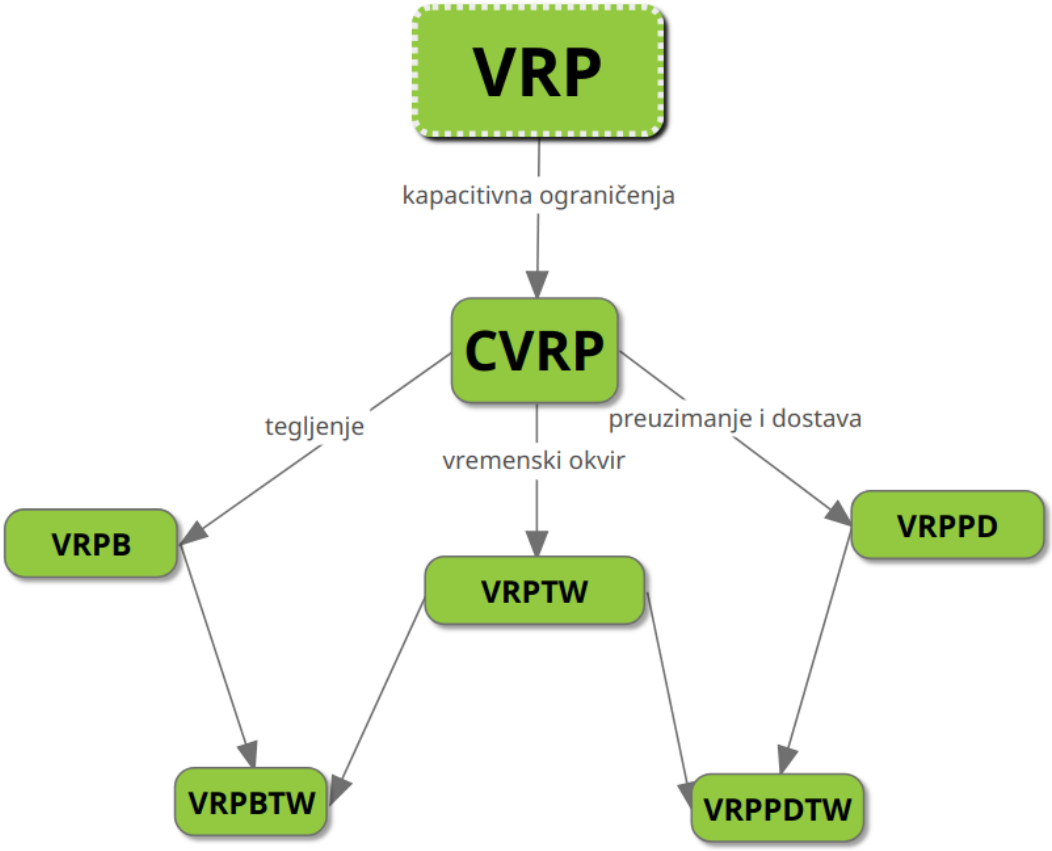
Slika 2.1: VRP (*www.neo.lcc.uma.es/vrp*)

Problem usmjeravanja vozila usko je vezan uz problem trgovačkog putnika. Problem trgovačkog putnika, poznatiji kao TSP (engl. *Travelling Salesman Problem*), jedan je od najpoznatijih i najistraživanijih problema kombinatorne optimizacije. U TSP-u je definiran skup gradova koje trgovački putnik mora posjetiti uz uvjet da svaki grad posjeti točno jednom i da se na kraju putovanja vrati u grad iz kojeg je krenuo. Cilj je pronaći redoslijed obilaska gradova koji rezultira minimalnim troškovima.

Ukoliko definiramo trgovačke putnike kao vozila, a gradove koje putnik posjećuje kao klijente kojima vozila dopremaju dobra preveli smo problem trgovačkog putnika u problem usmjeravanja vozila.

Primarni cilj problema usmjeravanja vozila je minimizacija broja vozila, a sekundarni cilj je najčešće minimizacija vremena ili ukupnog prijeđenog puta prilikom posluživanja klijenata. Pojedine varijante problema usmjeravanja vozila mogu imati različita ograničenja npr. broj skladišta, količina tereta koje je potrebno prevesti te ostala ograničenja vezana uz svojstva klijenata ili vozila.

Zbog fleksibilnosti zahtjeva klijenata iz početnog problema usmjeravanja vozila proizašlo je mnoštvo inačica problema raznih ograničenja. Na slici 2.2 prikazane su najčešće promatrane varijante problema.



Slika 2.2: Inačice VRP-a (*www.wikipedia.org*)

2.1. CVRP

Kapacitivni problem usmjeravanja vozila, poznatiji kao CVRP (eng. Capacitated Vehicle Routing Problem), je najosnovniji tip problema usmjeravanja vozila. U ovoj inačici problema broj ograničenja je minimalan. Fiksiran je broj vozila i kapacitet koji je svim vozilima jednak, lokacija klijenata i njihova potražnja za robom. Dodatni uvjeti su da svakog klijenta posluži isključivo jedno vozilo te da sva vozila kreću iz skladišta i u skladište se vraćaju nakon što odvoze svoje rute. Također, zbroj potražnje za robom klijenata jedne rute ne smije premašiti kapacitet vozila.

Funkcija cilja definirana je minimiziranjem ukupnog troška pri posluživanju svih klijenata. Rješavanje problema podrazumijeva pronalazak rješenja po kojima će ukupna prevaljena udaljenost biti minimalna.

2.2. VRPTW

Problem usmjeravanja vozila s vremenskim ograničenjima, poznatiji kao VRPTW (engl. *Vehicle Routing Problem with Time Windows*), je proširenje kapacitivnog problema usmjeravanja vozila gdje osim ograničenja u kapacitetu svakog vozila postoje i ograničenja svakog klijenta u kontekstu vremena posluživanja. Svaki klijent mora biti poslužen unutar vremenskog okvira [$t1$, $t2$], dok za vrijeme posluživanja vozilo mora biti u stanju mirovanja.
Tolerira se dolazak vozila na mjesto posluživanje klijenta prije predviđenog vremena dolaska i u tom slučaju vozilo čeka na posluživanje do definiranog trenutka $t1$ kada započinje s istovarom. Osim vremena $t1$ i $t2$ definirano je i posluživanje klijenta. Rješenje VRPTW čini N ruta s minimalnim troškom i zadovoljavanjem svih uvjeta kapacitivnog problema usmjeravanja vozila uz dodatan uvjet posluživanja klijenata unutar vremenskog okvira [$t1$, $t2$].

2.3. VRPPD

Problem usmjeravanja vozila s dostavom i prikupljanjem, poznatiji pod akronimom VRPPD (engl. *Vehicle Routing Problem with Pickup and Delivery*), je proširenje problema usmjeravanja vozila u kojem se javljaju dvije dodatne vrste zahtjeva svakog klijenta, zahtjev za dostavom i zahtjev za prikupljanjem dobara.
Klijenti se dijele na one koji primaju i na one koji šalju dobra te je svaki zahtjev definiran klijentima koji obavljaju razmjenu robe te količinom robe koja se prevozi. Ovo svojstvo čini planiranje rute puno težim i može dovesti do lošeg iskorištavanja kapaciteta vozila, povećanjem ukupne prijeđene udaljenosti i potrebom za novim vozilima. Pretpostavka je da je vozilo koje obavlja rutu u kojoj se ima izvršiti dostava tereta prije dostavljanja prethodno prikupilo teret klijenta pošiljatelja. Osim toga, važno je zadovoljiti uvjet ograničenog kapaciteta vozila prilikom prikupljanja tereta na mjestu klijenta pošiljatelja.

3. Najčešći pristupi rješavanja

Postoje dva osnovna pristupa pri rješavanju problema usmjeravanja vozila. Jedan je pronaći egzaktno rješenje, koje je zapravo optimalni Hamiltonov ciklus u grafu. [6] Za manji broj klijenata to nije tako teško napraviti — u težinskom grafu treba pronaći sve Hamiltonove cikluse i odabrati onaj najmanje težine.
Problem se komplicira pri nešto većem broju klijenata kad je egzaktni pristup zbog faktorijelne složenosti nepoželjan jer za posljedicu ima iscrpljivanje resursa računala.
Drugi pristup je pronaći dovoljno dobro rješenje problema koristeći heurističke i metaheurističke funkcije.

3.1. Egzaktni pristup

Kao što samo ime kaže, ovaj pristup nalaže izračunavanje svakog mogućeg rješenja sve dok se ne pronađe ono optimalno. Egzaktni algoritmi su optimalni[1], no zbog izrazite vremenske i prostorne složenosti u slučaju većeg broja klijenata puno rjeđe su korišteni u praksi. Pristup je primjenjiv isključivo za manji broj korisnika.

3.1.1. Branch and bound

Jedan od najuspješnijih pristupa problemu usmjeravanja vozila, naročito varijante CVRP, je metoda K-stabla koju je 1994. opisao Fisher [4] koji je tada uspješno riješio problem za 71 klijenta. Fisherova metoda koristi algoritam **grananja i ograđivanja** (engl. *branch and bound*) koji se bazira na principu podijeli pa vladaj (lat. *divide et impera*).

Algoritam se može podijeliti na tri koraka: grananje (engl. *branching*), ograđivanje (engl. *bounding*) i procjenjivanje (engl. *fathoming*). U prvom koraku je osnovna ideja podijeliti problem na više rješivih potproblema tako što se klijente grupira po

[1] optimalan algoritam - algoritam koji pronalazi optimalno rješenje (ono s najmanjom cijenom)

bridovima incidencije[2]. U drugom koraku ograničavanjem vrijednosti određenih varijabli prostor rješenja se dijeli unutar grana na potprostore i konačno procjenom gornje i donje granice rješenja unutar grane moguće je odbaciti dijelove potprostora rješenja koji se ne nalaze unutar definiranih granica. Trajanje postupka direktno ovisi o broju čvorova u stablu grananja koji se obrađuju stoga je za veće probleme ipak potrebno koristiti heurističke i metaheurističke metode.

3.2. Heuristički pristup

Heuristički pristup je pristup rješavanju problema koristeći heuristike. Heuristike su tehnike rješavanja problema temeljene na iskustvenim pravilima o prirodi problema i osobinama cilja čija je svrha pretraživanje brže usmjeriti k cilju. Suprotno od egzaktnog pristupa, heuristički pristup nema znanja o strukturi ili relacijama modela zadanog problema. Heuristički algoritmi su algoritmi nastali eksperimentiranjem u svrhu dobivanja zadovoljavajućeg rješenja.

