OPTIMIZACIJA POSLOVANJA DOSTAVNE SLUŽBE KORIŠTENJEM GENETSKIH ALGORITAMA

Šaban, Martin

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

OPTIMIZACIJA POSLOVANJA DOSTAVNE SLUŽBE KORIŠTENJEM GENETSKIH ALGORITAMA

Martin Šaban

Zagreb, veljača 2020.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, veljača 2020.

Predgovor

Zahvaljujem se mentoru, profesoru Kučak Danijelu, na pomoći i savjetima tijekom izrade ovog rada.

Zahvaljujem se svim profesorima i asistentima od kojih sam imao priliku učiti, te također svima koji su mi omogućili uspješno studiranje, a posebno svojoj obitelji.

Sažetak

U ovom radu opisane su osnove genetskih algoritama, njihovi sastavni dijelovi, te način njihovog funkcioniranja. Objašnjena je podjela evolucijskog računarstva, grane umjetne inteligencije u koju genetski algoritmi spadaju. Zatim je opisan problem usmjeravanja vozila, te je ostvaren i opisan genetski algoritam za rješavanje spomenutog problema. Provedeno je testiranje s različitim parametrima algoritma, te je prikazana analiza rezultata.

Ključne riječi: evolucijsko računarstvo, genetski algoritmi, problem usmjeravanja vozila

Abstract

In this document basics of genetic algorithms are described, their integral components, and the way they work. Evolutionary computation is described, a branch of artificial intelligence to which genetic algorithms belong, Next, vehicle routing problem is described, and the genetic algorithm for solving the given problem is implemented and described. Testing with different algorithm parameters was performed and an analysis of the results is presented.

Keywords: evolutionary computation, genetic algorithms, vehicle routing problem

Sadržaj

1. Uvod

Mnoge tvrtke susreću se s problemom optimiziranja prijevoza, kao što su npr. tvrtke za dostavu, autoprijevoznici, pošta, tvrtke za zbrinjavanje otpada te mnogi drugi.
Te je jedan od njihovih najvećih problema organizirati prijevoz na način da maksimalno smanje troškove prijevoza, odnosno optimizacija spomenutih troškova. Taj problem je opće poznat kao problem usmjeravanja vozila (engl. *Vehicle Routing Problem*, skraćeno *VRP*).

Ideja ovog rada je da se uz pomoć genetskih algoritama osmisli programsko ostvarenje koje će za određeni broj lokacija i vozila isporučiti optimalne rute prema kojima će dostavna služba moći dostaviti svoju robu, a sve u svrhu smanjenja troškova prijevoza.
U radu će također biti pobliže objašnjeni genetski algoritmi kao metoda kojom će se pokušati riješiti zadani problem.

Rad se sastoji od pet poglavlja, nakon ovog uvodnog u drugom poglavlju je dan kratak opis evolucijskog računarstva, grane umjetne inteligencije, te njihova podjela, koja kao sastavni dio sadrži i genetske algoritme.

U sljedećem, trećem poglavlju, opisani su genetski algoritmi, njihove osnove, sastavni dijelovi te način njihovog funkcioniranja.

U četvrtom poglavlju opisan je problem koji će se pokušati riješiti u ovome radu, te je opisana implementacija sustava, odnosno programsko rješenje koje je osmišljeno u sklopu ovog rada.

U zadnjem, petom, poglavlju opisani su testni podaci, te je provedena i opisana analiza utjecaja parametara na performanse algoritma.

2. Evolucijsko računarstvo

Evolucijsko računarstvo je grana umjetne inteligencije koja se sastoji od algoritama inspiriranih biološkom evolucijom, a koji se koriste za rješavanje optimizacijskih problema [3].

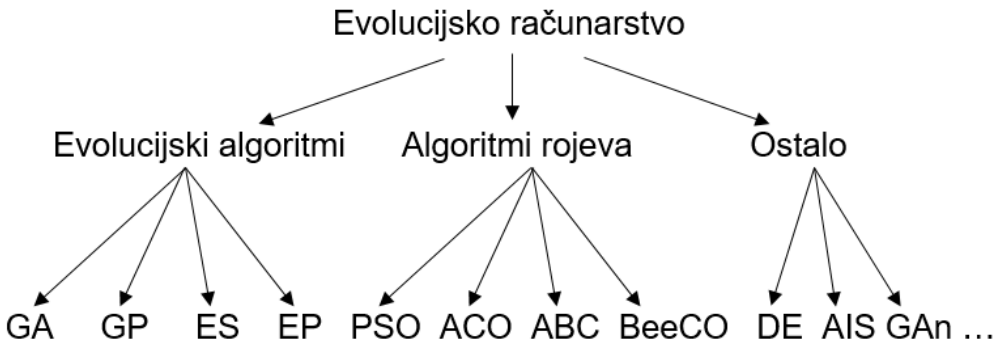
Tehnički gledano to su algoritmi za rješavanje optimizacijskih problema zasnovani na populaciji te dolasku do rješenja metodom pokušaja i pogrešaka.

S obzirom da ne pretražuju cijeli prostor rješenja, nego se heuristikom usmjeravaju, nema garancije da će doći do optimalnog rješenja, a i u većini problema koje ti algoritmi rješavaju zadovoljavajuće je i rješenje koje je blizu optimalnog.

Evolucijsko računarstvo dijeli se na tri grane:

- Evolucijske algoritme
- Algoritme rojeva
- Ostale algoritme

Ova podjela te odabrani algoritmi prikazani su na slici 2.1.



Slika 2.1. Evolucijsko računarstvo - podjela

2.1. Evolucijski algoritmi

Evolucijski algoritmi su stohastičke metode pretraživanja koje su razvijene prema principima biološke evolucije. Reprodukcija, mutacija, selekcija i rekombinacija su mehanizmi pomoću kojih evolucijski algoritmi dolaze do optimalnog rješenja.

Evolucijski algoritmi se koriste za rješavanje problema koji se ne mogu riješiti u polinomijalnom vremenu, kao što su klasični NP-teški problemi.

Evolucijski algoritmi obuhvaćaju široko područje a dijelimo ih na:

- Genetske algoritme
- Genetsko programiranje
- Evolucijsko programiranje
- Evolucijske strategije

U evolucijske algoritme mnogi stručnjaci još ubrajaju i klasifikatorske sustave. To su prilagodljivi sustavi koji kroz interakciju s okolišem izvršavaju zadatke. Oni se novim stanjima okoliša prilagođavaju postupkom pokušaja i pogrešaka.

2.2. Algoritmi rojeva

Algoritmi rojeva su populacijski algoritmi inspirirani ponašanjem rojeva različitih insekata u prirodi poput mrava, pčela, osa, te također ponašanjem drugih životinja. Populacija se sastoji od velikog broja jedinki kojima je cilj pretražiti prostor rješenja na temelju prethodnog vlastitog iskustva i iskustva susjednih jedinki.

U algoritme rojeva ubrajamo:

- Algoritam kolonije mrava
- Algoritam roja čestica
- Algoritam kolonije pčela
- Ostali

2.3. Ostali algoritmi

U granu ostali algoritmi, koji spadaju u evolucijsko računarstvo, pripada još niz drugih algoritama, a od značajnijih genetsko kaljenje(GAn), umjetni imunološki algoritmi(AIS), diferencijska evolucija(DE) te algoritam harmonijske pretrage [3].

3. Genetski algoritmi

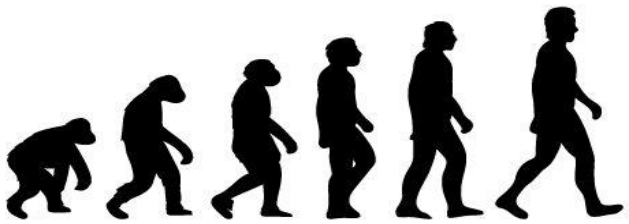
Genetski algoritam je heuristička metoda koja imitira prirodni evolucijski proces. To je algoritam koji pretražuje prostor rješenja s ciljem pronalaska optimalnog rješenja problema.

Genetski algoritmi dominantno rješavaju različite probleme optimizacije te se primjenjuju u područjima kao što su problem traženja najkraćeg puta, problem trgovačkog putnika, područje učenja kod neuronskih mreža, kod transporta, itd.

Evolucija, premisa na kojoj se temelje genetski algoritmi, je prirodni proces traženja najbolje i najprilagodljivije jedinke na okolinu i uvjete u prirodi, odnosno to je proces prilagođavanja živih bića na uvjete u kojima žive. Prema tome, možemo zaključiti da je evolucija metoda optimiranja.

Charles Darwin je sredinom 19. stoljeća u svojoj knjizi *Postanak vrsta* (1859.) postavio znanstvenu Teoriju evolucije putem prirodnog odabira. On je utvrdio da je prirodna selekcija glavni pokretač evolucije.

U evoluciji priroda određuje uvjete za život a one jedinke s boljim genetskim materijalom bolje će se prilagoditi tim uvjetima, a slabije jedinke će s vremenom izumrijeti. Dok neke vrste



Slika 3.1. Evolucija čovjeka [6]

izumiru, druge vrste će stvarati potomke te na taj način broj živih bića uvijek ostaje otprilike jednak.

