GENETSKI ALGORITAM ZA RJEŠAVANJE ASIMETRIČNOG PROBLEMA TRGOVAČKOG PUTNIKA

Vulić, Lukrecia

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studiji

GENETSKI ALGORITAM ZA RJEŠAVANJE ASIMETRIČNOG PROBLEMA TRGOVAČKOG PUTNIKA

Završni rad

Lukrecia Vulić

Osijek, 2021

IZJAVA O ORIGINALNOSTIRADA

Osijek, 10.09.2021.

Ime i prezime studenta:	Lukrecia Vulić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4298, 26.07.2018.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Genetski algoritam za rješavanje asimetričnog problema trgovačkog putnika**

izrađen pod vodstvom mentora Doc.dr.sc. Zdravko Krpić

i sumentora Doc.dr.sc. Dražen Bajer

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek,02.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Lukrecia Vulić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4298, 26.07.2018.
OIB studenta:	76672914644
Mentor:	Doc.dr.sc. Zdravko Krpić
Sumentor:	Doc.dr.sc. Dražen Bajer
Sumentor iz tvrtke:	
Naslov završnog rada:	Genetski algoritam za rješavanje asimetričnog problema trgovačkog putnika
Znanstvena grana rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	02.09.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

Sadržaj

1. UVOD

Problem trgovačkog putnika je poznat po brojnim imenima te je jedan od najpoznatijih kombinatornih problema. Problem zahtjeva pronalazak slijeda obilaska gradova i povratak u početni tako da put bude minimalan. Ovaj problem je vrlo istražen problem u području optimizacije. Problem je fascinirao mnoge jer iako samo opisan zvuči jednostavno za riješiti, ovaj problem spada pod NP-teške probleme kombinatorne optimizacije, te je težina opisana kao O(N!), gdje N predstavlja broj gradova. Problem je relativno jednostavan za riješiti kada je u pitanju manji broj gradova, no porastom broja gradova povećava se kompleksnost pa time i vrijeme potrebno da se pronađe optimalno rješenje. Kako je problem razmatran godinama pojavile su se razne varijante problema, jedna od kojih je asimetrični problem trgovačkog putnika. Asimetrična inačica razlikuje se od simetrične po tome što put od jednog do drugog grada ne mora biti jednak obrnutom putu. Postoje mnoge primjene i simetrične i asimetrične inačice bili to problemi koji se izravno povezuju s problemom trgovačkog putnika ili se svode na taj problem.

Kako problem spada pod NP-teške optimizacijske probleme ne postoji algoritam koji ga rješava u polinomijalnom vremenu, već vrijeme rješavanja raste eksponencijalno s povećanjem složenosti problema. Velik broj primjena problema trgovačkog putnika potaknulo je pronalazak raznih algoritama koji su uspješni u pronalasku rješenja, jedan od kojih je genetski algoritam. Genetski algoritmi pripadaju familiji evolucijskih algoritama i najpoznatiji su tip tih algoritama. Algoritmi ove familije imitiraju proces evolucije te postoje razne varijante varijacijskih operatora koji su prikladni za rješavanje razmatranog problema.

1985. godine osmišljena je prva verzija genetskog algoritma za problem trgovačkog putnika te se vremenom implementacija mijenjala tako da se prilagodi pronalasku rješenja i za specifične verzije problema trgovačkog putnika pod koje spada asimetrična inačica. Ovakav pristup problemu već se pokazao uspješnim u pronalasku prihvatljivih rješenja za ovaj problem i njegove varijacije. Također je, s obzirom na broj mogućih rješenja, bitna stavka odabira algoritma vrijeme izvršavanja te se i u ovom pogledu genetski algoritam pokazao kao prikladan odabir.