3.2.1. Clarke and Wright

Clarke and Wright algoritam uštede [1] je jedan od najpoznatijih i najkorištenijih heurističkih algoritama za rješavanje problema usmjeravanja vozila.
Ideja algoritma je vrlo jednostavna. Imamo skladište D i n klijenata. Pretpostavimo na početku da se rješenje sastoji od n vozila i slanja jednog vozila svakom od n klijenata. Ukupna duljina puta za ovo rješenje je očito $2 \cdot \sum_{i=1}^{n} d(D,i)$. Ako u sljedećem koraku jedno vozilo posluži dva klijenta u istoj ruti, primjerice i i j, ukupna duljina puta je umanjena za s(i,j):

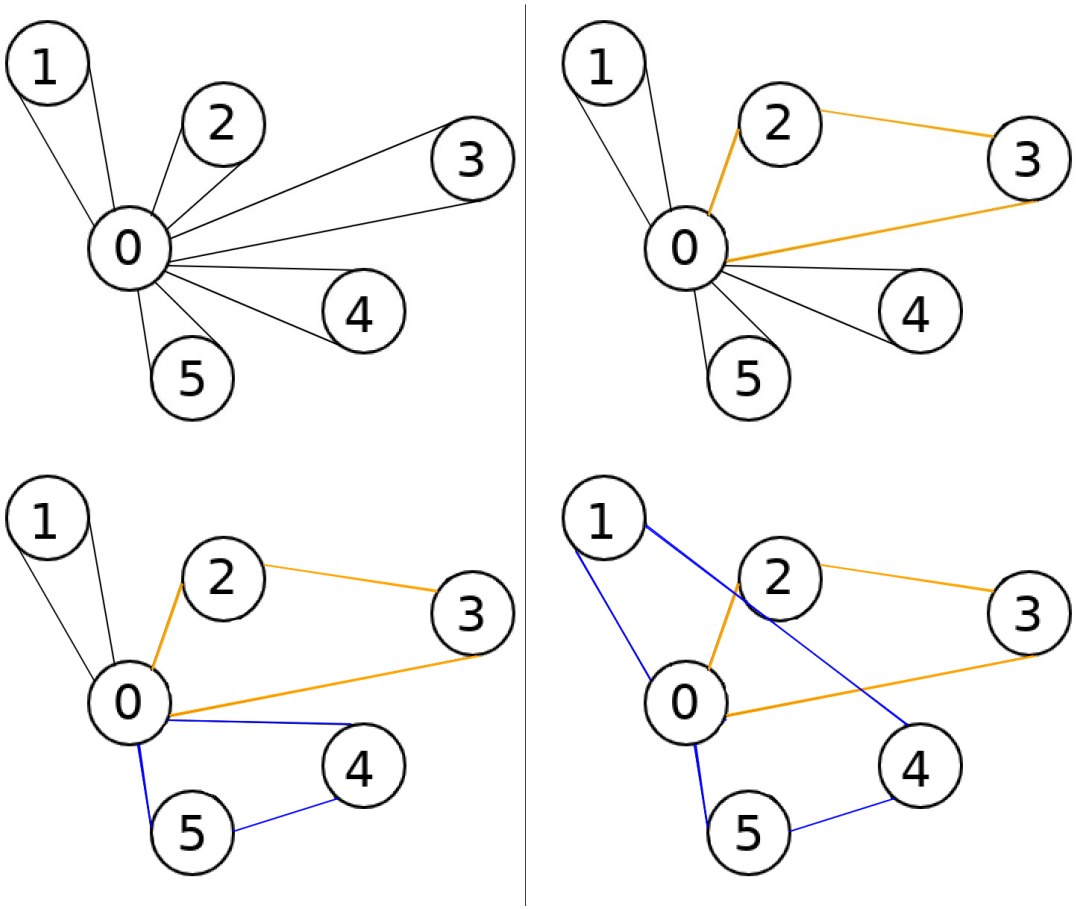
$$s(i,j) = 2d(D,i) + 2d(D,j) - [d(D,i) + d(i,j) + d(D,j)] \tag{3.1}$$

$$s(i,j) = d(D,i) + d(D,j) - d(i,j) \tag{3.2}$$

[2] susjedni bridovi

Algoritam uštede može se opisati u četiri koraka:

1. Izračun ušteda $s(i,j) = d(D,i) + d(D,j) - d(i,j)$ za svaki par klijenata (i, j)
2. Sortiranje ušteda $s(i,j)$ od najveće prema najmanjoj i stvaranje "liste ušteda".
3. Uzimamo najveću $s(i,j)$ iz liste i njezinu rutu proširujemo obilaskom klijenata k i l pod uvjetom da proširenje rute klijentima (k,l) ne narušava ni jedno od zadanih ograničenja problema i pod uvjetom da se:
   (a) dva klijenta već ne nalaze u istoj ruti.
   (b) nijedan klijent ne nalazi odmah pored rute koju proširujemo.
4. Korak 3 se ponavlja dok se ne generiraju rute maksimalno mogućih ušteda.



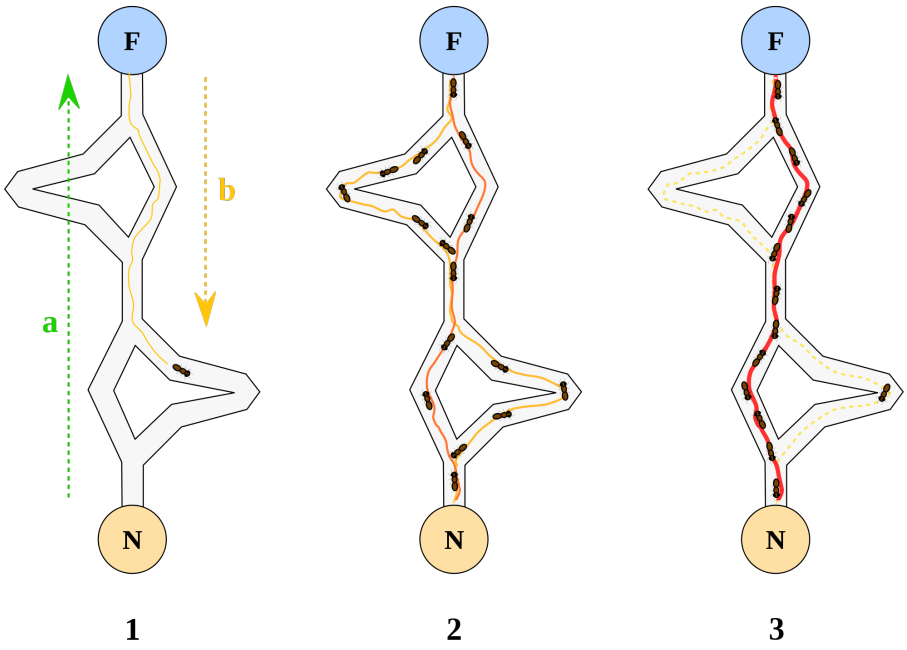
Slika 3.1: Algoritam uštede

3.3. Metaheuristički pristup

Metaheuristike su algoritmi koji se koriste za definiranje heurističkih metoda na širok skup problema. Izuzetno su primjenjivi na problemima čije je razumijevanje ograničeno za izradu konkretne heuristike te na problemima optimizacije.
Metaheurističke algoritme dijelimo u dvije skupine: algoritme koji djeluju nad jednim rješenjem (simulirano kaljenje, tabu pretraživanje) i na populacijske algoritme koji djeluju nad skupom rješenja (evolucijski i genetski algoritmi) [8]. Velik broj ovih algoritama je inspiriran procesima iz prirode. Pri rješavanju NP-teških optimizacijskih problema, koji nisu rješivi egzaktnim algoritmima u realnom vremenu, ovaj se pristup pokazao iznimno učinkovitim.

3.3.1. Algoritam kolonije mrava

Ova metaheuristika je inspirana prirodnom metodom — suradnjom koju ostvaruju mravi u stvarnom životu koja im omogućava da pronađu puteve od mravinjaka do hrane. Kanal kojim mravi komuniciraju je kemijski spoj, feromon, koji se ostavlja na tlo. Dok bi mrav sam u svojoj ekspediciji za hranom uglavnom lutao, mrav u koloniji otkriva put feromona koji će slijediti te će svojim vlastitim feromonom taj put ojačati. Na temelju tog ponašanja u prirodi nastao je algoritam kolonije mrava.
Vjerojatnost da će drugi mravi slijediti određeni put u budućnosti povećava se s brojem mrava koji su ranije prošli tim putem i slijedili ga. To dovodi do formiranja najkraćeg puta kojeg je akumulirani feromon "utabao".
U algoritmu poznatom pod akronimom ACO (engl. *Ant Colony Optimization*) broj umjetnih mrava kreira rješenje na nasumičan i pohlepan način. Svaki mrav odabire sljedeći element koji će ugraditi u svoje trenutno djelomično rješenje temeljem heurističke evaluacije tog elementa i količine feromona. Heuristička evaluacija je predstavljena težinom koja joj je pridijeljena. Feromon predstavlja memoriju sustava i prisutnost elementa dobrog rješenja kojeg su mravi generirali.
ACO je prvotno uspješno primijenjen na problem trgovačkog putnika (TSP), a potom je prilagođen i problemu usmjeravanja vozila kao i nekim njegovim inačicama.



Slika 3.2: Algoritam kolonije mrava (*www.sciencedirect.com*)

3.3.2. Evolucijski algoritmi

Evolucijske algoritme čine stohastičke metode pretraživanja koje su inspirirane biološkom evolucijom. Baš kao što u prirodi preživljavanje neke jedinke ovisi o kvaliteti adaptacije okolini tako se i ovi algoritmi temelje na principu preživljavanja najboljih jedinki u populaciji. Taj mehanizam se u evolucijskim algoritmima naziva ključ selekcije. Selekcija se obavlja nad populacijom genetskog algoritma koja je pandan živim bićima u prirodi. Korištenjem genetskih operatora moguće je mutirati genetski kod te tako stvoriti novu populaciju nad kojom se ponavlja postupak dok nije zadovoljen određeni kriterij zaustavljanja.

Evolucijski algoritmi su: evolucijsko programiranje, evolucijska strategija, genetsko programiranje te genetski algoritam koji će se u nastavku poglavlja pobliže objasniti i čija će se primjena naći u programskog ostvarenju konkretnog problema.

3.4. Genetski algoritam

Genetski algoritam GA (engl. *Genetic Algorithm*) je tehnika globalne pretrage koja rješava probleme oponašajući evolucijske procese u prirodi. GA ima vrlo široke mogućnosti upotrebe, posebice za rješavanje problema koji se teško definira. Genetski algoritam razvija populaciju grupe bitova ili genoma, u kojoj svaki genom predstavlja jedno od rješenja problema.

Sama evolucija populacije odvija se primjenom operatora koji oponašaju operacije nad genima u stvarnom životu. Genetski algoritam inspiriran je Darwinovom[3] teorijom evolucije, koristeći tehnike selekcije, mutacije, nasljeđivanja i križanja. Ideja genetskog algoritma je zbog toga vrlo intuitivna i jasna svima, programerima i laicima. Mada je princip vrlo jednostavan, primjena ovog algoritma je iznimno široka. Najčešća metoda korištena u genetskim algoritmima je moderiranje grupe jedinki dane populacije. Tako stvorene jedinke se ocjenjuju evaluacijskom funkcijom.

Rješenje problema generirano genetskim algoritmom evoluira. U usporedbi s ostalim metodama umjetne inteligencije, genetski algoritam ima puno prednosti. Robusniji je i sposobniji za brzo otkrivanje lošijih jedinki uzrokovane manjim promjenama. Obzirom na druge optimizacijske metode kao što su linearno programiranje, heurističke metode, pretraživanje u širinu i ostale genetski algoritam pruža bolje rezultate pri pretraživanju velikih stanja.

3.4.1. Početak genetskog algoritma

Genetski algoritam započinje sa skupom rješenja (predstavljen genotipom) koji nazivamo **populacija**. Rješenja jedne populacije se koriste za generiranje novih populacija rješenja. Osnovna motivacija ovog algoritma je nada da će nova populacija biti bolja od one prethodne. Rješenja koja su odabrana (*roditelji*) za formiranje novog rješenja (*potomak*) odabrana su temeljem funkcije dobrote - što su prikladnije rješenju problema to je veća šansa da će postojeće rješenje biti odabrano za roditelja. Postupak se ponavlja sve dok određeni uvjet nije zadovoljen. Taj uvjet u ovom kontekstu je najčešće veličina populacije ili uvjet zaustavljanja prilikom stagnacije najboljeg rješenja.

[3] Charles Darwin, autor moderne teorije o evoluciji

3.4.2. Funkcija dobrote

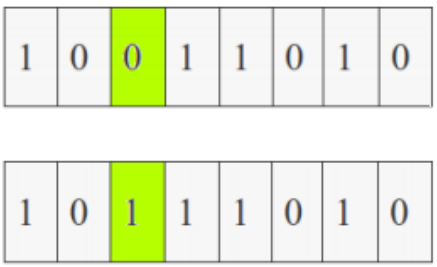
Funkcija dobrote (engl. *fitness function*) je jedna od najvažnijih sastavnica genetskog algoritma. Ona služi za ocjenu kvalitete jedinke i predstavlja ključ selekcije. U najjednostavnijoj varijanti genetskog algoritma funkcija dobrote je ekvivalent funkcije koju treba optimizirati. Međutim, u rješavanju ostalih problema ispravno određivanje funkcije dobrote je od krucijalne važnosti za konačan ishod algoritma. Određuje se uzimanjem u obzir svih parametara i mogućih scenarija realnog problema. Temeljem vrijednosti funkcije dobrote određuje se koje jedinke u populaciji ostaju, a koje se odbacuju i to na način da jedinke s manjom vrijednošću imaju manju vjerojatnost preživljavanja. Na taj način najbolje jedinke, baš kao i u Darwinovoj teoriji, preživljavaju i utječu na buduće generacije, dok najlošije odumiru.