Ono što su okolina i uvjeti u prirodi kao glavni ključ selekcije nad živim bićima, to je funkcija cilja ili dobrote u genetskim algoritmima. U prirodnim uvjetima bića koja su se uspješno prilagodila uvjetima i okolini u kojima žive, imaju najveće šanse za preživljavanje i parenje, a time i prenošenje svojeg genetskog materijala. Procesom evolucije postupno će se mijenjati

genetski materijal živih bića, te se vremenom ona prilagođavaju okolini u kojoj se nalaze. Kod genetskih algoritama populacija se sastoji od određenog broja jedinki, dok jedna jedinka predstavlja jedno rješenje. Selekcijom, koja se bazira na funkciji dobrote, odabiru se dobre jedinke koje se prenose u sljedeću generaciju, te se manipulacijom njihovog genetskog materijala stvaraju nove jedinke. Tim procesom iz generacije u generaciju algoritam stvara sve bolje i bolje jedinke te u konačnici dolazi do optimalnog rješenja problema [1].

Genetske algoritme izumio je John Holland, a razvio ih je s kolegama i svojim studentima kroz 1960e i 1970e. Hollandova ideja bila je proučavanje mehanizama prilagodbe u prirodi te razvoj načina na koje bi te mehanizme mogao prenijeti na računalne sustave.

Kao apstrakciju biološke evolucije, te teoretski okvir za adaptaciju, Holland je 1975. godine predstavio genetske algoritme u svojoj knjizi *Adaptation in Natural and Artificial Systems*. Holland je koristio terminologiju biologije, te je kodirana rješenja nazvao kromosomima, osnovnu građevnu jedinicu kromosoma genima, a vrijednosti koje geni mogu poprimiti alelima.
Prema Hollandu genetski algoritam je metoda za prijelaz iz jedne populacije kromosoma u drugu populaciju korištenjem „prirodne selekcije“ zajedno s genetski inspiriranim operatorima odnosno rekombinacijom, mutacijom i inverzijom.

No i prije Hollandovog predstavljanja genetskih algoritama mnogi drugi računalni znanstvenici proučavali su kako bi iskoristili mehanizme prirodne evolucije kao inspiraciju te upotrebom populacije rješenja došli do rješavanja praktičnih inženjerskih problema.

1960ih godina Rechenberg je predstavio evolucijske strategije, metodu koju je koristio za optimiziranje realnih parametara za različite uređaje. Njegova metoda se sastojala od „populacije“ od dvije individue, jednog roditelja te jednog potomka, kod koje je potomak nastao procesom mutacije roditelja.

Fogel, Owens i Walsh razvili su 1960ih godina evolucijsko programiranje, te su također koristili samo mutaciju u svrhu dobivanja varijacije.

Kada je Holland predstavio algoritam koji se bazirao na populaciji s mehanizmima rekombinacije, mutacije i inverzije, ostvario je velik uspjeh te je njegov algoritam okarakteriziran kao velika inovacija.

Mnogi drugi znanstvenici u tom su razdoblju također razvijali algoritme inspirirane prirodnom evolucijom za optimizaciju te strojno učenje. U tom području radili su Friedman, Box, Bledsoe,

Reed i drugi, ali njihov se rad nije toliko istaknuo kao što su evolucijske strategije, evolucijsko programiranje te genetski algoritmi [4].

3.1. Osnove genetskih algoritama

Genetski algoritam radi na sljedeći način. Algoritam radi s populacijom jedinki, u prvom koraku kreira populaciju mogućih rješenja problema, najčešće slučajnim odabirom. Svaka jedinka unutar populacije predstavlja jedno moguće rješenje. Svakoj jedinki pomoću funkcije dobrote (engl. *fitness function*) dodjeljuje njezinu dobrotu (engl. *fitness*). Selekcijom odabire individue iz populacije koje predstavljaju roditelje, s time da one individue s većom dobrotom imaju veće šanse da će biti odabrane. Roditelji pomoću operatora križanja (engl. *crossover)* te pomoću rekombinacije, odnosno procesom „parenja“, izmjenjuju genetski materijal i stvaraju novu generaciju jedinki. Na novu generaciju određenim stupnjem vjerojatnosti utječe i mehanizam mutacije.

Iz generacije u generaciju algoritam pronalazi sve bolja i bolja rješenja, sve do završetka algoritma, koji je uglavnom unaprijed određen brojem generacija, stupnjem konvergencije ili vremenskim ograničenjem, kada algoritam dolazi do najboljeg rješenja problema [3].

Potrebno je napomenuti da genetski algoritmi, jednako kao i ostali algoritmi koji spadaju u evolucijsko računarstvo, ne garantiraju pronalazak optimalnog rješenja problema s obzirom da ne pretražuju cijeli prostor rješenja.

Osnovni građevni elementi genetskog algoritma:

- Populacija jedinki
- Funkcija dobrote
- Selekcija
- Križanje
- Mutacija

Inverzija, prema Hollandu također jedan od građevnih elemenata genetskih algoritama, u današnje vrijeme se rijetko koristi.

Primjer jednostavnog genetskog algoritma:

1. `slučajnim odabirom generira se populacija` *`pop`* `s` *`n`* `jedinki`

2. sve dok nisu ispunjeni uvjeti završetka algoritma, radi sljedeće (svaka iteracija zove se generacija):
    - funkcijom dobrote svakoj jedinki izračuna se njezina dobrota
    - stvori novu populaciji *new-pop* s *n* jedinki
    - sve dok *new-pop* nije popunjena radi sljedeće:
        - slučajnim odabirom odaberi dvije individue iz *pop;* individue s većom dobrotom imaju veće šanse da će biti odabrane
        - rekombinacijom dvije odabrane individue dobiva se nova individua koja ulazi u *new-pop*
    - svaka individua iz *new-pop* ima šanse za mutiranje s unaprijed određenom vjerojatnošću mutacije
    - zamjena *pop* s *new-pop*
3. odabir jedinke iz *pop* s najvećom dobrotom kao rješenje problema

Iz ovih koraka možemo zaključiti da su genetski algoritmi iterativni proces u kojem svaka iteracija predstavlja jednu generaciju. Broj generacija varira, ovisno o problemu u kojem se genetski algoritmi koriste. Uvjet završetka, kao što je već spomenuto, može biti broj generacija, vremensko ograničenje ili stupanj konvergencije. Nakon što se izvrši određeni broj iteracija, očekujemo da ćemo dobiti optimalno rješenje problema.

3.2. Prostor rješenja

Podjela najčešćih prikaza rješenja:

- Prikaz nizom bitova
- Prikaz poljem ili vektorom brojeva
- Prikaz permutacijama i matricama
- Prikaz složenijim strukturama podataka
- Prikaz stablima

Odabir prikaza rješenja je vrlo važan i osjetljiv zadatak iz razloga što on može bitno utjecati na učinkovitost samog izvršavanja genetskog algoritma.

Najjednostavniji način prikaza rješenja jest u obliku niza bitova, pri čemu je svaka jedinka predstavljena kao niz nula i jedinica jednake duljine. Ovakav prikaz podataka naziva se binarni prikaz ili prikaz nizom bitova.

No, postoje različiti problemi u kojima nije moguće koristiti binarni prikaz, pa je tada potrebno koristiti druge prikaze podataka. Tako jedinka može biti prikazana i kao polje brojeva ako je problem višedimenzionalni i kao realan broj pa i struktura podataka. Jedinka može biti prikazana kao bilo kakva struktura podataka koja opisuje svojstva određene jedinke. Npr. u slučaju problema pronalaska najkraćeg puta jedinka može biti polje cijelih brojeva u kojem svaki broj označava mjesto koje se mora posjetiti, a cijeli niz brojeva označava put koji se mora proći od početnog mjesta do mjesta koje označava cilj.

Za genetske algoritme važno je da jedinka predstavlja jedno moguće rješenje zadanog problema [1].

U problemu koji se obrađuje u ovome radu jedinka je prikazana kao lista ruta, dok jednu rutu predstavlja polje lokacija koja se moraju obići unutar jedne rute.

3.3. Populacija i inicijalizacija

U genetskim algoritmima populacija predstavlja skup jedinki od kojih je svaka jedinka potencijalno rješenje zadanog problema. U procesu inicijalizacije stvara se početni skup jedinki, odnosno početna populacija (engl. *initial population*). Jako je važno odrediti optimalnu veličinu populacije za određeni problem iz razloga što veličina populacije utječe na sveukupne performanse genetskog algoritma. Pozitivna strana manje veličine populacije jest brže izvođenje algoritma a negativna strana je što može voditi samo do pronalaska lokalnog optimuma. Dok s druge strane veća populacija za rezultat ima nešto sporije izvođenje ali zato ima veću vjerojatnost pronalaska globalnog optimuma.

Neke od metoda inicijalizacije:

- Uniformno generiranje jedinki – sve jedinke su jednake
- Generiranje jedinki slučajnim odabirom
- Inicijalizacija rješenjem dobivenim nekom drugom optimizacijskom metodom

3.4. Funkcija dobrote

Funkcija dobrote je funkcija koja ocjenjuje koliko je pojedino rješenje, odnosno jedinka, dobro s obzirom na zadani problem, te mu dodjeljuje dobrotu. Može se reći da je funkcija dobrote funkcija koja ocjenjuje kvalitetu jedinke. U literaturi se još naziva funkcija sposobnosti, funkcija cilja ili eval funkcija [1].