U poglavlju 2 detaljnije je opisan asimetrični problem trgovačkog putnika te genetski algoritam. Također su opisane neke druge inačice problema. Način rada genetskog algoritma je detaljno opisan ujedno s principima rada operatora korištenih u programskom rješenju. Također su opisani

parametri koji utječu na rad algoritma. U poglavlju 3 opisano je ostvareno programsko rješenje. Opisano je na koji je način implementiran genetski algoritam u odnosu na problem koji rješava, koji su ulazi i izlazi iz programa te koji se operatori koriste za što efikasnije i fleksibilnije korištenje programskog rješenja. Nadalje, u poglavlju 4 provedena je eksperimentalna analiza implementiranim programskim rješenjem nad odabranim primjercima problemima asimetričnog trgovačkog putnika. Eksperimentalnom analizom provjerena je učinkovitost određenih operatora mutacije pri različitim vrijednostima vjerojatnosti mutacije.

1.1. Zadatak završnog rada

U radu je potrebno opisati problem trgovačkog putnika, njegovu asimetričnu inačicu kao i neke druge inačice. Uz to, potrebno je opisati genetski algoritam kao i njegovu primjenu za rješavanje razmatranog problema. U praktičnom dijelu rada nužno je razviti programsko rješenje koje predstavlja inačicu genetskog algoritma za pronalazak rješenja asimetričnog problema trgovačkog putnika. Također, potrebno je napraviti eksperimentalnu analizu na nekoliko primjeraka problema iz literature.

2. ASIMETRIČNI PROBLEM TRGOVAČKOG PUTNIKA I GENETSKI ALGORITAM

Asimetrični problem trgovačkog putnika predstavlja optimizacijski problem kojem je cilj smanjiti duljinu puta obilaska gradova. Primjene u stvarnom svijetu,prema [1, str. 43],uključuju probleme kao što su problem skupljanja kovanica na govornicama i problem pokretne trake bez čekanja. Iako asimetrična varijanta problema ima razne primjene u praksi, prema [1, str. 33], puno je manje istražen od simetrične varijante. Neki od načina rješavanja problema su algoritam najbližeg susjeda i drugi pohlepni algoritmi koji svoje varijante imaju i kod simetrične varijante problema. Učinkovit pristup rješavanju ovog problema nude evolucijski algoritmi te bi kod ovog problema bio prikladan genetski algoritam.

Algoritmi iz ove familije vuku inspiraciju iz procesa evolucije. Kako je asimetrični trgovački putnik optimizacijski problem poznato je da je željeni rezultat minimalna duljina puta obilaska gradova. Problem se dakle svodi na pretraživanje mogućih rješenja i pronalazak najboljeg u nekom okvirnom vremenu ili pod nekim drugim uvjetom završetka. Genetski algoritam, prema [2, str. 100], je najpoznatija varijanta evolucijskih algoritama te je zbog korištenja za rješavanje raznih kombinatornih problema bio prikladan za primjenu i na ovom problemu.

2.1. Asimetrični problem trgovačkog putnika

Problem trgovačkog putnika zadan je kao N gradova i za svaki par gradova c_i i c_j zadana je udaljenost $d(c_i, c_j)$ između njih. Prema [1, str. 32], rješenjem problema smatra se permutacija π koja minimizira izraz (2-1).

$$\sum_{i=1}^{N-1} d(c_\pi(i), c_\pi(i+1)) + d(c_\pi(N), c_\pi(1)) \qquad (2\text{-}1)$$

Ovaj problem također spada pod NP-teške kombinatorne optimizacijske probleme. Težina problema dolazi iz stavke permutacije gradova čiji potencijalni broj rješenja raste eksponencijalno, jer se broj permutacija računa na način N!. Zbog velikog broja mogućih rješenja problem je doseći optimalno rješenje u prihvatljivom vremenskom periodu u slučaju većeg broja gradova.

Prema [3, str. 9], primjena problema trgovačkog putnika i njegovih varijacija proteže se daleko izvan planiranja puta već se primjenjuje u područjima matematike, računarstva, genetike, inženjerstva i elektronike. Kao neke konkretne primjene mogu se navesti sljedeći primjeri: problem kreiranja rasporeda, strukturiranje matrica i u staničnoj proizvodnji.