3.4.3. Selekcija

Selekcija je jedna od glavnih značajki evolucijskih algoritama. Operator selekcije je od velike važnosti jer ima za zadaću propagirati najbolje jedinke u sljedeću generaciju. Odabir se temelji na vrijednosti funkcije dobrote, propagira se ono s najvećom dobrotom.

3.4.4. Mutacija

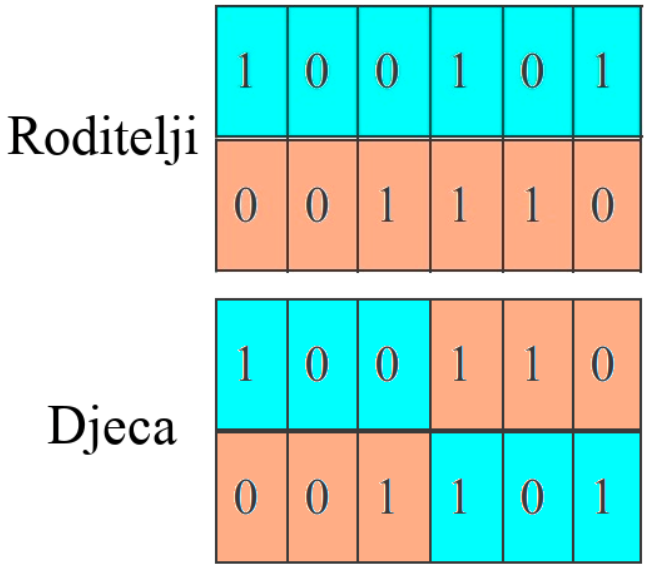
Mutacija je genetski operator u genetskom algoritmu koji, kako mu samo ime kaže, mutira genom. Mutiranje je najčešće posljedica nekih vanjskih čimbenika. Baš kao i u Darwinovoj teoriji evolucije, naizgled nebitnim odlukama i postupcima živih bića može se promijeniti njihov genetski materijal tako da nakon promjene bude bolje prilagođen uvjetima oko sebe i tako si poveća šansu za preživljavanjem. Mutacija se obavlja nad jednom jedinkom te je uz sam operator vezana i vjerojatnost kojom se jedinka mutira.



Slika 3.3: Primjer mutiranja jedinke

3.4.5. Križanje

Križanje, rekombinacija ili reprodukcija su sinonimi za genetski operator u genetskom algoritmu koji imitira prirodni proces razmnožavanja živih bića. Ideja je da prilikom križanja dviju jedinki (roditelja) koje su prošle kroz operator selekcije nastaje nova jedinka (potomak) koja sadrži dijelove genetskog materijala oba roditelja. Križanjem dviju kvalitetnih jedinki očekuje se jedinka također kvalitetnog genetskog materijala. Ovim postupkom se brzo postiže optimum, međutim moguće je da taj bude i lokalni optimum od kojeg se teže odmaknuti. Najjednostavniji primjer operacije križanja je križanje s jednom točkom prekida koje uzima genetski materijal do određene točke prvog roditelja te na njega nadovezuje genetski materijal drugog roditelja od te iste točke. Taj postupak prikazan je na slici 3.4:



Slika 3.4: Križanje dviju jedinki

4. Programsko ostvarenje

4.1. Prikazi rješenja

Rješenja svake promatrane inačice problema usmjeravanja vozila prikazana su pomoću triju različitih struktura kako bi se ispitao utjecaj strukture prikaza na kvalitetu konačnog rješenja i složenost algoritama nad pojedinim strukturama. Korištene su dinamičke strukture koje su pogodne za manipuliranje podacima problema.
Za svaki genotip registriran je operator križanja i mutacije. Prilikom izvršavanja algoritma sustav u kojem se evaluira rješenje operatore poziva s vjerojatnošću definiranom u datoteci za postavljanje parametara.

4.1.1. Prikaz rješenja permutacijom

Genotip, odnosno zapis rješenja problema je zapisan permutacijskim nizom. To je najjednostavniji prikaz rješenja problema usmjeravanja vozila. Prikaz se sastoji od liste cijelih brojeva koja je veličine $nV + nC$, gdje nV predstavlja broj vozila, a nC broj klijenata. U početnoj listi kojoj indeksi idu redom od 1 do $nV + nC$ se indeksi nasumično ispremještaju te se tako slučajno generiraju rješenja, pritom pazeći da prvi indeks bude indeks vozila, a ne indeks klijenta. Prikaz se dekodira na način da se nakon indeksa vozila navode klijenti onim redom kojim će ih vozilo obilaziti u ruti. Ako su dva indeksa vozila neposredno jedan iza drugog znači da vozilo pod prvim indeksom ima praznu rutu i nema dodijeljenih klijenata. Na slici 4.1 je prikazan primjer konačnog prikaza rješenja problema za 5 vozila i 25 klijenata.

26	17	18	13	11	27	6	23	22	21	2	1	9	28	7	16	24	3	14	29	4	20	25	8	30	10	15	12	19	5

Slika 4.1: Oblik zapisa rješenja permutacijskim nizom

Operator mutacije: U prikazu rješenja permutacijom operator mutacije je vrlo jednostavan. Nasumičnim odabirom bira dva člana genotipa i mijenja im mjesta.

Izgled genotipa nakon mutacije nad rješenjem sa slike 4.1:

26	17	18	13	11	27	6	23	22	21	15	1	9	28	7	16	24	3	14	29	4	20	25	8	30	10	2	12	19	5

Slika 4.2: Genom nakon djelovanja operatora mutacije

Operator križanja: Na početku operator križanja dobiva zadana tri genotipa strukture vektora. Nakon toga odabire dva slučajna indeksa koji određuju početak i kraj cjelobrojnog niza iz genoma prvog roditelja. Genom potomka čini tako dobiveni niz te preostali klijenti i vozila koji se ne nalaze već u istom nizu.

4.1.2. Prikaz rješenja permutacijskim nizom i listom vozila

Vrlo je slično prikazu permutacijskim nizom, razlika je u tome što se ovdje rješenje prikazuje dvjema cjelobrojnim listama — jednu listu čine klijenti, a druga lista iste veličine sadrži identifikatore vozila na odgovarajućem indeksu liste za svakog klijenta te tako definira pripadnost klijenta vozilu. Za već viđeno rješenje problema 5 vozila i 25 klijenata na slici 4.1 u ovom prikazu imamo sljedeći zapis:

6	7	16	17	23	4	18	20	10	13	22	21	24	25	15	3	11	2	14	1	12	19	9	5	8
27	28	28	26	27	29	26	29	30	26	27	27	28	29	30	28	26	27	28	27	30	30	27	30	29

Slika 4.3: Rješenje prikazano listom klijenata i vozila

Operator mutacije: u ovom prikazu radi sličnu mutaciju kao i u prikazu permutacijskim nizom, samo što se slučajno odabire lista nad kojom će se mutacija izvršiti (klijenata ili vozila).

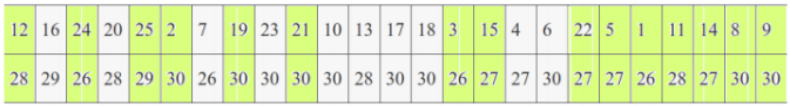
Operator križanja: Na početku su zadana tri genotipa promatrane strukture. Potom se nasumično određuju dva indeksa koja određuju koji dio liste klijenata i liste vozila prvog roditelja će činiti liste klijenata i vozila potomka. Nakon toga se iz genoma

drugog roditelja puni ostatak obiju lista klijentima i vozilima koji se još nisu našli u listama potomka.

Programska implementacija križanja za prikaz rješenja dvjema listama:

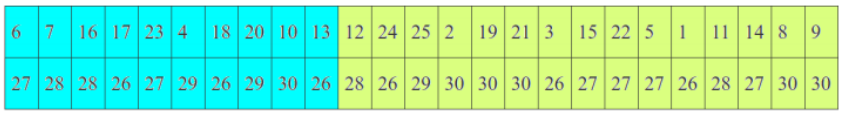
6	7	16	17	23	4	18	20	10	13	22	21	24	25	15	3	11	2	14	1	12	19	9	5	8
27	28	28	26	27	29	26	29	30	26	27	27	28	29	30	28	26	27	28	27	30	30	27	30	29

genotip prvog roditelja



12	16	24	20	25	2	7	19	23	21	10	13	17	18	3	15	4	6	22	5	1	11	14	8	9
28	29	26	28	29	30	26	30	30	30	30	28	30	30	26	27	27	30	27	27	26	28	27	30	30

genotip drugog roditelja



6	7	16	17	23	4	18	20	10	13	12	24	25	2	19	21	3	15	22	5	1	11	14	8	9
27	28	28	26	27	29	26	29	30	26	28	26	29	30	30	30	26	27	27	27	26	28	27	30	30

dijete roditelja

4.1.3. Prikaz rješenja mapom

Ovaj genotip predstavlja zapis rješenja problema strukturom mape kojoj su ključevi vozila, a vrijednost pridružena svakom ključu jest vektor klijenata.
Efektivno mapa predstavlja rješenje kao listu liste klijenata. Redoslijed klijenata u vektoru pod ključem vozila je ujedno i redoslijed obilaska klijenata u ruti vozila. Krajnje rješenje se sastoji od vozila koja imaju definirane rute i onih koja imaju prazne rute. Jedan klijent ne smije se naći u rutama dva različita vozila i svaki klijent može biti posjećen samo jedanput.

Tablica 4.1: Rješenje sa slike 4.1 prikazano mapom

ključ	vrijednost
V11	[4, 8, 2]
V12	[9]
V13	[7, 5, 10]
V14	[1, 6, 3]
V15	[]

Operator mutacije: Genom je predstavljen mapom, a operator mutacije nad mapom vrši lokalne promjene. Moguća su tri scenarija; klijenti se mijenjaju između vozila, jedan klijent se premješta drugom slučajno odabranom vozilu ili klijenti unutar jednog vozila mijenjaju redoslijed. Operator nasumično odabire dva vozila i ukoliko su različita nasumično bira po jednog klijenta u ruti svakog vozila i međusobno ih mijenja. Ukoliko je oba puta nasumice odabrano isto vozilo mutacija se vrši samo ako pod tim vozilom ima više od jednog klijenta, inače bira nova vozila za mutaciju.

Operator križanja: Operator križanja djeluje nad dva genotipa čija je struktura mapa — dva roditeljska i jedan genotip koji predstavlja budućeg potomka tih roditelja. Rekombinacija se obavlja na način da se lista klijenata prvog roditelja preslika u novu cjelobrojnu listu do slučajno odabranog indeksa. Nakon toga se redom iz liste klijenata drugog roditelja puni novostvorena lista pazeći pritom na moguće duplikate. Potom se u novostvorenoj listi klijenata odabire slučajan broj klijenata koji će se pridjeliti nekom vozilu sve dok se svi klijenti ne pridruže vozilima te tada operator vraća novi genotip nastao iz onih roditeljskih. Primjer križanja dva genotipa strukture mape od kojih je prvi roditelj definiran tablicom 4.1:

Tablica 4.2: Drugi roditeljski genotip

ključ	vrijednost
V26	[4, 9, 14, 1, 12]
V27	[6, 11, 25, 22, 24, 8]
V28	[18, 21, 16, 10, 17, 7, 2]
V29	[23, 5, 3, 15]
V30	[19, 20]

Slika 4.4: Operator križanja za prikaz rješenja mapom

Tablica 4.3: Genotip strukture mape dobiven operatorom križanja

ključ	vrijednost
V26	[17, 18, 13]
V27	[11, 6, 23, 22]
V28	[21, 4, 9, 14, 1, 12]
V29	[25, 24, 8, 16, 10, 7]
V30	[2, 5, 3, 15, 19, 20]

4.2. Evaluacija rješenja

Prije evaluacije rješenja funkcijom dobrote svi se prikazi rješenja svode na najadekvatniji od navedena tri zapisa za evaluaciju – permutacijski niz. Taj zapis odabran je zbog najjednostavnijeg dekodiranja rješenja.