Što jedinka ima veću dobrotu, veće su joj šanse za preživljavanje, odnosno ima veće šanse za reprodukciju te prenošenje genetskog materijala na sljedeću generaciju. S obzirom da je funkcija dobrote najčešće korištena funkcija u genetskim algoritmima treba posvetiti puno pažnje kod njezine implementacije. Te je odabir funkcije dobrote od iznimne važnosti kod procesa izrade genetskog algoritma. Jer sporo izvršavanje funkcije dobrote može negativno utjecati na sveukupne performanse genetskog algoritma. Funkcija dobrote trebala bi biti što jednostavnija a njezino izvođenje što brže.

Općeniti zahtjevi vezani za funkciju dobrote:

- trebala bi biti jasno definirana, odnosno izračun dobrote jedinke trebao bi biti lako razumljiv
- trebala bi biti efikasno implementirana. U slučaju loše implementacije, odnosno sporog izvođenja, sveukupna efikasnost izvođenja genetskog algoritma biti će smanjena
- trebala bi kvantitativno izmjeriti koliko je pojedino rješenje dobro s obzirom na određeni problem
- trebala bi generirati intuitivne rezultate, odnosno najbolja jedinka trebala bi imati najveću dobrotu i obrnuto

3.5. Selekcija

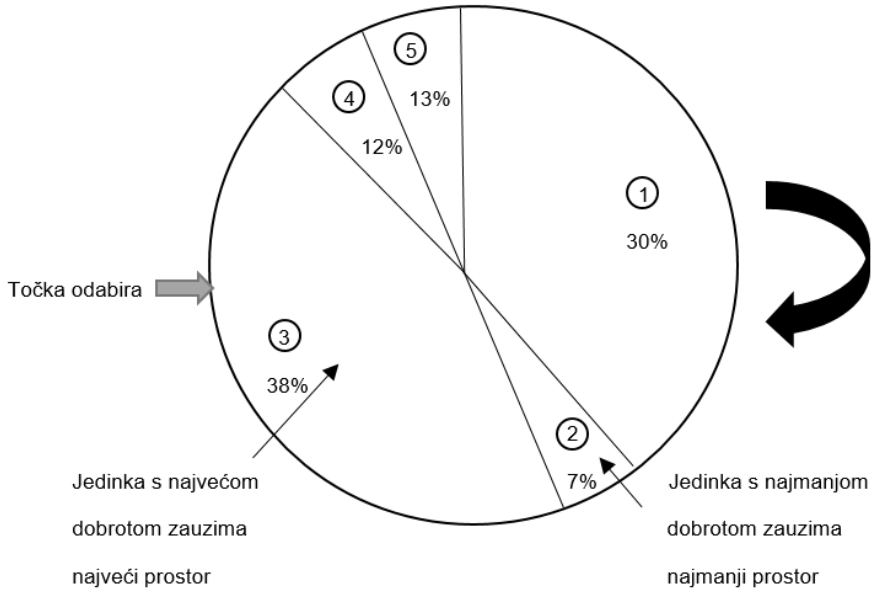
Uloga selekcije je odabir jedinki koje će sudjelovati u stvaranju nove populacije, odnosno očuvanje i prenošenje dobrih svojstava na sljedeću generaciju jedinki. Krajnji cilj selekcije je da svaka sljedeća generacija bude bolja od prethodne. To se postiže tako da boljim jedinkama dodijelimo veću vjerojatnost za sudjelovanje u procesu stvaranja nove generacije. Ali bitno je napomenuti da se ne smiju zanemariti jedinke s lošijom dobrotom iz razloga što one također mogu sadržavati dobar genetski materijal. Zato je potrebno nekim mehanizmom osigurati da i

lošije jedinke imaju neku, uglavnom manju, vjerojatnost da budu odabrane i korištene u procesu reprodukcije [1].

U popularnije tehnike selekcije ulaze: jednostavna selekcija, turnirska selekcija, eliminacijska selekcija i elitizam kao dodatni mehanizam koji se može primijeniti kod svake vrste selekcije.

3.5.1. Jednostavna selekcija

Jednostavna selekcija (engl. *roulette wheel parent selection*) je selekcija kod koje se roditelji biraju proporcionalno prema svojoj dobroti. Što jedinka ima veću dobrotu, ima veću vjerojatnost da bude odabrana. Ovaj način selekcije generira novu populaciju iz stare, odnosno prethodne, generacije koja ima jednak broj jedinki kao i stara generacija.



Slika 3.2. Roulette wheel parent selection

Kod jednostavne selekcije može doći do *problema skale*, što znači da u slučaju da sve jedinke imaju veliku vrijednost dobrote, vjerojatnost odabira najbolje i najlošije jedinke je praktički ista. Ovaj problem je dobio naziv *problem skale,* zbog toga što ovaj postupak selekcije dobro funkcionira za male brojeve a sve lošije kako raste skala (magnituda brojeva) [3].

3.5.2. Eliminacijska selekcija

Za razliku od jednostavne selekcije, koja u jednom koraku raspolaže s dvije populacije, jednom iz prethodne generacije i drugom koju generira reprodukcijom boljih jedinki, eliminacijska

selekcija bira one lošije koje će eliminirati i reprodukcijom zamijeniti novim jedinkama. Algoritam selekcije loših jedinki sličan je jednostavnoj selekciji, ali umjesto da je vjerojatnost selekcije proporcionalna funkciji dobrote, vjerojatnost selekcije je veća što je dobrota manja. Te se umjesto funkcije dobrote definira funkcija kazne. Funkcija kazne predstavlja vjerojatnost odabira jedinke za izbacivanje, a izračunava se tako da se od maksimalne dobrote oduzme dobrota jedinke. Tim postupkom se rješava problem očuvanja jedinke s najvećom dobrotom zato jer je njezina kazna jednaka nuli.

3.5.3. Turnirska selekcija

Turnirska selekcija je selekcija koja iz stare populacije *vel_pop* puta (onoliko koliko ima jedinki unutar populacije) odabire s jednakom vjerojatnošću *n* (2,3,4,…) jedinki, uspoređuje ih te najbolju od njih kopira u bazen za reprodukciju. U sljedećem koraku, nakon što genetski algoritam *vel_pop* puta izvrši turnir, na odabrane jedinke djeluju genetski operatori te stvaraju novu populaciju. Turnirska selekcija se može koristiti i na način da se umjesto odabira najbolje jedinke iz turnira odabire najlošija jedinka te se ona eliminira i nadomješta djetetom dvije preostale jedinke. Takva selekcija naziva se eliminacijska turnirska selekcija.

3.5.4. Elitizam

Elitizam je mehanizam koji se dodatno može koristiti uz već navedene vrste selekcije. To je mehanizam koji osigurava da se najbolja ili nekoliko najboljih jedinki u generaciji ne izgube tijekom procesa selekcije ili da ne budu promijenjene prilikom djelovanja genetskih operatora. Nije ga potrebno koristiti kod eliminacijskih selekcija iz razloga što kod takvih selekcija vjerojatnost eliminacije najbolje jedinke jednaka je nuli. Primjenom elitizma osiguravamo da se svaka nova generacija kreće prema globalnom optimumu, odnosno rješenju problema. No, s druge strane, korištenje elitizma znači da u svakom koraku evolucije moramo pronaći najbolju jedinku, a to zahtjeva više procesorskog vremena koje u konačnici može dovesti do usporavanja izvršavanja genetskog algoritma [1].

3.6. Genetski operatori

Operatori koji se koriste u genetskim algoritmima su križanje, mutacija i inverzija. Dok su križanje i mutacija često korišteni genetski operatori, operator inverzije, kako je već ranije spomenuto, u današnje vrijeme se rijetko koristi. Jedna od najvažnijih karakteristika genetskih

algoritama je reprodukcija. Reprodukcija je razmnožavanje pomoću genetskog operatora križanja u kojem sudjeluju dobre jedinke koje su preživjele proces selekcije. U procesu reprodukcije može doći do promjene nekih gena unutar jedinke, a taj mehanizam se zove mutacija [1].

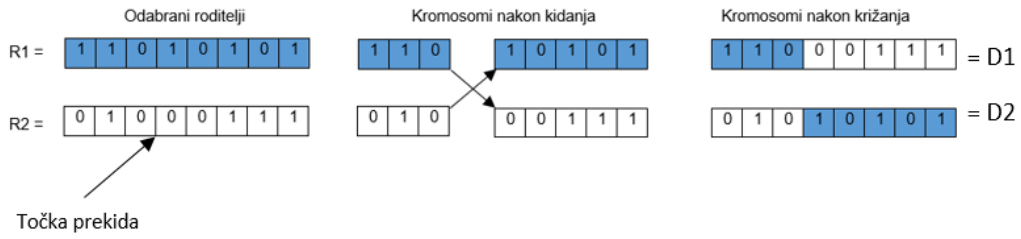
3.6.1. Križanje

Križanje (engl. *crossover*) je binarni operator u kojem sudjeluju dvije jedinke koje predstavljaju roditelje. Procesom križanja nastaju jedna ili dvije jedinke koje se nazivaju djeca, te ona u tom procesu nasljeđuju svojstva, odnosno genetski materijal, od svojih roditelja. Očekuje se da se križanjem dobrih roditelja dobije dijete koje će biti jednako dobro ako ne i bolje od roditelja [1].

Postoji velik broj vrsta križanja od kojih se neke vrste mogu koristiti samo kod problema s bitovnim prikazom, te druge vrste križanja koje su prilagođene problemima s drugačijim vrstama prikaza.

Neke od vrsta križanja kod problema s bitovnim prikazom su: križanje s jednom točkom prekida, križanje s t-točaka prekida, uniformno križanje, p-uniformno križanje.