Inačica koja se razmatra u ovom radu je asimetrični problem trgovačkog putnika. Prema [3, str. 446], razlika između simetrične i asimetrične inačice je u tome što je kod simetrične inačice prethodno navedena udaljenost $d(c_i, c_j)$ je jednaka udaljenosti $d(c_j, c_i)$. Kod asimetrične inačice ovo nije slučaj već se ove udaljenosti mogu razlikovati. Prema [3, str. 29], za simetričnu inačicu problema može se reći da je poseban slučaj asimetrične inačice. Ovo je naravno izazvalo razlike u pronalasku rješenja za probleme. Također valja reći kako se većina napora za rješavanje problema usmjerilo na simetričnu inačicu te za asimetrični problem postoji puno manje radova.

2.1.1. Neke druge varijante problema trgovačkog putnika

Prema [3, str. 7], različite varijante ovog problema su se pojavile primjenom jednostavnih transformacija i imaju svoju primjenu u raznim područjima. Ovdje su nabrojane samo neke varijante trgovačkog putnika, no postoje mnoge. Prema [3, str. 7], jedna od varijacija naziva se maksimalni problem trgovačkog putnika iz čijeg se imena može zaključiti da se traži maksimalan mogući put obilaska gradova. Može se riješiti isto kao običan problem, jedina razlika je što umjesto zbrajanja duljine puta između gradova, duljine se oduzimaju. Kao primjer primjene može se navesti problem najduljeg podniza koji dalje ima svoje primjene u DNK sekvenciranju i kompresiji podataka.

Nadalje, prema [4], postoji varijacija problema, selektivni trgovački putnik, u kojem je cilj pronaći redoslijed posjećivanja gradova tako da se maksimizira profit posjećivanja uz troškove putovanja koji ne prelaze nekakvu predodređenu vrijednost. Ovakva varijanta problema koristi se kod problema usmjeravanja zaliha i kod kontekstnog orijentacijskog natjecanja [4].

Generalizirani problem trgovačkog putnika je varijanta klasičnog simetričnog trgovačkog putnika gdje su gradovi podijeljeni u skupine te trgovac iz svake skupine mora posjetiti barem jedan grad iz svake skupine. Postoji također drugačija verzija ove varijante u kojoj trgovac mora posjetiti točno jedan grad iz svake skupine [4]. Obje ove verzije bitne su kod rješavanja kritičnih problema koji uključuju istovremene odluke o odabiru i sekvenciranju.

Prema [3, str. 7], također postoji varijanta koja ima višestruku posjetu gradova. Problem je opisan tako da trgovac krene iz nekog početnog grada, obiđe sve gradove barem jednom i vrati se u početni. Ovaj problem se može svesti na standardni problem trgovačkog putnika.

2.2. Genetski algoritam

Genetski algoritam je najpoznatija varijanta evolucijskih algoritama i oponašaju genetsku evoluciju. Prema [5, str. 143], jedinke su prikazane korištenjem genotipa te su glavni operatori koji unaprjeđuju populaciju selekcija, križanje i mutacija. Prema [2, str. 99], genetski algoritmi uglavnom se sastoje od koraka prikazanih pseudokodom na slici 2.1.

```
ZAPOČNI
    INICIJALIZACIJA populacije sa nasumičnim kandidat rješenjima
    VREDNOVANJE svakog kandidat rješenja
    PONAVLJANJE DOK (nije zadovoljen uvjet završetka)
        1 ODABIR roditelja
        2 KRIŽANJE  parova roditelja
        3 MUTACIJA dobivenih potomaka
        4 VREDNOVANJE novih kandidata
        5 ODABIR kandidata za novu generaciju
```

Slika 2.1. Shema genetskog algoritma u pseudokodu

Prvi korak kod postavljanja algoritma je odabir načina na koji će se predstavljati moguća rješenja. Prema [2], moguća rješenja u kontekstu zadanog problema nazivaju se fenotipovi, a njihovo kodiranje u genetskom algoritmu nazivaju se genotipovi. Svaka jedinka također ima svoju kvalitetu koja predstavlja skup poželjnih karakteristika za preživljavanje i ovo ovisi o problemu koji se rješava. Genotipovi se čuvaju u populaciji koja je također jedinica evolucije. Populacija se definira svojom veličinom, odnosno brojem jedinki koje sadržava te je definirana kao multiskup što znači da je moguće da u sebi sadrži dvije ili više istih jedinki. Veličina populacije je konstantna kako bi se imitirala ograničenost resursa koji se pojavljuje u prirodi i koji igra ulogu u opstanku jedinke. Prema [2, str. 31], još jedna mjera populacije je njena raznolikost što se odnosi na broj različitih jedinki prisutnih u populaciji. Raznolikost nema jedinstvenu mjeru, već se može gledati po kvalitetu, fenotipu ili genotipu. Prema [2, str. 30], kvaliteta genotipa vrednuje se funkcijom dobrote. Ovom funkcijom se definira što to znači napredak populacije jer ima ulogu predstaviti standarde koje populacija mora dostići s ciljem preživljavanja. Funkcija dobrote svakom genotipu dodjeljuje mjeru kvalitete.