U implementaciji problema usmjeravanja vozila korištena je funkcija dobrote koja minimizira ukupno prijeđenu udaljenost i vremensko trajanje posluživanja klijenata u rutama vozila. Uslijed mnogobrojnih operacija nad genotipima može se dogoditi da nas jedinke genetskog algoritma dovedu do neprihvatljivih rješenja. Zbog toga se kao usmjerivače algoritma uvode kazne kojima se rješenje kažnjava. Vrijednost funkcije dobrote u ovoj implementaciji uvećava se kaznom kako bi na kraju takvo rješenje algoritam odbacio.

Uobičajena funkcija dobrote za rješavanje problema usmjeravanja vozila genetskim algoritmom je:

$$f(x) = totaldistance(x) + \lambda \cdot overcapacity(x) + \mu \cdot overtime(x) \tag{4.1}$$

I prekomjerna potražnja klijenata za robom i dolazak prije početka vremenskog okvira definiranog za dostavljanje vraćaju količinu kapaciteta i vrijeme koje narušava zadana ograničenja. Ako nijedno od funkcijskih ograničenja nije narušeno, $f(x)$ vraća ukupnu prijeđenu udaljenost. U suprotnom i kapacitet i vrijeme množe se faktorima λ i μ i time penaliziraju vrijednost funkcije dobrote.

Za pronalaženje rješenja triju promatranih varijanti VRP-a korišten je izraz (4.1.) s time da se samo za VRPTW uzima overtime pošto su definirani vremenski okviri. Faktor λ iznosi 1000, a μ iznosi 1 za VRTPTW, odnosno 0 za CVRP i VRPPD. U slučaju da vozilo dođe nakon kraja zadanog vremenskog okvira rješenje postaje neprihvatljivo te se vrijednost funkcije dobrote kažnjava množenjem s brojem vozila.

Isječak koda pri kraju evaluacije:

```
if (input -> noTW) {
    fitness = totalDistance + 1000 * overCapacity;
} else {
    fitness = totalDistance + 1000 * overCapacity + overTime;
}
if (!isFeasible) {
    fitness *= input->getNumVehicles();
}
```

5. Analiza rezultata

Prije same analize rezultata različiti parametri vezani uz svaki od algoritama su bili optimirani. Variranjem parametara algoritama proučavao se utjecaj istih na konačnu dobrotu rješenja i takvim eksperimentiranjem su utvrđene optimalne vrijednosti parametara i koji je od tri prikaza rješenja najbolji za pojedine instance problema.
Reprezentativni primjerci koji su korišteni prilikom evaluacija rješenja su odabrani iz najpoznatijih testnih primjeraka (engl. *benchmark*) za promatrane inačice problema – *Solomon Benchmark* [7] za VRPTW i CVRP te *Li and Lim Benchmark* [5] za VRPPD. Iz *Solomonovih* testnih primjeraka uzeti su primjerci s različitim brojem klijenata, 25 i 50, a iz *Li and Lim* testnih primjeraka uzeta je instanca problema sa 100 klijenata.

5.1. Korišteni algoritmi

Algoritmi korišteni prilikom evaluiranja rješenja su *SteadyStateTournament* i *Evolution Strategy*. Ovi algoritmi su implementirani u sustavu ECF *[3]* te su odabrani iz razloga što su neovisni o vrsti genotipa, odnosno primjenjivi su na sva tri prikaza rješenja. Pozivanje pojedinog algoritma definira se u parametarskoj datoteci gdje se podešavaju i vrijednosti njima svojstvenih parametara.

5.1.1. SteadyStateTournament

Algoritam SteadyStateTournament je elitistički i obavlja takozvanu turnirsku selekciju. Definiran je parametrom $tsize$ koji predstavlja broj jedinki koje ulaze u algoritam selekcije. Od $tsize$ jedinki turnira odabire onu najgoru i mijenja je potomkom dvoje slučajno odabranih roditelja koji se također nalaze u turniru. U ovom završnom radu turnir se sastojao od triju jedinki, odnosno provedena je 3-turnirska selekcija.
Pseudokod algoritma nad jednom generacijom dan je u nastavku:

```
ponavljaj za veličinu populacije {
    dodaj tsize slučajnih jedinki u turnir;
    odaberi najgoru jedinku u turniru;
    roditelji = odaberi dvije slučajne jedinke u turniru;
    dijete = križaj(roditelji);
    mutiraj(dijete);
}
```

Slika 5.1: SteadyStateTournament pseudokod *[3]*

5.1.2. Evolution Strategy

Algoritam implementira evolucijsku strategiju ($\mu/\rho +/, \lambda$) čiji su parametri:

- μ: veličina populacije roditelja
- λ: broj potomaka stvorenih u svakoj generaciji
- ρ: broj roditelja koji se koriste pri izradi potomaka
- selekcija: $+/,$

Roditelji se odabiru deterministički dvjema strategijama, zarez-strategijom (engl. *comma-selection*) ili plus-strategijom (engl. *plus-selection*). Zarez-strategija je strategija kod koje se roditelji odabiru iz skupa λ potomaka, a kod plus-strategije selekcija za novu populaciju se vrši nad zajedničkim skupom genotipa roditelja (μ) i potomaka (λ). Plus-strategija ima elitističko svojstvo pa se najbolja jedinka roditelja pojavljuje i u novoj populaciji. U slučaju plus-strategije od μ roditelja stvara se λ potomaka koji se dodaju u skup roditelja te se iz tog ažuriranog skupa bira najboljih μ jedinki za novu populaciju roditelja. Parametar ρ može imati dvije vrijednosti – 1 ili 2, a njima se označava koliko roditeljskih genotipa sudjeluje u stvaranju potomka. U prvom slučaju potomak će biti mutacija jednog roditelja, a u drugom slučaju potomak će nastati križanjem dvoje roditelja.

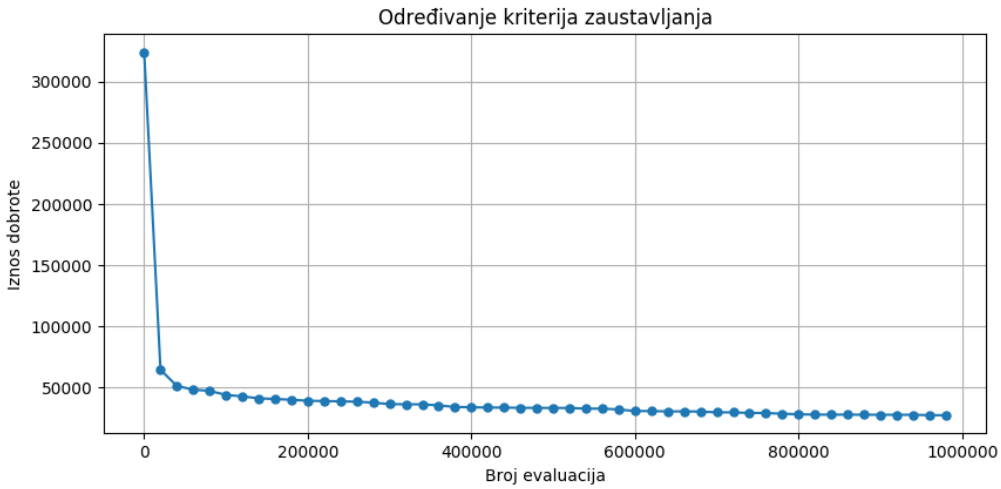
Pseudokod algoritma evolucijske strategije ($\mu/\rho +/, \lambda$) nad jednom generacijom:

```
ponavljaj za svaku podpopulaciju {
    dodaj μ jedinki u roditelje;
    stvori λ potomaka koristeći slučajnih ϱ roditelja;
    ako zarez-strategija {
        roditelji = μ najboljih iz skupa potomaka;
    } inače {
        roditelji = μ najboljih iz skupa potomaka i roditelja;
    }
}
```

Slika 5.2: Evolution Strategy pseudokod *[3]*

5.2. Kriterij zaustavljanja

Kako bi se izvođenje algoritma ubrzalo uvodi se kriterij zaustavljanja, odnosno broj evaluacija pri kojem dobivamo zadovoljavajuće rezultate. Proučavanjem prosjeka kvalitete rješenja određene funkcijom dobrote definiran je kriterij zaustavljanja, evaluacija u kojoj dobrota rješenja prestaje s poboljšanjem i počinje konvergirati. Na slici 5.3 prikazan je napredak kvalitete rješenja za reprezentativni primjerak VRPPD problema za 100 klijenata. Iz grafa je vidljivo da poslije naglog napretka vrijednosti funkcije dobrote, otprilike nakon 300000 evaluacija poboljšanja stagniraju te se zato taj broj uzima kao kriterij zaustavljanja. Eventualno bi se većim brojem evaluacija proporcionalno dospjelo i do još boljih rješenja, međutim za potrebe optimiranja parametara i analize općenito proučava se samo ponašanje funkcije dobrote kroz evaluacije.



Slika 5.3: Poboljšanje dobrote rješenja kroz evaluacije

5.3. Rezultati u statičkim uvjetima

U statičkim uvjetima rješenja svih triju instanci su generirana pomoću već spomenutih algoritama uz variranje njima svojstvenih parametara te parametra veličine populacije. Na svim *boxplot* grafovima prikazana je vrijednost funkcije dobrote u ovisnosti o veličini populacije. Vrijednosti su dobivene statistički iz dvadeset pokretanja za četiri različite veličine populacije.

5.3.1. Veličina populacije

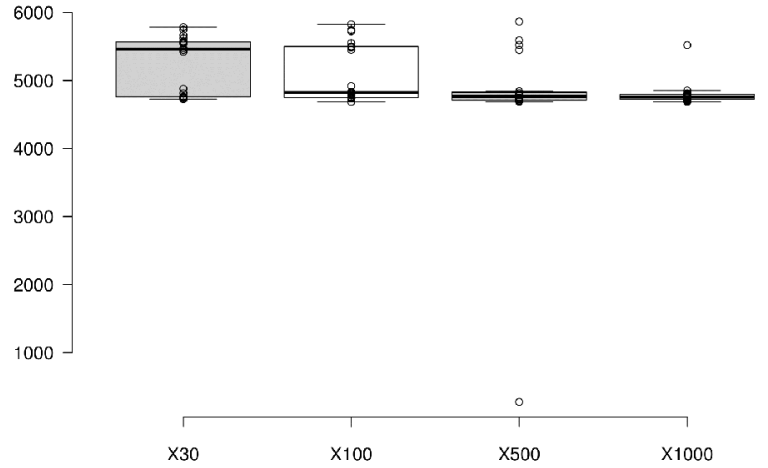
Parametar veličine populacije je od iznimne važnosti u populacijskim evolucijskim algoritmima. Prilikom generiranja rješenja korištene su četiri veličine populacije – $30, 100, 500, 1000$. Zbog definiranog kriterija zaustavljanja, odnosno fiksiranog broja evaluacija pri kojem funkcija dobrote poprima najveće vrijednosti najbolji rezultati su u više slučajeva dobiveni algoritmima s manjim populacijama. Primjerice kad se broj jedinki (populacija) poveća na 1000 tada se unutar jedne generacije napravi i toliko više evaluacija, a zbog ograničenog broja evaluacija posljedično algoritam prolazi kroz manje generacija te je širina prostora pretraživanja manja.