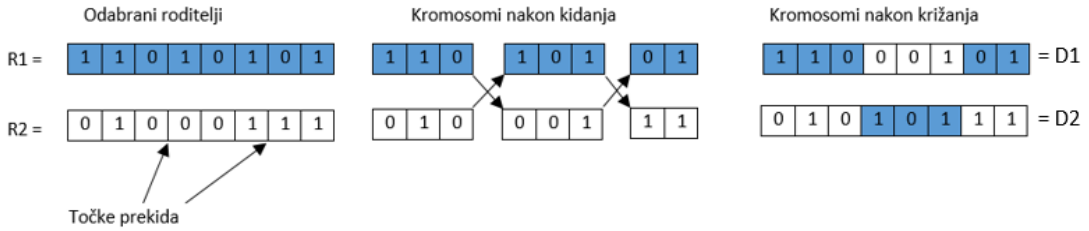
Najjednostavnija vrsta križanja je križanje s jednom točkom prekida, koje je prikazano na slici 3.3. Nakon što se odaberu dva roditelja koja će sudjelovati u procesu križanja, slučajnim odabirom odabire se točka prekida koja je zajednička za oba roditelja. Potom se stvaraju dvije nove jedinke – dijete D1 i dijete D2. Dijete D1 dobiva prvi dio bitova od roditelja R1 i drugi dio bitova od roditelja R2, a drugo dijete prvi dio bitova od roditelja R2 i drugi dio bitova od roditelja R1.



Slika 3.3. Križanje s jednom točkom prekida

Križanje s *t*-točaka prekida je općenitiji slučaj križanja s jednom točkom prekida. U ovom slučaju broj točaka prekida može biti između 1 i $b - 1$, gdje je b broj bitova korištenih za prikaz rješenja. Kao i kod križanja s jednom točkom prekida i u ovoj vrsti križanja stvaraju se dva

djeteta uzimanjem cik-cak segmenata bitova iz jednog pa iz drugog roditelja. Primjer križanja s dvije točke prekida prikazan je na slici 3.4.



Slika 3.4. Križanje s dvije točke prekida

Uniformno križanje je križanje s $b - 1$ prekidnih točaka, gdje je b broj bitova korištenih za prikaz rješenja, koje je prikazano na slici 3.5. Bitovni prikaz oba roditelja se „raspadne“ nakon svakog bita, te nakon toga svako dijete bira što će uzeti od kojeg roditelja. Vjerojatnost da dijete naslijedi svojstvo od jednog roditelja je 50%, odnosno jednake su šanse da će naslijediti svojstvo od jednog i od drugog roditelja [3]. Ako se vjerojatnosti odabira svojstava od roditelja razlikuju tada se takvo križanje naziva *p*-uniformno križanje.

b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	
1	1	0	1	1	1	0	1	0	0	RODITELJI
0	1	1	0	1	1	1	0	0	1	
1	1	0	0	1	1	0	1	0	1	DIJETE

Slika 3.5. Uniformno križanje

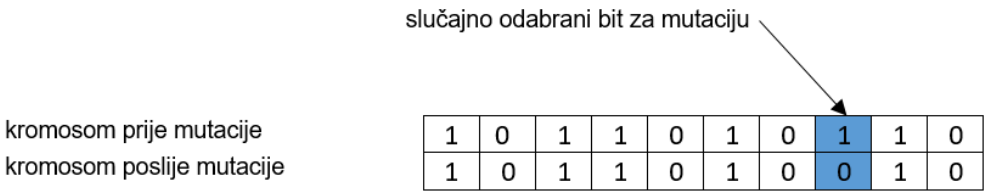
3.6.2. Mutacija

Mutacija je operator koji djeluje na jedinku tako da slučajnim odabirom mijenja njezin jedan ili više gena. S obzirom da djeluje nad jednom jedinkom mutacija je unarni operator, a njezin rezultat je izmijenjena jedinka. Mutacija se koristi kao mehanizam koji osigurava da algoritam ne zapne u nekom lokalnom optimumu. U slučaju da cijela populacija završi u lokalnom optimumu, jedino slučajnim pretraživanjem prostora rješenja, odnosno mutacijom, algoritam pronalazi bolja rješenja.

Vrijednost parametra vjerojatnosti kod genetskih algoritama je uglavnom vrlo mala. U slučaju da vjerojatnost mutacije teži prema jedinici, tada se može reći da algoritam prelazi u algoritam

slučajne pretrage, a ako vjerojatnost teži prema nuli algoritam će najvjerojatnije već u početku izvršavanja stati u nekom lokalnom optimumu.

S obzirom na različite vrste prikaza rješenja, postoji i velik broj vrsta mutacije. Spomenut ćemo samo neke od njih. Jednostavna mutacija s jednakom vjerojatnošću mijenja svaki bit kromosoma, prikazana je na slici 3.6. Potpuna mutacija za razliku od jednostavne djeluje nad jedinkama, a ne nad genima. Ona slučajnim odabirom odabire jedinku i nakon toga ispremiješa sve bitove. Miješajuća mutacija također slučajnim odabirom bira jedinku nad kojom će izvršiti mutaciju, prvu i drugu granicu, nakon toga ili izmiješa gene ili ih slučajno generira ili invertira [1].



Slika 3.6. Jednostavna mutacija

3.7.Parametri genetskih algoritama

Da bi genetski algoritam mogao isporučiti zadovoljavajuće rezultate potrebno je pronaći odgovarajuće vrijednosti parametara. Osnovni parametri kod genetskih algoritama su veličina populacije, vjerojatnost križanja i vjerojatnost mutacije.

Veličina populacije je broj jedinki unutar populacije u svakoj generaciji genetskog algoritma. Što je veličina populacije veća to će prostor rješenja kojeg genetski algoritam može pretražiti biti veći a to će u konačnici dovesti do točnijeg i globalno optimalnog rješenja, no ipak prevelika populacija može dovesti do usporavanja izvođenja algoritma.
S druge strane manja veličina populacije većinom će rezultirati s manje poželjnim rješenjima koji se nalaze u lokalnim optimumima, ali zato zahtijeva manje vremena za izračunavanje po generaciji. Stoga, za optimalne performanse algoritma potrebno je pronaći balans, odnosno odabrati prikladnu veličinu populacije za zadani problem.

Vjerojatnost križanja također ima utjecaj na sveukupne performanse genetskog algoritma. U slučaju velike vjerojatnosti križanja algoritam omogućava pronalazak mnogih novih i potencijalno boljih jedinki. Dok mala vjerojatnost križanja omogućava da algoritam sačuva

genetski materijal boljih jedinki za novu generaciju. Vjerojatnost križanja trebala bi biti dosta visoka iz razloga da algoritam iz generacije u generaciju može pronalaziti sve bolja rješenja.

Vjerojatnost mutacije predstavlja vjerojatnost kojom će određeni gen unutar jedinke, odnosno kromosoma, biti podvrgnut genetskom operatoru mutacije. Kao i kod prethodno spomenutih parametara, ne postoji točna vrijednost vjerojatnosti mutacije za neki genetski algoritam, ali neke vrijednosti će dati bolje rezultate od drugih. Velika vjerojatnost mutacije omogućava veliku genetsku raznovrsnost unutar populacije, te se također može smatrati mehanizmom koji sprječava da algoritam zapne unutar nekog lokalnog optimuma, no ipak prevelika vjerojatnost može imati i negativan utjecaj. Jer u slučaju velike vjerojatnosti rješenja u svakoj generaciji su praktički dobivena na slučajan način. A u slučaju premale vjerojatnosti mutacije algoritmu može biti potrebno previše vremena u pretraživanju prostora rješenja za pronalaskom zadovoljavajućeg rješenja. Vjerojatnost mutacije trebala bi biti takva da omogućava dovoljno veliku raznolikost, te da sprječava da algoritam zapne u nekom od lokalnih optimuma, no s druge strane ne toliko velika da se ne izgubi dobar genetski materijal iz prethodne generacije [5].

Uz navedene parametre genetski algoritmi također mogu sadržavati i mnoge druge parametre kao što su: veličina turnira u slučaju turnirske selekcije, broj elitnih jedinki u slučaju korištenja elitizma, maksimalan broj generacija, broj jedinki za eliminaciju kod eliminacijskog genetskog algoritma i drugi.

4. Optimizacija poslovanja dostavne službe

4.1. Motivacija

U današnje vrijeme zbog široke mogućnosti pristupa Internet trgovinama ljudi sve više koriste usluge dostavnih službi.

Jedna od ključnih zadaća dostavnih službi je dostava robe korisnicima u što kraćem vremenskom periodu, no s druge strane, u interesu samih dostavnih službi je i optimizacija troškova.

Dostavne službe žele optimizirati dostavljačke rute, odnosno smanjiti troškove dostave na način da osmisle optimalne rute kojima će dostaviti robu kupcima.

Problem optimiziranja ruta kojeg dostavne službe pokušavaju riješiti je opće poznat problem zvan problem usmjeravanja vozila, kao što je već spomenuto u uvodnom dijelu ovog rada.

Cilj ovog rada je rješavanje spomenutog problema na način da će se izraditi prototip softvera koji će pokušati optimizirati dostavljačke rute korištenjem genetskih algoritama.

4.2. Opis problema

Problem usmjeravanja vozila (VRP) je NP-težak kombinatorni optimizacijski problem. VRP problem je definiran kao skup vozila *m*, inicijalno lociranih u skladištu, koji moraju dostaviti određenu količinu robe do *n* kupaca. Određivanje optimalnih ruta za određeni broj vozila koja će poslužiti korisnike, uz minimalnu cijenu, predstavlja VRP problem. Rješenje problema predstavlja lista ruta koje sve počinju i završavaju u istom skladištu, te zadovoljavaju ograničenje da svaki kupac bude poslužen samo jednom [7].

Zbog sve kompleksnijih problema u stvarnom svijetu nastala su mnoga proširenja VRP problema kao što su: problem usmjeravanja vozila s ograničenim kapacitetom (engl. *Capacitated Vehicle Routing Problem,* skraćeno CVRP), problem usmjeravanja vozila s vremenskim ograničenjem (engl. *Vehicle Routing Problem with Time Windows,* skraćeno VRPTW*),* problem usmjeravanja vozila s više skladišta (engl. *Multiple Depot Vehicle Routing Problem,* skraćeno MDVRP) te mnogi drugi.

Najuobičajenije metode za rješavanje VRP problema su heuristike i metaheuristike iz razloga što u slučaju većeg broja gradova nijedan egzaktni algoritam ne može garantirati da će pronaći optimalne rute u prihvatljivom vremenskom periodu. Neke od najpopularnijih metoda su simulirano kaljenje, kolonija mrava, genetski algoritmi, tabu pretraga. U ovom radu za rješavanje VRP problema, kao što je već spomenuto, koristit će se genetski algoritmi.

Također je potrebno napomenuti da u ovom radu za rješavanje spomenutog problema u obzir nisu uzete cestovne prometnice, već se za udaljenost između dviju lokacija koristila zamišljena ravna linija koja ih spaja. Lokacije su određene vlastitom geografskom širinom i dužinom te se na temelju toga izračunava njihova međusobna udaljenost. U obzir nije uzeta mogućnost da je udaljenost između istih lokacija različita u dva smjera, već je za potrebe ovog rada uzeta pretpostavka da su udaljenosti u oba smjera između dvije lokacije jednake. Zanemaruju se uvjeti na cesti i eventualna veća potrošnja goriva u različitim okolnostima.

4.3. Implementacija sustava

Programsko rješenje koje je izrađeno u sklopu ovog rada ostvareno je u programskom jeziku Java.
Osmišljen je algoritam koji na temelju podataka, odnosno lokacija s pripadajućim geografskim širinama i dužinama, izračunava najkraće moguće rute za *n* vozila. Svaka ruta sastoji se od lokacija koje vozilo mora proći prema određenom rasporedu, s time da je početna lokacija, odnosno sjedište, ujedno i lokacija u kojoj se ruta završava.

U sklopu programskog rješenja ostvarena su dva modula. Jedan modul se odnosi na konzolnu aplikaciju koja se pokreće putem naredbenog retka preko kojeg joj se moraju predati odgovarajući parametri algoritma. Te drugi modul koji se odnosi na aplikaciju s grafičkim prikazom (engl. *Graphical User Interface*, skraćeno GUI), koja sadrži i vizualizaciju određenih ruta, radi lakšeg snalaženja i korištenja.

Programsko ostvarenje dohvaća lokacije iz baze podatka, u ovom radu korišten je MSSQL softver, te je prije korištenja potrebno popuniti bazu podataka s lokacijama koje se trebaju posjetiti, odnosno na koje je potrebno dostaviti robu.

4.3.1. Evolucija algoritma

Algoritam ovog programskog rješenja izvodi evoluciju kroz generacije tako što u svakoj generaciji stvara novu populaciju iste veličine.

Algoritam u prvom koraku generira inicijalnu populaciju određene veličine, a ona je određena vanjskim parametrom algoritma. Nakon toga ocijeni kvalitetu inicijalne populacije, pa sve do zaustavljanja algoritma u svakoj generaciji generira novu populaciju križanjem stare populacije, te na novu populaciju primjenjuje operator mutacije. Na kraju izvođenja algoritam isporuči najbolje rješenje problema. Uvjet završetka algoritma je ili maksimalan broj generacija ili ako se dosegne određeni broj generacija bez napretka u dobroti jedinke.

Parametri algoritma su: veličina populacije, vjerojatnost križanja, vjerojatnost mutacije, broj elitnih jedinki, veličina turnira te broj vozila.

Ovaj proces detaljnije je opisan sljedećim pseudokom:

```
inicijaliziraj glavni algoritam ga s pripadajućim parametrima:
velPop, vjKrizanja, vjMutacije, brElitnih, velTurnira, brVozila
generiraj inicijalnu populaciju pop veličine velPop
ocijeni populaciju pop
brGeneracija postavi na 1
        while broj generacija bez napretka < maxBrBN ili
        brGeneracija < maxBrGen
        križanjem stvori novu populaciju newPop te zamijeni
        pop s newPop
        primijeni operator mutacije na populaciju pop
        ocijeni populaciju pop
        uvećaj brGeneracija za 1
prikaži najbolje rješenje iz posljednje generacije
```

Parametri:

ga – glavna klasa genetskog algoritma
velPop – veličina populacije
vjKrizanja – vjerojatnost križanja
vjMutacije – vjerojatnost mutacije
brElitnih – broj elitnih jedinki
velTurnira – veličina turnira
brVozila – broj vozila, odnosno broj ruta unutar jedne jedinke
pop – trenutna populacija
brGeneracija – broj generacija genetskog algoritma
maxBrBN – maksimalan broj generacija bez napretka
maxBrGen – maksimalan broj generacija
newPop – nova populacija u sljedećoj generaciji

U sljedećim će poglavljima detaljnije biti objašnjeni sastavni dijelovi genetskog algoritma koji se obrađuje u ovom radu: prikaz rješenja, funkcija dobrote, selekcija, križanje te mutacija.

4.3.2. Prikaz rješenja

Prikaz rješenja, odnosno jedinka, u ovom radu prikazana je kao lista ruta, dok jednu rutu predstavlja određeni broj lokacija koja se moraju obići unutar jedne rute. Lokacija, kao što je već prije spomenuto, je prikazana kao objekt koji se sastoji od naziva lokacije, te geografske dužine i širine prema kojima se izračunava međusobna udaljenost između dvije lokacije.

Primjer jedinke s 3 vozila i 15 lokacija koje treba obići, te pripadajućim udaljenostima, prikazan je na slici 4.1.

```
Ruta[ HQ-Zagreb - Osijek - Ilok - Vukovar - Vinkovci - Slavonski Brod - HQ-Zagreb, udaljenost: 566.37km ]
Ruta[ HQ-Zagreb - Zapresic - Krapina - Ivanec - Varaždin - Koprivnica - HQ-Zagreb, udaljenost: 203.99km ]
Ruta[ HQ-Zagreb - Jastrebarsko - Karlovac - Gospić - Crikvenica - Rijeka - HQ-Zagreb, udaljenost: 399.39km ]
```

Slika 4.1. Primjer jedinke s 3 vozila

4.3.3. Funkcija dobrote

U problemima kao što je ovaj, te sličnim, funkcija dobrote se relativno lako izračunava. Obično je ona predstavljena kao ukupna udaljenost svih ruta koja će vozila morati proći.

No, samo zbog lakšeg razumijevanja i preglednosti, dobrota se u ovom radu izračunava tako da se 100 podijeli s ukupnom udaljenosti. Tako da, što jedinke imaju manju udaljenost, imati će veću dobrotu.

Npr. jedinka kojoj je ukupna udaljenost 1500km ima vrijednost dobrote od 0,067, dok jedinka s ukupnom udaljenosti od 1250km ima vrijednost dobrote od 0,08.

$$\text{Funkcija dobrote} = \frac{100}{\text{Ukupna udaljenost}}$$

Slika 4.2. Funkcija dobrote

4.3.4. Selekcija

Vrsta selekcija odabrana u ovom radu je turnirska selekcija s mehanizmom elitizma. Mehanizam elitizma, kao što je već ranije spomenuto, omogućava da se *n* najboljih jedinki prebaci u iduću generaciju, odnosno sačuva od mogućih promjena putem genetskih operatora križanja i mutacije. U ovom slučaju broj elitnih jedinki je određen kao vanjski parametar algoritma.

Algoritam omogućava svakoj jedinki iz generacije mogućnost *parenja*, dok će one bolje jedinke više puta sudjelovati u *parenju,* te im se tako daje veća vjerojatnost da će svoj genetski materijal prenijeti na iduću generaciju.

Algoritam prolazeći redom kroz listu jedinki u svakom koraku odabire prvog roditelja, dok se drugi roditelj dobiva kroz turnir. Slučajnim odabirom odabire se *n* jedinki koje sudjeluju u turniru, te se ona s najvećom dobrotom određuje kao drugi roditelj. Veličina turnira je također određena kao vanjski parametar algoritma.

4.3.5. Križanje

U ovome problemu geni, te poredak gena unutar kromosoma jako su bitni. Kod problema usmjeravanja vozila ne smije postojati više od jedne kopije gena unutar jednog kromosoma, odnosno unutar jednog rješenja svaka lokacija unutar svih ruta smije biti navedena samo jednom. A to sve iz razloga što bi se inače stvarala nevažeća rješenja.

Primjer jedne važeće i nevažeće rute s 4 lokacije: Lok1, Lok2, Lok3, Lok4.

Ruta1: Lok1, Lok2, Lok3, Lok4.
Ruta2: Lok2, Lok3, Lok4, Lok2.