5.3.2. Rezultati dobiveni algoritmom SteadyStateTournament

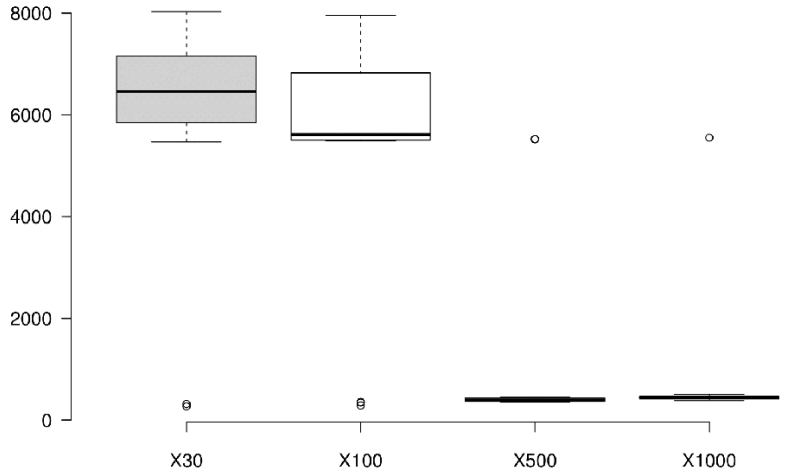
Prilikom generiranja rješenja algoritmom SteadyStateTournament korištene su dvije vrijednosti parametra vjerojatnosti mutacije – sustavom zadana vjerojatnost 0.3 i dvostruko veća 0.6. Parametar algoritma $tsize$ u svim evaluacijama iznosio je 3, odnosno vršila se selekcija između triju jedinki u svakoj iteraciji algoritma.

Vjerojatnost mutacije 0.3

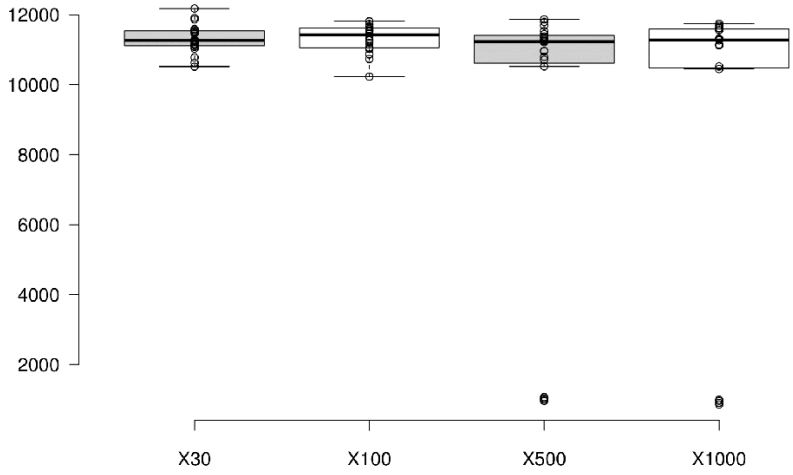
Za kapacitivni problem usmjeravanja vozila za 25 klijenata algoritam SteadyStateTournament s parametrom vjerojatnosti mutacije 0.3 najbolje rezultate dao je uz prikaz rješenja dvjema listama vozila što je vidljivo na slikama 5.4 – 5.6 gdje *boxplot* opisuje kvalitetu rješenja pojedinog prikaza rješenja ovisno o veličini populacije. Vidljivo je da je u prikazu rješenja permutacijskim nizom i mapom algoritam kroz generacije ostao vrlo blizu početnog rješenja dok se u prikazu dvjema listama algoritam uspio odmaknuti iz tog područja pretrage i pronaći vrlo dobra rješenja uz veću veličinu populacije.



Slika 5.4: CVRP 25 klijenata - rješenje permutacijskim nizom

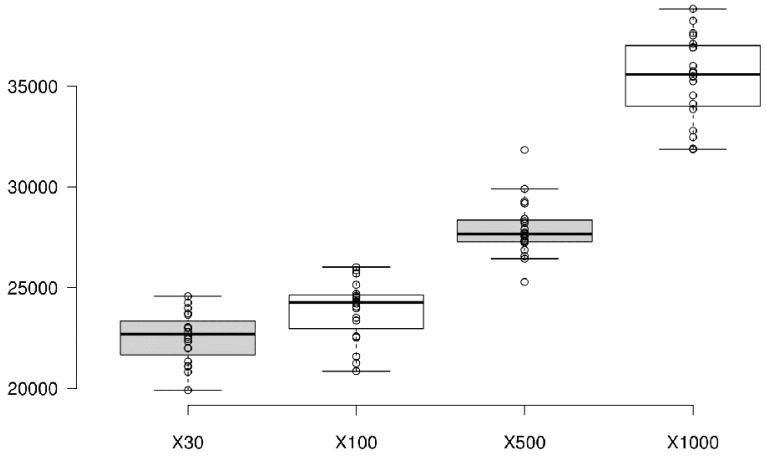


Slika 5.5: CVRP 25 klijenata - rješenje dvjema listama

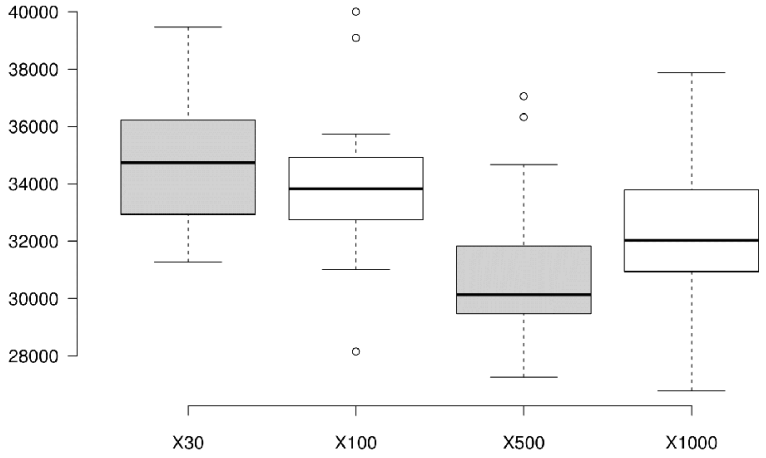


Slika 5.6: CVRP 25 klijenata -rješenje mapom

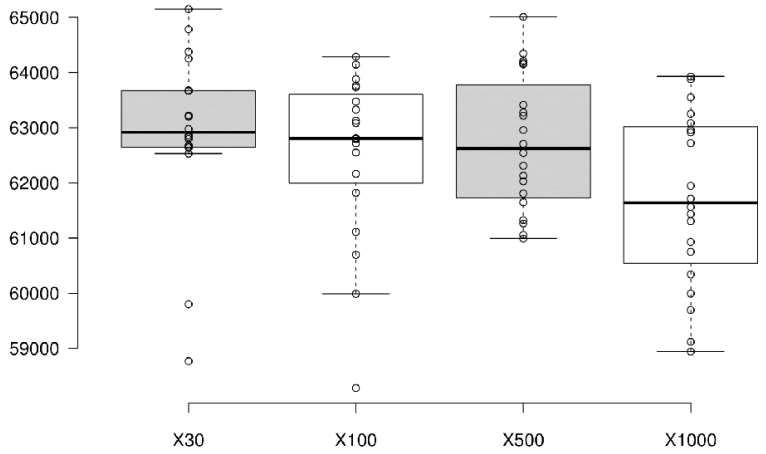
Za CVRP s 50 klijenata, najboljim se pokazao prikaz permutacijskim nizom:



Slika 5.7: CVRP 50 klijenata - rješenje permutacijskim nizom

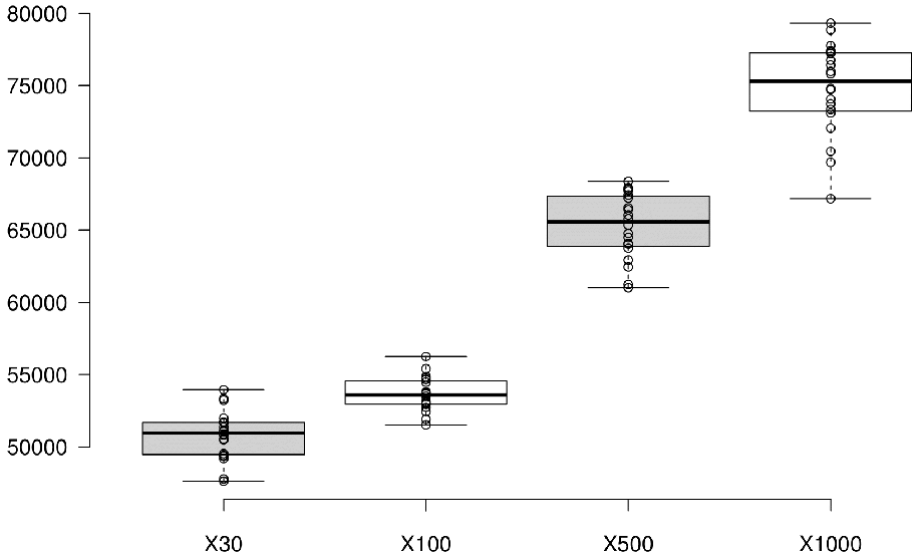


Slika 5.8: CVRP 50 klijenata - rješenje dvjema listama

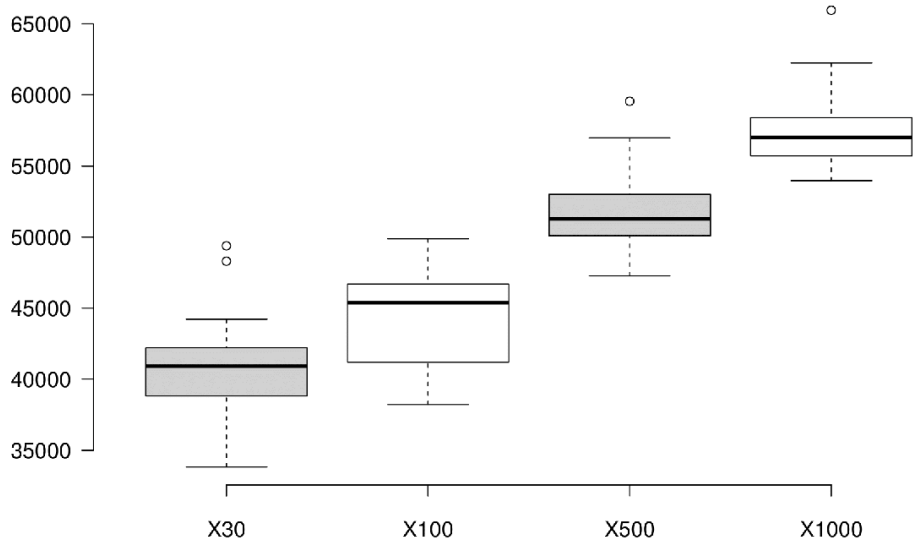


Slika 5.9: CVRP 50 klijenata - rješenje mapom

I za instance problema VRPTW i VRPPD kao najbolji prikaz rješenja pokazao se permutacijski niz uz veličinu populacije 30:



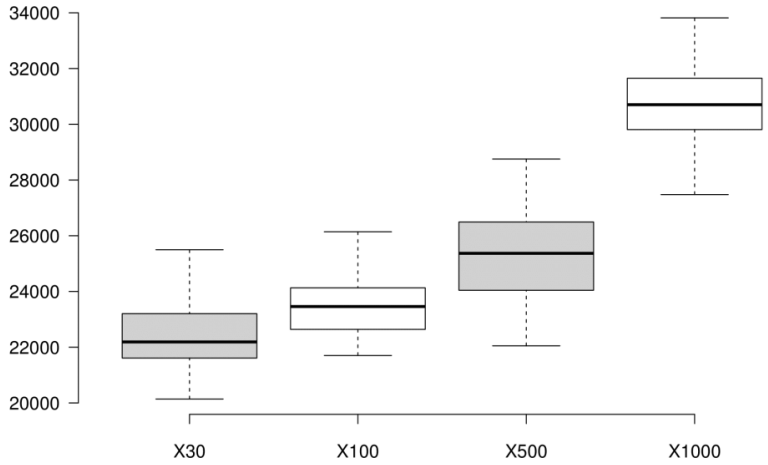
Slika 5.10: VRPTW 50 klijenata - rješenje permutacijskim nizom