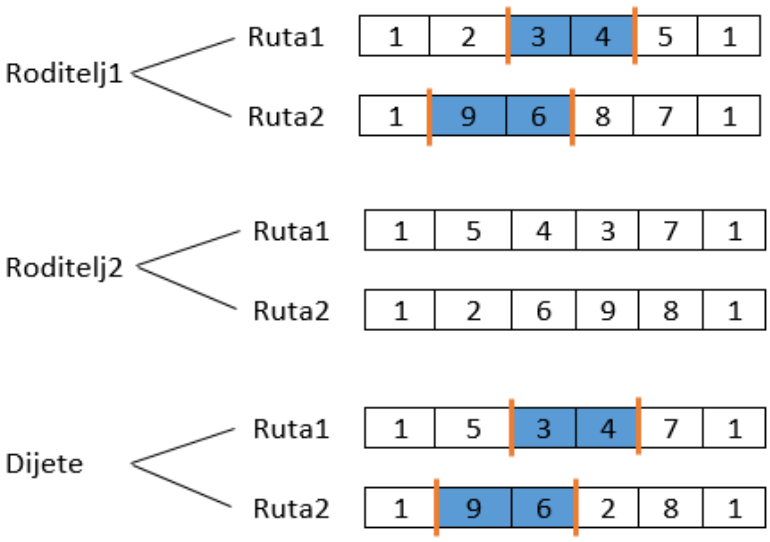
Ruta2 je nevažeća iz razloga što će Lok2 biti dva puta posjećena, dok Lok1 neće uopće biti posjećena.

Te je također važno poštivati poredak gena unutar roditelja, odnosno poredak lokacija unutar ruta određenog rješenja, iz razloga što poredak utječe na vrijednost dobrote samog rješenja. Jer ruta s lokacijama Lok1, Lok2, Lok3, Lok4 je potpuno drugačija od rute Lok2, Lok1, Lok4, Lok3, pa te dvije rute s jednakim genima imaju različite vrijednosti dobrote.

Stoga je odabir metode križanja, koja će za rezultat isporučivati važeća rješenja za naš problem, od iznimne važnosti.

S obzirom da se metode križanja spomenute u prethodnom poglavlju odnose na bitovni prikaz rješenja, u ovom radu nije ih bilo moguće upotrijebiti. Odabrana metoda križanja je uređeno križanje (engl. *Order Crossover*, skraćeno *OX*). Ideja ovog križanja je da se dio gena prvog roditelja kopira u kromosom djeteta na iste pozicije, dok se preostali geni, koji nisu već u kromosomu djeteta, kopiraju iz drugog roditelja. U slučaju da se jedinka sastoji od više od jedne rute, isti proces se ponavlja za svaku rutu posebno.

Primjer takve vrste križanja s 8 lokacija te dva vozila može se vidjeti na slici 4.3. U ovom primjeru brojevi predstavljaju lokacije koje trebaju biti posjećene, dok brojka 1 predstavlja sjedište, odnosno polazno mjesto, te mjesto na kojem se ruta završava.



Slika 4.3. Uređeno križanje

Koraci odabrane metode križanja:

- Unutar svake rute prvog roditelja nasumično se odabiru početna i završna točka
- Za svaku rutu prvog roditelja, kopiraju se geni između početne i završne točke u dijete na iste te pozicije
- Potom, prolazi se kroz gene djeteta, te oni geni koji su prazni popunjavaju se genima iz drugog roditelja (osim onih gena koji već postoje u djetetu)

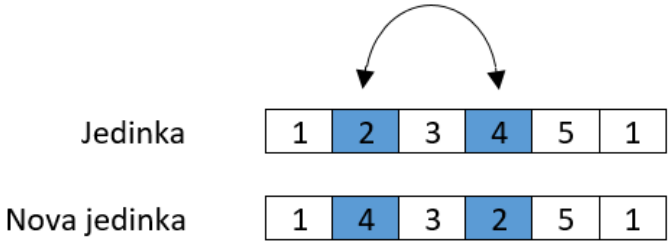
Ova metoda križanja zadržava dio poretka od roditelja, te što je najbitnije omogućava da rješenja nakon križanja ostanu važeća.

4.3.6. Mutacija

Jednako kao što je i odabir vrste križanja važan kod problema usmjeravanja vozila, tako je i veoma važno odabrati odgovarajuću vrstu mutacije. A to sve iz razloga što moramo osigurati da je rješenje važeće i nakon što na njega primijenimo operator mutacije.

Stoga, vrsta mutacije odabrana za ovu vrstu problema je metoda mutacije razmjenom (engl. *Swap Mutation*). Ideja ove vrste mutacije je vrlo jednostavna, a to je da dva različita gena

zamjene svoje pozicije, kao što se može vidjeti na slici 4.4.



Slika 4.4. Mutacija razmjenom

Algoritam prolazi kroz sve gene unutar jedinke, te je svaki gen podložan operatoru mutacije s određenom vjerojatnošću koja je definirana kao vanjski parametar algoritma. Ako je gen odabran za mutaciju, on mijenja svoju poziciju s drugim nasumično odabranim genom iz iste rute kao i odabrani gen.

Koraci odabrane metode mutacije:

- Algoritam prolazi kroz sve gene određene jedinke
- Ako je gen odabran za mutaciju, na nasumičan način odabire se drugi gen za razmjenu
- Dva odabrana gena razmjenjuju svoje pozicije unutar jedinke

Ovakva vrsta mutacije osigurava da unutar jedinke neće biti dva jednaka gena, te da će jedinka biti važeća i nakon mutacije.

5. Podešavanje parametara

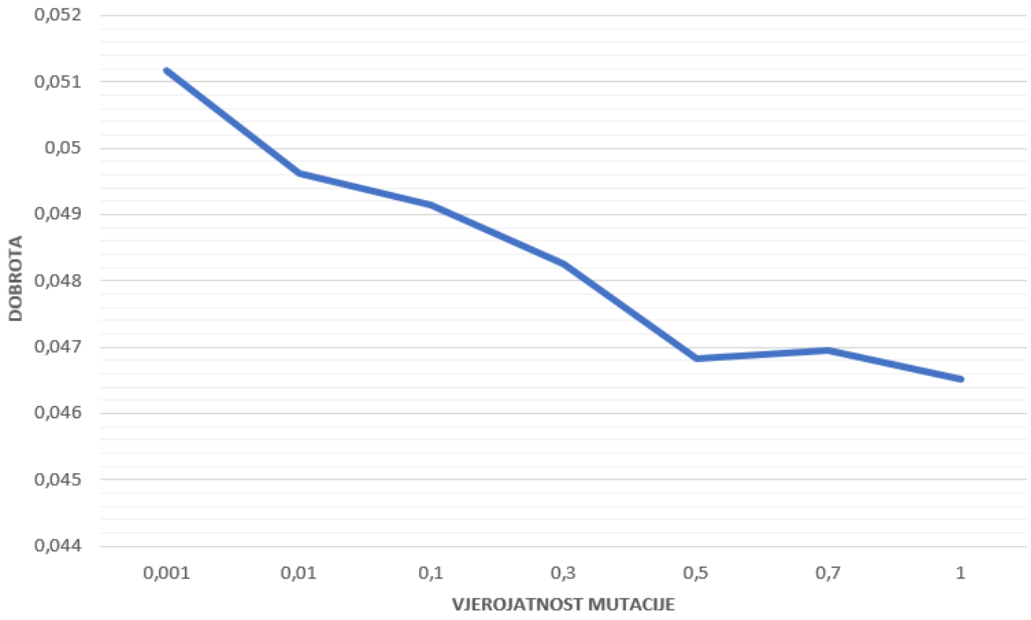
Provođenjem statistike nad dobivenim rezultatima testiranja pokušavaju se utvrditi utjecaji parametara na performanse samog algoritma, te utvrditi najbolje parametre za korištenje istog.

U ovom radu kao uzorak podataka nad kojima će se vršiti testiranje odnosno analiza uzeti su glavni gradovi svih županija Republike Hrvatske, te su oni sa svojim pripadajućim geografskim širinama i dužinama spremljeni u bazu podataka. Podaci se odnose na 19 lokacija na koje je potrebno dostaviti robu dok se sjedište, mjesto iz kojeg se dostavlja roba i u kojem sva vozila moraju završiti svoje rute, nalazi u Zagrebu.

Testiranje nad nekim skupom parametara se ponavljalo 20 puta, dok se prikaz rezultata kroz grafove odnosi na prosjek jedinki s najvećom dobrotom.

5.1. Utjecaj vjerojatnosti mutacije

Mjerenja su izvođena na podacima s već spomenutih 19 gradova odnosno lokacija na koje je potrebno dostaviti robu. Korištena su 3 vozila, odnosno algoritam je isporučio 3 rute za dostavu robe na određene lokacije. Svi parametri algoritma osim vjerojatnosti mutacije su držani fiksnim. Populacija se sastoji od 100 jedinki, vjerojatnost križanja je 0.9, broj elitnih jedinki je 3, te veličina turnira također 3. Algoritam se izvodi u 1000 generacija s mogućnošću prekida izvođenja u slučaju da nema napretka nakon 200 generacija.



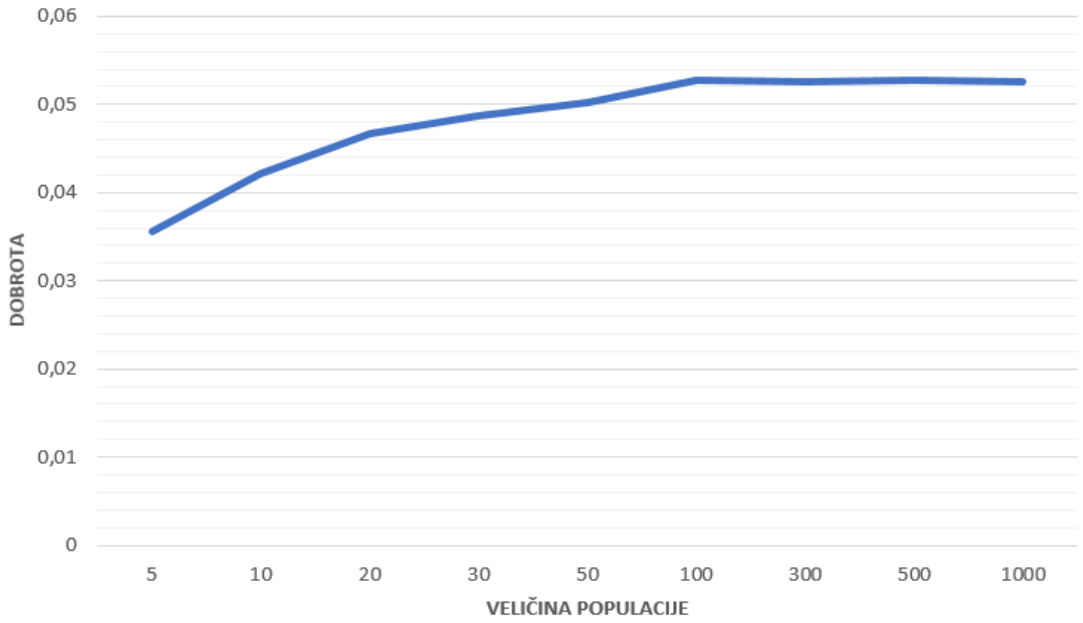
Slika 5.1. Utjecaj vjerojatnosti mutacije

Prema rezultatima provedene analize utjecaja vjerojatnosti mutacije na dobrotu jedinki, koja je prikazana na slici 5.1., možemo zaključiti da se algoritam ponaša u skladu s očekivanjima.

Bez obzira što razlike između prosječnih vrijednosti dobrota najboljih jedinki nisu velike vidi se da s povećanjem vjerojatnosti mutacije prosječna vrijednost dobrota ima tendenciju padanja. Algoritam pokazuje najbolje rezultate dok je vjerojatnost mutacije poprilično mala, oko vrijednosti 0,001, dok s povećavanjem vjerojatnosti mutacije prosječna dobrota najboljih jedinki pada. Provedeno je i testiranje s vrijednošću vjerojatnosti mutacije 1 da bi se dokazao loš utjecaj prevelike vjerojatnosti mutacije na rad algoritma. U tom slučaju, kao što je već ranije spomenuto u trećem poglavlju, algoritam prelazi u algoritam slučajne pretrage pa i isporučuje slabije rezultate.

5.2. Utjecaj veličine populacije

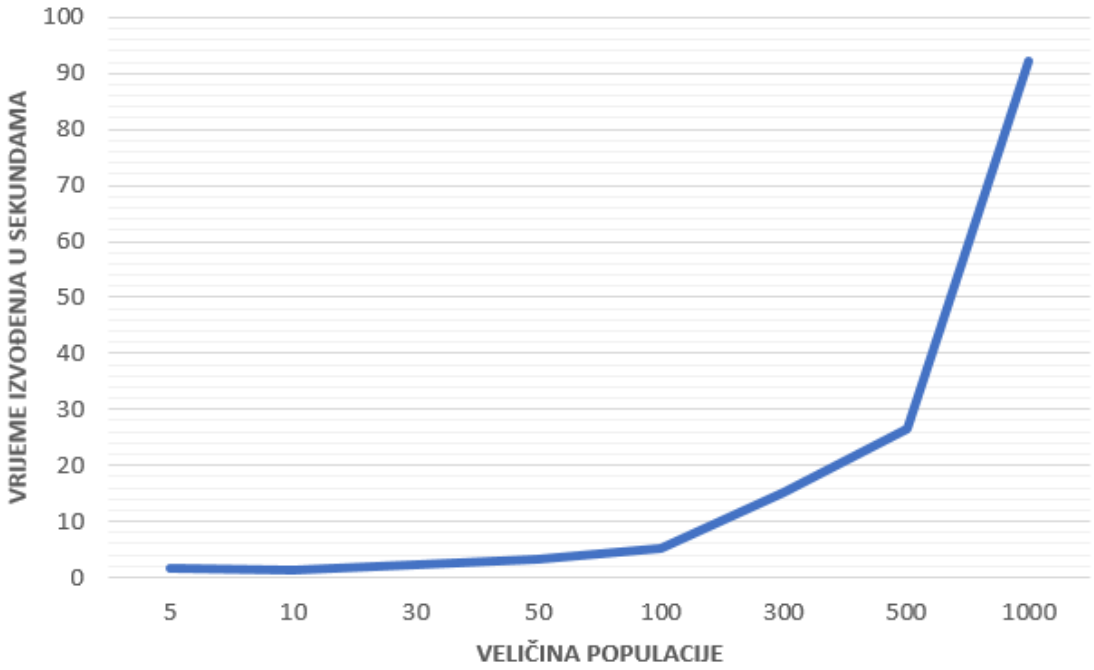
Svi parametri algoritma osim veličine populacije su držani fiksnim. Vjerojatnost mutacije je 0.001, vjerojatnost križanja je 0.9, broj elitnih jedinki je 3, te veličina turnira također 3. Algoritam se izvodi u 1000 generacija s mogućnošću prekida izvođenja u slučaju da nema napretka nakon 200 generacija.



Slika 5.2. Utjecaj veličine populacije

Provedenom analizom utjecaja veličine populacije na prosječnu dobrotu jedinke došli smo do zaključka da male veličine populacija daju loše rezultate iz razloga što u tom slučaju algoritam radi s premalom količinom genetskog materijala, a s povećanjem veličine populacije algoritam

dolazi do sve boljih rješenja. No to ne znači da sa što većom populacijom algoritam isporučuje sve bolje rezultate. U ovom slučaju, kao što je vidljivo iz slike 5.2., algoritam s populacijom od 100 jedinki daje otprilike jednake rezultate kao i populacije od 300, 500 i 1000 jedinki. A bitno je i napomenuti da s povećanjem veličine populacije, vrijeme potrebno za izvođenje algoritma se bitno povećava što je i vidljivo na slici 5.3. Tako da s obzirom na performanse algoritma i prosječnu dobrotu jedinke, u ovom slučaju algoritam isporučuje najbolje rezultate s populacijom veličine 100 jedinki.

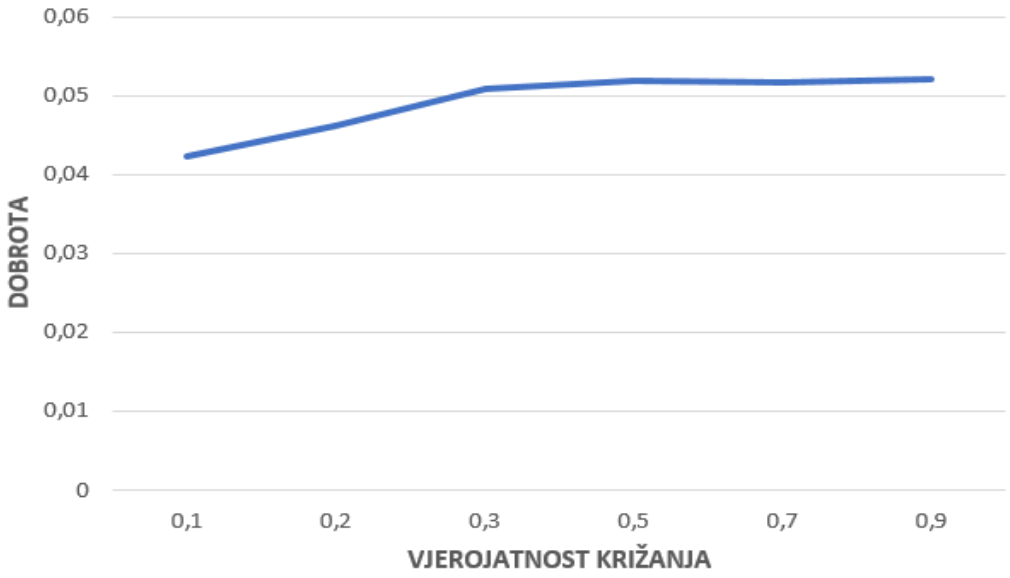


Slika 5.3. Vrijeme izvođenja s obzirom na veličinu populacije

5.3. Utjecaj vjerojatnosti križanja

Svi parametri algoritma osim vjerojatnosti križanja su držani fiksnim. Veličina populacije je 100, vjerojatnost mutacije je 0.001, broj elitnih jedinki je 3, te veličina turnira također 3. Algoritam se izvodi u 1000 generacija s mogućnošću prekida izvođenja u slučaju da nema

napretka nakon 200 generacija.