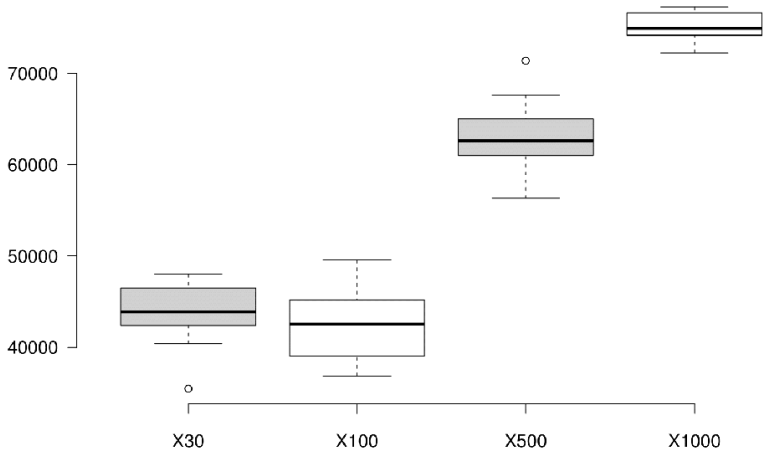
Slika 5.11: VRPPD 100 klijenata - rješenje permutacijskim nizom

Vjerojatnost mutacije 0.6

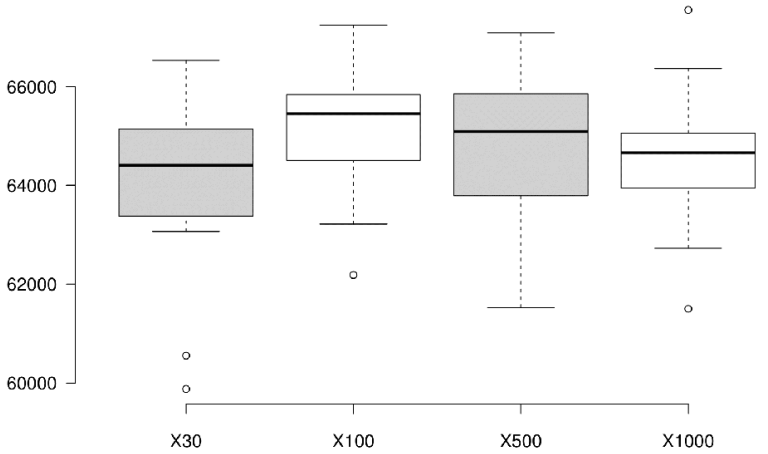
Rezultati dobiveni povećanjem vrijednosti parametra vjerojatnosti mutacije na 0.6 nisu donijeli znakovite promjene kvalitete rješenja. Prikaz rješenja dvjema listama te prikaz rješenja mapom daju približno jednake ili lošije rezultate, no prikaz permutacijskim nizom daje nešto bolje rezultate u odnosu na one gdje je vjerojatnost mutacije bila manja, posebice pri većim veličinama populacije.



Slika 5.12: CVRP 50 klijenata - rješenje permutacijskim nizom uz vjerojatnost mutacije 0.6

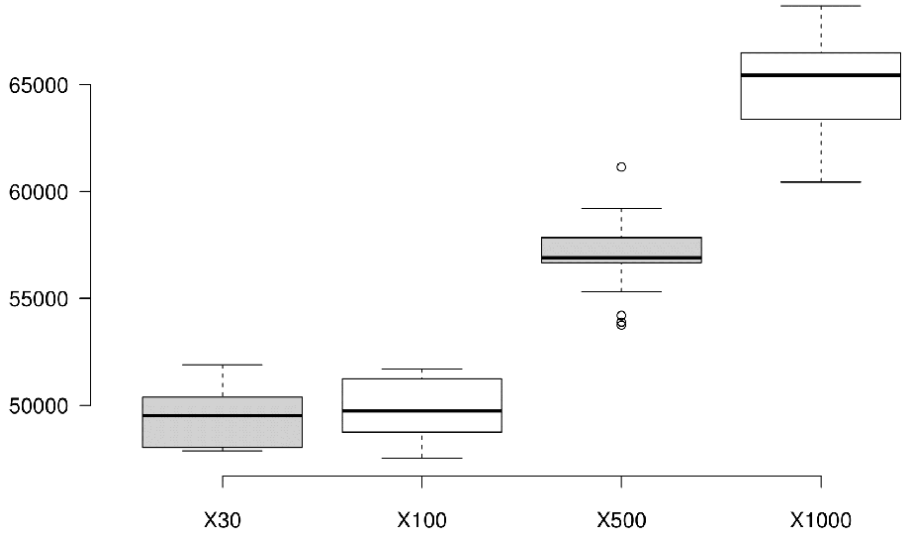


Slika 5.13: CVRP 50 klijenata - rješenje dvjema listama uz vjerojatnost mutacije 0.6

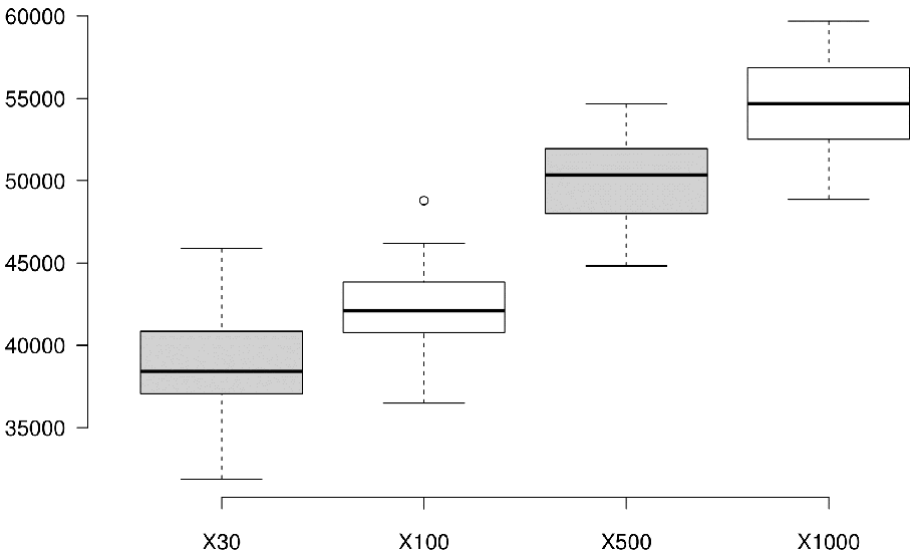


Slika 5.14: CVRP 50 klijenata - rješenje mapom uz vjerojatnost mutacije 0.6

Također, za instance problema VRPTW i VRPPD korisnost povećanja mutacije pokazala se samo pri prikazu permutacijskim nizom. U odnosu na rezultate dobivene s manjom vjerojatnošću mutiranja jedinke medijan vrijednosti funkcije dobrote je niži za svaku veličinu populacije.



Slika 5.15: VRPTW 50 klijenata - rješenje permutacijskim nizom uz vjerojatnost mutacije 0.6



Slika 5.16: VRPPD 100 klijenata - rješenje permutacijskim nizom uz vjerojatnost mutacije 0.6

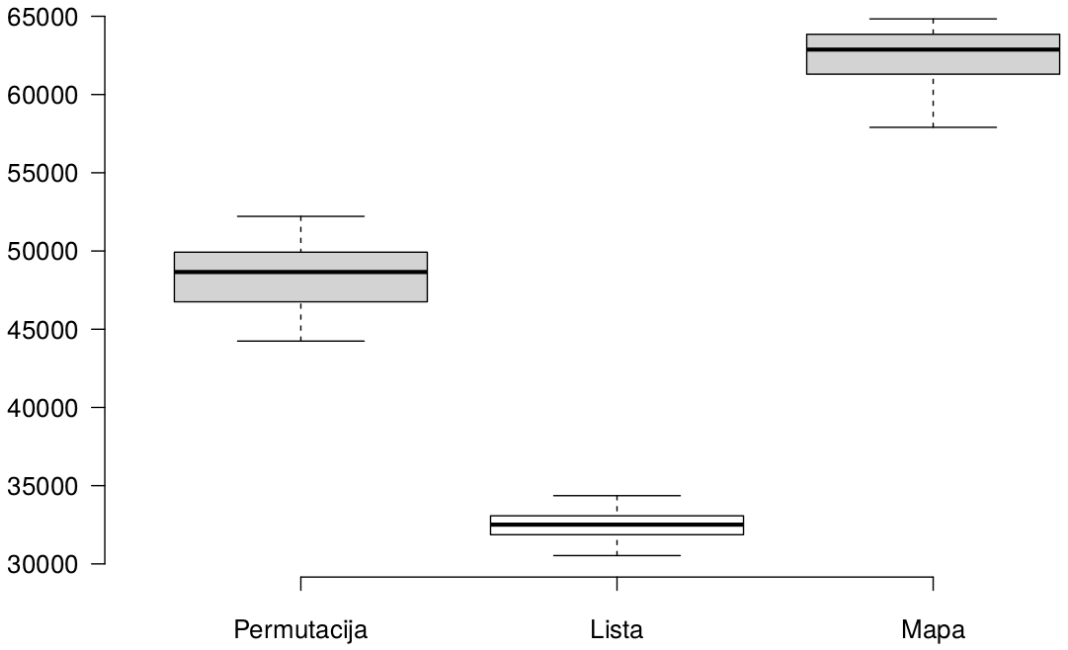
Najboljim odabirom pri vjerojatnosti mutacije 0.6 pokazao se permutacijski niz uz veličinu populacije 30.

5.3.3. Rezultati dobiveni algoritmom Evolution Strategy

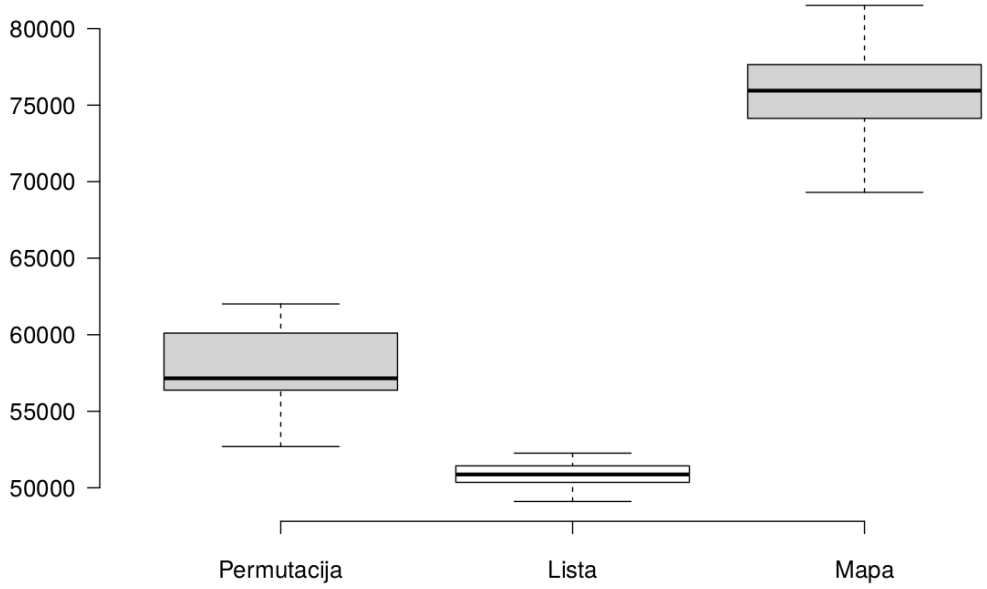
Rezultati su dobiveni za dvije različite vrijednosti parametra ρ. U prvom slučaju ρ je iznosio 1 pa se koristio samo jedan roditelj, odnosno vršila se mutacija nad istim prilikom izrade novih potomaka, a u drugom slučaju ρ je iznosio 2 te su slučajno odabrana dva roditelja operatorom križanja generirala nove potomke. Korištena je plus-strategija, $\lambda = 4$, $\mu = 1$ za $\rho = 1$, odnosno 2 za $\rho = 2$.

Parametar $\rho = 1$

U slučaju korištenja jednog roditelja prilikom stvaranja novih genotipa, najboljim prikazom rješenja, za razliku od rezultata dobivenih algoritmom SteadyStateTournament, u većini slučajeva pokazao se prikaz dvjema listama.
Za CVRP i VRPTW probleme 50 klijenata uz populaciju 30 kvaliteta rješenja u ovisnosti o prikazu vidljiva je na slikama 5.17 i 5.18:

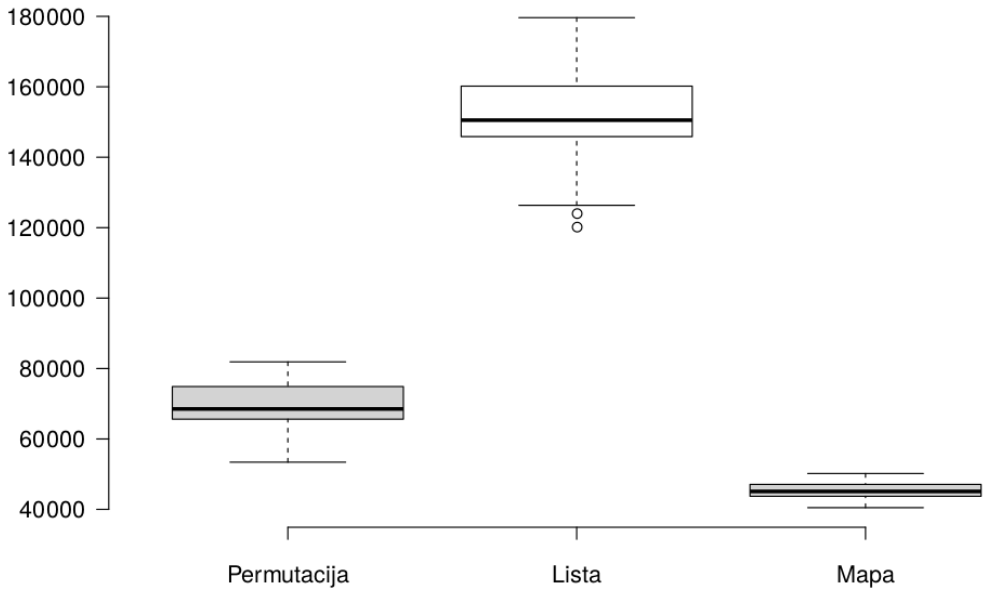


Slika 5.17: CVRP 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30



Slika 5.18: VRPTW 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30

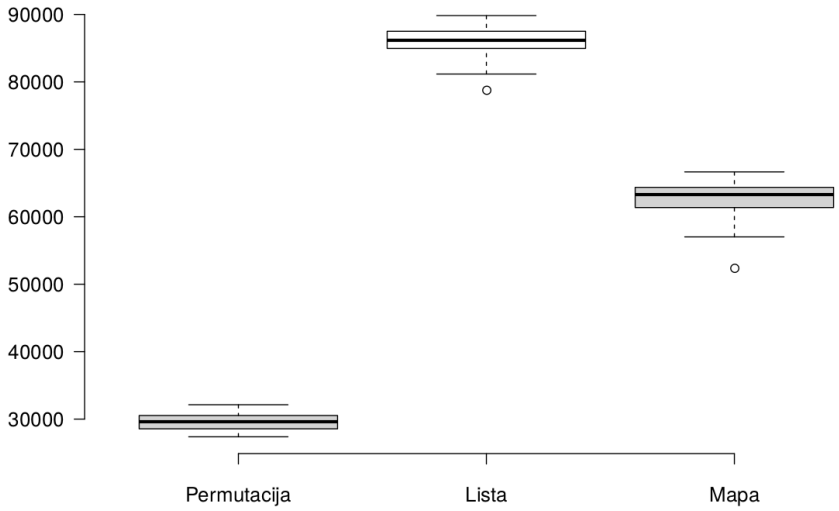
Zanimljivo je da su rezultati VRPPD problema najbolji korištenjem mape za prikaz rješenja. VRPPD problem 100 klijenata uz veličinu populacije 30 dan je na slici 5.19:



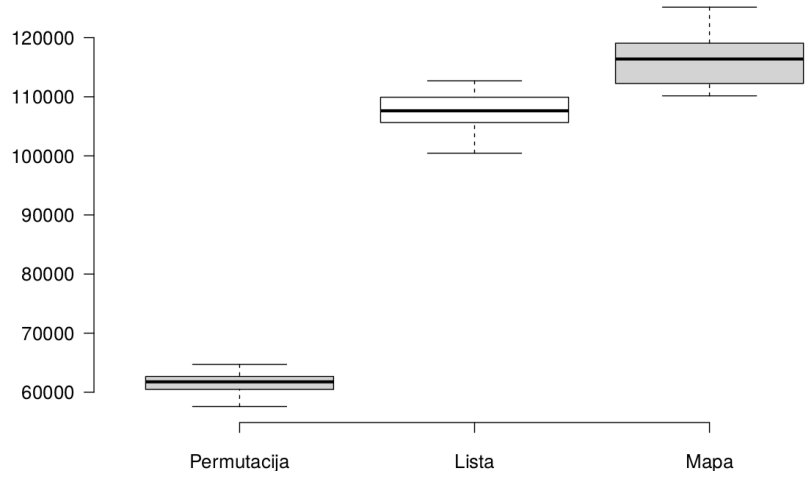
Slika 5.19: VRPPD 100 klijenata - usporedba prikaza rješenja uz veličinu populacije 30

Parametar $\rho = 2$

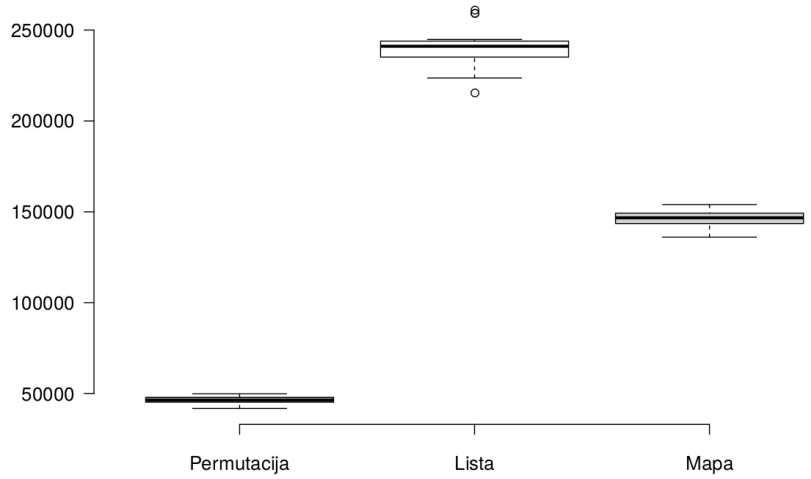
Pri korištenju dva roditelja pri izradi potomaka u sve tri varijante problema najboljim se opet pokazao prikaz rješenja permutacijskim nizom uz veličinu populacije 30.



Slika 5.20: CVRP 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30



Slika 5.21: VRPTW 50 klijenata - usporedba prikaza rješenja uz veličinu populacije 30



Slika 5.22: VRPPD 100 klijenata - usporedba prikaza rješenja uz veličinu populacije 30

5.3.4. Najbolje kombinacije parametara

U skladu s dobivenim rezultatima može se odrediti najbolja kombinacija parametara za pojedine instance problema. Parametri su navedeni u tablici:

Tablica 5.1: Tablica najboljih parametara

	CVRP	VRPTW	VRPPD
BROJ KLIJENATA	50	25, 50	100
PRIKAZ RJEŠENJA	PERMUTACIJA	PERMUTACIJA	PERMUTACIJA
ALGORITAM	SST	SST	SST
VJEROJATNOST MUTACIJE	0.6	0.6	0.6
BROJ RODITELJA ρ	–	–	–
VELIČINA POPULACIJE	30	30	30

5.4. Rezultati u dinamičkim uvjetima

Kao dinamička komponenta problema u dva scenarija promatrana su vremena potrebna za generiranje rješenja reprezentativnih primjeraka VRPTW i VRPPD problema.
Jedna analiza vezana je uz vrijeme potrebno do pronalaska rješenja koje je jednako dobro ili bolje od heuristički dobivenog rješenja u kontekstu ukupno prevaljenog puta svih vozila, a u drugom slučaju su za problem s izrazito velikim brojem klijenata uspoređivana vremena potrebna za generiranje rješenja različitim prikazima.

5.4.1. Vrijeme do pronalaska heuristički dobivenih rješenja

Analiza se vršila nad tri reprezentativna primjerka – VRPTW za 25 i 50 klijenata iz *Solomonovih* instanci problema te VRPPD za 100 klijenata iz *Li and Lim* testnih primjeraka. Heuristički dobivena rješenja za ove probleme u kontekstu prijeđenog puta su dana u tablici:

Tablica 5.2: Instance problema sa zadanim rješenjima dobivenim heuristikama

VRPTW 25	VRPTW 50	VRPPD 100
370,231	811,435	1222,86

Ovom analizom htjelo se ispitati kojim prikazom i kojim algoritmom rješenja najbrže konvergiraju danim heuristički dobivenim rješenjima. Rezultati ovog ispitivanja su u skladu s prethodno ispitanim kvalitetama rješenja ovisno o prikazu i algoritmu. U tablicama znak '/' predstavlja slučaj u kojemu je za taj prikaz određeni algoritam zapeo u lokalnom optimumu iz kojeg nije mogao izaći kroz trideset minuta. U tablicama su zapisana prosječna vremena u sekundama $[s]$ iz dvadeset pokušaja pronalaska zadanog rješenja. Crvenom bojom su u sve tri tablice označena vremena dobivanja danog rješenja za kombinaciju parametara koja daje najbolje rezultate

Za primjerak problema VRPTW s 25 klijenata (tablica 5.3) vrijeme pronalaska optimalnog rješenja algoritmom SteadyStateTournament je toliko malo da bi se trebalo izraziti u milisekundama (u tablici označeno s ~$0s$). Algoritam EvolutionStrategy uz korištenje mutacije jednog roditeljskog genotipa za sva tri prikaza je bio uspješan, dok je uz korištenje križanja dvoje roditelja za generiranje novih potomaka bio neuspješan koristeći prikaz rješenja dvjema listama.

Tablica 5.3: Vremena konvergencije ka rješenju za VRPTW problem 25 klijenata

	PERMUTACIJA	DVIJE LISTE	MAPA
SST 0.3	~$0s$	~$0s$	~$0s$
SST 0.6	~$0s$	~$0s$	~$0s$
ESM	~$0s$	$0.2s$	$12.9s$
ESX	~$0s$	/	$0.3s$

Za VRPTW instancu s 50 klijenata najboljim se pokazao prikaz permutacijskim nizom dok je nešto sporije ka rješenju konvergirao algoritam koristeći prikaz dvjema listama. Niti u jednoj kombinaciji s prikazom rješenja mapom algoritmi nisu uspjeli naći zadano rješenje.