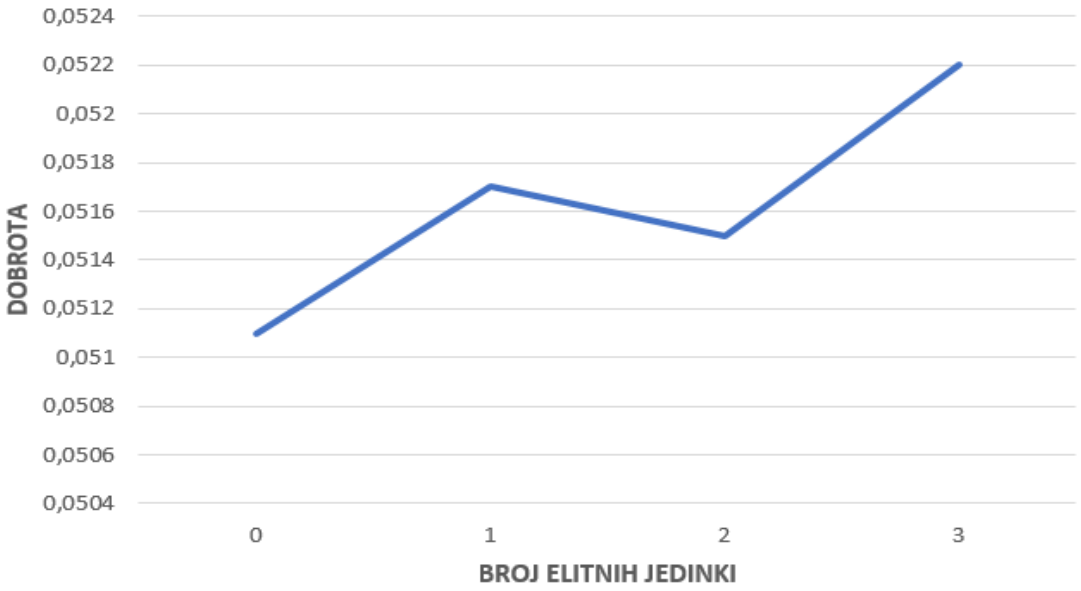
Slika 5.4. Utjecaj vjerojatnost križanja

Provedenom analizom utjecaja vjerojatnosti križanja na prosječnu dobrotu najboljih jedinki došlo se do zaključka da algoritam isporučuje najbolje rezultate kada je vjerojatnost križanja veća od 0,5, ali između manjih i većih vrijednosti vjerojatnosti križanja i nisu toliko velike razlike u prosječnoj vrijednosti dobrota najboljih jedinki. No, bez obzira što razlike nisu velike vidi se da s povećanjem vjerojatnosti križanja prosječna vrijednost dobrote raste te za vrijednosti od 0,5 do 0,9 isporučuje otprilike jednake rezultate. Algoritam pokazuje najbolje rezultate dok je vjerojatnost križanja poprilično velika, oko vrijednosti 0.9, dok u slučaju niske vjerojatnosti križanja prosječna dobrota najboljih jedinki pada.

5.4. Utjecaj elitizma

Svi parametri algoritma osim broja elitnih jedinki su držani fiksnim. Veličina populacije je 100, vjerojatnost križanja je 0.9, vjerojatnost mutacije je 0.001, te je veličina turnira 3. Algoritam se izvodi u 1000 generacija s mogućnošću prekida izvođenja u slučaju da nema napretka nakon

200 generacija.



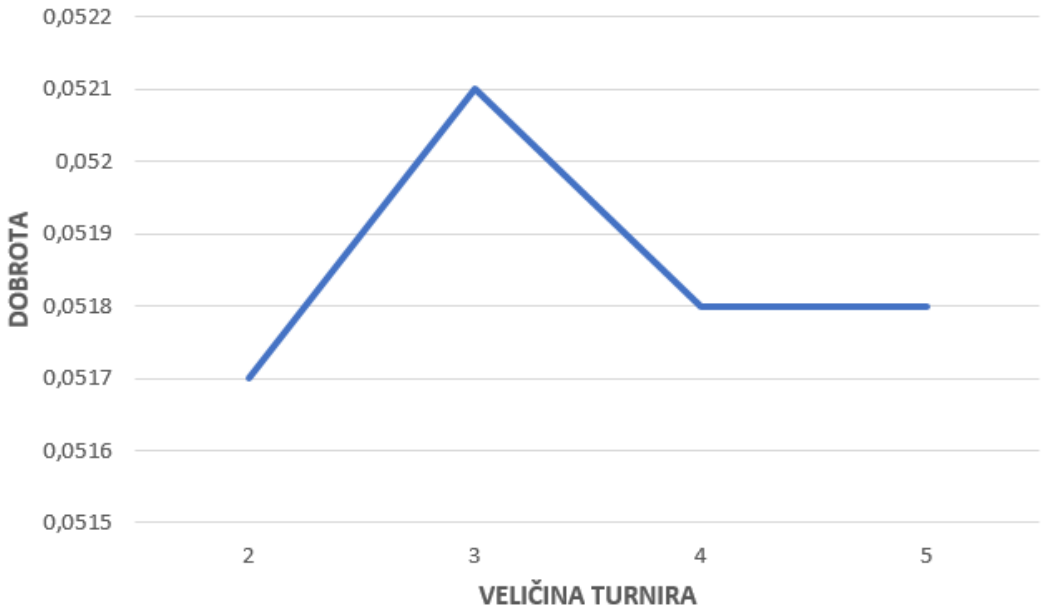
Slika 5.5. Utjecaj broja elitnih jedinki

Provedenom analizom utjecaja broja elitnih jedinki na prosječnu dobrotu najboljih jedinki došlo se do zaključka da broj elitnih jedinki nema preveliki utjecaj na prosječnu dobrotu, ali algoritam isporučuje nešto bolje rezultate kada se koristi mehanizam elitizma nego u slučaju ne korištenja istog.

5.5. Utjecaj veličine turnira

Svi parametri algoritma osim veličine turnira su držani fiksnim. Veličina populacije je 100, vjerojatnost križanja je 0.9, vjerojatnost mutacije je 0.001, te je broj elitnih jedinki 3. Algoritam se izvodi u 1000 generacija s mogućnošću prekida izvođenja u slučaju da nema napretka nakon

200 generacija.



Slika 5.6. Utjecaj veličine turnira

Provedenom analizom utjecaja veličine turnira na prosječnu dobrotu najboljih jedinki došlo se do zaključka da veličina turnira nema preveliki utjecaj na prosječnu dobrotu, no ipak algoritam isporučuje nešto bolje rezultate kada je broj jedinki koje sudjeluju u turniru 3.

Zaključak

U ovome radu prikazana je primjena genetskih algoritama na problem usmjeravanja vozila. Objašnjene su osnove genetskih algoritama, njihovo funkcioniranje te detaljno opisani njihovi sastavni dijelovi.

Ostvareno je programsko rješenje koje prema ustupljenim lokacijama, koje predstavljaju mjesta na koje je potrebno dostaviti robu, isporučuje optimalne rute prema kojima dostavne službe trebaju dostaviti već spomenutu robu.

Genetski algoritam se pokazao kao vrlo dobro rješenje u problemu usmjeravanja vozila, s obzirom da u relativno kratkom vremenu može isporučiti optimalno rješenje problema. Kod upotrebe genetskih algoritama treba biti svjestan da genetski algoritmi ne garantiraju pronalazak najboljeg rješenja. No u većini problema, koji se pokušavaju riješiti pomoću genetskih algoritama, najbolje rješenje nam i nije potrebno, nego je i zadovoljavajuće ono rješenje koje je blizu optimalnog.

Provedena je analiza utjecaja parametara na performanse algoritma te su doneseni sljedeći zaključci. Algoritam isporučuje najbolje rezultate s veličinom populacije od 100 jedinki, vjerojatnošću križanja od 0.9, vjerojatnošću mutacije od 0.001, te s korištenjem mehanizma elitizma. Analizom se došlo do zaključka da različit broj elitnih jedinki, kao i veličina turnira, ne donosi velike razlike u performansama algoritma.

Na kraju ovog rada, nakon provedene analize i testiranja programskog rješenja, može se zaključiti da su se genetski algoritmi pokazali kao vrlo učinkoviti u rješavanju problema usmjeravanja vozila.

Popis kratica

VRP	*vehicle routing problem*	problem usmjeravanja vozila
GAn	*genetic annealing*	genetsko kaljenje
AIS	*artificial immune system*	
DE	*differential evolution*	diferencijska evolucija
CVRP	*capacitated vehicle routing problem*	problem usmjeravanja vozila s ograničenim kapacitetom
VRPTW	*vehicle routing problem with time windows*	problem usmjeravanja vozila s vremenskim ograničenjem
MDVRP	*multiple depot vehicle routing*	problem usmjeravanja vozila s više skladišta
GUI	*graphical user interface*	grafički prikaz
OX	*order crossover*	uređeno križanje

Popis slika

Literatura

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

[2] HAUPT, R.L., HAUPT, S.E., *Practical Genetic Algorithms*, 2004

[3] ČUPIĆ, M., *Prirodom inspirirani optimizacijski algoritmi. Metaheuristike*, 2013

[4] MITCHELL, M., *An Introduction to Genetic Algorithms*, 1999

[5] JACOBSON, L., KANBER, B., *Genetic Algorithms in Java Basics*, 2015

[6] https://www.creativefabrica.com/wp-content/uploads/2019/08/Evolution-characters-Ape-human-black-580x386.jpg

[7] CARIĆ, T., GOLD H., *Vehicle Routing Problem*, 2008