Tablica 5.4: Vremena konvergencije ka rješenju za VRPTW problem 50 klijenata

	PERMUTACIJA	DVIJE LISTE	MAPA
SST 0.3	0.2	$3.3s$	/
SST 0.6	$0.2s$	$1.5s$	/
ESM	/	$2.4s$	/
ESX	$1s$	/	/

Pronalazak zadanog rješenja inačice VRPPD za 100 klijenata bio je najzahtjevniji, no za svaki algoritam pronađen je prikaz rješenja koji ipak nalazi optimalno rješenje u realnom vremenu. Proučavanjem *boxplot* grafova za promatranu instancu problema dobivene vrijednosti i scenariji su očekivani.

Tablica 5.5: Vremena konvergencije ka rješenju za VRPPD problem 100 klijenata

	PERMUTACIJA	DVIJE LISTE	MAPA
SST 0.3	$2.3s$	/	/
SST 0.6	$3.1s$	/	/
ESM	/	/	$38.9s$
ESX	$8.8s$	/	/

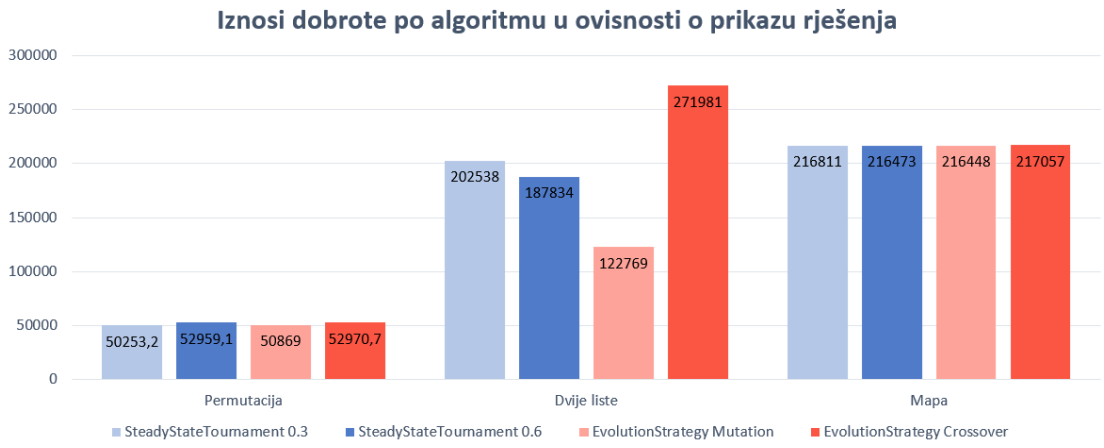
5.4.2. Usporedba vremena ovisno o prikazu

Prilikom ove analize korištena je instanca problema VRPTW za 1000 klijenata. Budući da je ovaj primjerak problema realniji u stvarnom svijetu svrha ove analize bila je promotriti koji prikaz *najbrže* pronalazi rješenje obzirom da je vrijeme posluživanja pokazatelj kvalitete nekog transportnog poduzeća. Za različite algoritme i parametre korištene prilikom generiranja rješenja u statičkim uvjetima jednom je pokrenuta evaluacija milijun jedinki za navedeni problem uz veličinu populacije 30. U tablici 5.7 su zapisana vremena potrebna za pronalazak rješenja ovisno o kombinaciji algoritma, parametara te prikaza rješenja.

Tablica 5.6: Vremena generiranja rješenja za VRPTW problem 1000 klijenata

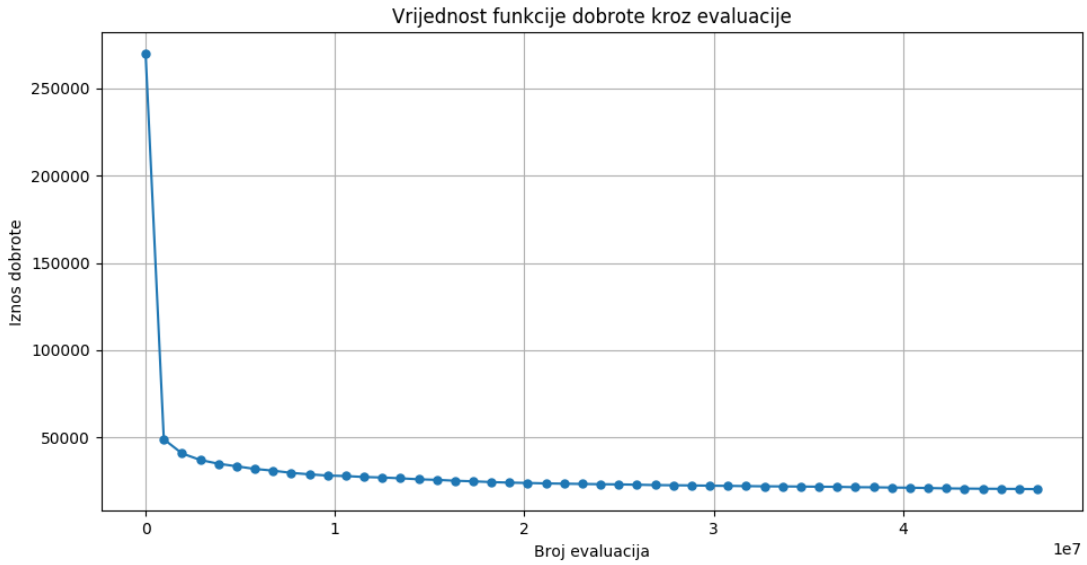
	PERMUTACIJA	DVIJE LISTE	MAPA
SST 0.3	$2h\ 58min$	$1h\ 54min$	$7h\ 19min$
SST 0.6	$3h\ 6min$	$1h\ 55min$	$7h\ 21min$
ESM	$2h\ 52min$	$1h\ 11min$	$7h\ 19min$
ESX	$3h\ 42min$	$1h\ 55min$	$7h\ 21min$

Kao što se moglo očekivati, vrijeme generiranja rješenja za prikaz rješenja mapom je znatno dulje od vremena potrebnih za generiranje rješenja prilikom korištenja jednostavnijih struktura. Nakon dobivenih vremena ovisno o prikazu rješenja htjelo se utvrditi koliko su $kvalitetna$ dobivena rješenja. Na slici 5.23 mogu se očitati iznosi dobrote rješenja generiranih kroz vremena iz tablice 5.7:



Slika 5.23: Iznos dobrote kroz evaluacije

Na kraju ispitano je vrijeme do pronalaska rješenja s iznosom dobrote $20121,9$ koje je unutar samo nekoliko minuta dobiveno genetskim programiranjem. Za navedenu instancu problema, uz "optimalne" parametre dobivene analizom rezultata u ovom radu, vrijeme pronalaska rješenja bilo je 609608 sekundi, odnosno 7 dana, 1 sat i 20 minuta. Na slici 5.24 vidljivo je kretanje vrijednosti funkcije dobrote kroz evaluacije za ovo testiranje.



Slika 5.24: Iznos dobrote kroz evaluacije

6. Zaključak

Ovaj završni rad bavi se rješavanjem problema usmjeravanja vozila. U radu se metaheurističkom metodom genetskog algoritma pristupilo rješavanju triju varijanti problema. Uz to promotrio se utjecaj strukturnog zapisa rješenja na konačnu kvalitetu rješenja.
U statičkim uvjetima koristeći algoritme turnirske selekcije te evolucijske strategije generirana su rješenja za nekoliko reprezentativnih primjeraka. Analizom dobivenih rješenja se iz istih dalo zaključiti da je programsko ostvarenje genetskog algoritma vrlo dobar pristup rješavanju ove vrste problema. Temeljem statistika kreirani su *boxplot* grafovi iz kojih se jasno mogao vidjeti utjecaj prikaza rješenja na kvalitetu krajnjeg rješenja. Od prikaza rješenja u većini slučajeva najboljim se pokazao prikaz permutacijskim nizom, međutim za neke instance problema uz specifične vrijednosti veličine populacije, vjerojatnosti mutacije i ostalih algoritmima svojstvenih parametara su se prikaz dvjema listama kao i prikaz mapom pokazali izuzetno dobrim odabirom.
U radu se dinamička komponenta problema promatrala kroz vrijeme potrebno za pronalazak rješenja VRPTW i VRPPD problema. Prvo je proučavano vrijeme nalaženja heuristički dobivenih rješenja u kontekstu prevaljenog puta te se i u ovoj analizi kao najbolji nametnuo prikaz permutacijskim nizom koji je u usporedbi s ostalim prikazima rješenja najbrže nalazio optimalno rješenje.
Također, razmatralo se vrijeme potrebno za generiranje rješenja zahtjevnije instance problema ovisno o prikazu te je i u ovom nadmetanju permutacijski niz odnio pobjedu obzirom na kvalitetu rješenja u potrebnom vremenu.
Cilj određivanja optimalnog prikaza rješenja je naoko ostvaren budući da se prikaz permutacijskim nizom u većini slučajeva pokazao najefikasnijim, no zbog raznolikosti inačica problema odgovor na pitanje koji je najbolji prikaz je potrebno naći za svaki slučaj zasebno.
Kao daljnji rad na ovom problemu vidim proučavanje kvalitete rješenja eksperimentiranjem vrijednosti parametara algoritama i onih općenitih u svrhu optimiranja parametara. Osim toga za budući rad previđam istraživanje i rješavanje problema nekim drugim metaheurističkim metodama.

LITERATURA

[1] G. Clarke i J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.*, 12(4):568–581, Kolovoz 1964. ISSN 0030-364X. doi: 10.1287/opre.12.4.568. URL http://dx.doi.org/10.1287/opre.12.4.568.

[2] G. B. Dantzig i J. H. Ramser. The truck dispatching problem. *Manage. Sci.*, 6 (1):80–91, Listopad 1959. ISSN 0025-1909. doi: 10.1287/mnsc.6.1.80. URL http://dx.doi.org/10.1287/mnsc.6.1.80.

[3] D. Jakobović et al. www.ecf.zemris.fer.hr/algorithms.html.

[4] M. L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[5] A. Li H., Lim. www.sintef.no/projectweb/top/pdptw/li-lim-benchmark.

[6] Anamari Nakić i Mario-Osvin Pavčević. *Matematika 3: uvod u teoriju grafova*. Zagreb: Element, 2014.

[7] M.M. Solomon. www.sintef.no/projectweb/top/vrptw/solomon-benchmark.

[8] Marko Čupić. *Prirodom inspirirani optimizacijski algoritmi*. 2009.

Rješavanje problema usmjeravanja vozila metaheuristikama u statičkim i dinamičkim uvjetima

Sažetak

U ovom radu opisan je i obrađen problem usmjeravanja vozila i rješavanje tri inačice tog problema metaheurističkim pristupom, konkretno genetskim algoritmom. Navedene su postojeće metode rješavanja problema i ostvarena je vlastita implementacija rješenja u tri oblika zapisa u statičkim uvjetima. Kao dinamička komponenta za različite genotipe i inačice problema ispitana je brzina konvergencije rješenja ka rješenjima dobivenim heurističkim pristupom kao i brzina pronalaska rješenja za fiksni broj evaluacija. Rezultati su uspoređeni međusobno te prikazani grafički.

Ključne riječi: problem usmjeravanja vozila, VRP, CRVP, VRPTW, VRPPD, statički uvjeti, dinamički uvjeti, metaheuristika, optimizacija, genetski algoritam

Solving vehicle routing problem using metaheuristics in static and dynamic conditions

Abstract

In this thesis, vehicle routing problem and solving three variants of it is described. Problem variants were solved using metaheuristic approach, specifically genetic algorithm. The existing methods of solving the VRP problem were mentioned as well as the solution implementation that was recorded in three forms in static conditions. As a dynamic component for different genotypes and variants of the problem, convergence speed to the solutions obtained by heuristic approach was examined as well as the speed of finding a solution for a fixed number of evaluations. The results were compared and presented graphically.

Keywords: vehicle routing problem, VRP, CVRP, VRPTW, VRPPD, static conditions, dynamic conditions, metaheuristic, optimisation, genetic algorithm