Nadalje, prema [2, str. 31], temeljni korak u algoritmu je selekcija roditelja za sljedeću generaciju. Uloga ovog operatora je da se, na temelju razlika u kvaliteti, odabiru roditelji koji će stvarati potomke. Roditelj se definira kao pojedinac koji je odabran da prođe kroz operatore varijacije kako bi stvorio potomke. Selekcija roditelja je u pravilu pristrana, te jedinke s većom kvalitetom imaju veće šanse da postanu roditelji od onih s manjom.

Operatori varijacije imaju ulogu stvarati nove jedinke od starih i postoje dva tipa unarni i n-arni. Mutacija je unarni operator varijacije. Primjenjuje se na jedan genotip te kao rezultat daje malo izmijenjene genotipove. Prema [2, str. 32], operator mutacije je uvijek stohastički što znači da potomak uvijek ovisi o nizu nasumičnih izbora. Operator treba obaviti nasumičnu nepristranu promjenu. U genetskim algoritmima operator mutacije se koristi kao pozadinski mehanizam koji u populaciju dovodi raznolikost. Mutacija ima svoju vjerojatnost događanja koja je ujedno parametar algoritma.

Operator križanja je binarni operator koji spaja informacije dva roditeljskih genotipa u jedan ili dva potomka. Također je stohastički operator jer je nasumično odabrano koji dijelovi i kako će se spojiti. Cilj križanja je da se križanjem dvije jedinke, od kojih oboje imaju poželjne osobine u kontekstu problema, dobije dijete koje će naslijediti neke,ili u najboljem slučaju sve, dobre osobine roditelja. Vjerojatnost križanja je parametar algoritma te opisuje vjerojatnost događanja križanja nad parom roditelja.

Kao kod selekcije roditelja, selekcija preživjelih ima ulogu razlikovati jedinke na temelju kvalitete, ali nakon stvaranja potomaka. Potreba za ovim mehanizmom dolazi iz toga što je veličina populacije konstantna te je nakon svakog ciklusa stvaranja potomaka napraviti selekciju kako bi se broj jedinki u populaciji smanjio. Odluka o tome koje jedinke ostaju u populaciji temelji se na kvaliteti pojedinih jedinki gdje se preferiraju jedinke s boljom kvalitetom u kontekstu problema. Za razliku od roditeljske selekcije, selekcija preživjelih je deterministička.

Uvjet završetka također je parametar algoritma i ima dva tipa. Prvi tip je, prema [2, str. 34], onaj kojemu je poznat optimalna kvaliteta i poznato je kada algoritam treba stati. Evolucijski algoritmi su stohastički i nije sigurno da će se dostići optimalno rješenje i u tom slučaju bi se algoritam izvodio zauvijek. U ovakvim slučajevima se mora se postaviti konačan uvjet završetka i neki od primjera takvog uvjeta su: fiksan broj ponavljanja, raznolikost populacije padne ispod određenog

praga i poboljšanje kvalitete ostaje ispod određenog praga tijekom određenog vremenskog perioda.

Asimetrični problem trgovačkog putnika već je uspješno rješavan korištenjem genetskog algoritma. Primjer ovoga može se pronaći u [6] gdje se testira efikasnost rješenja koje koristi posebni operator križanja. Nadalje, jedan od načina pristupa problemu se pronalazi u članku [7] gdje se koriste razne inovativne metode za rješavanje i simetrične i asimetrične inačice. Iako se asimetrična inačica često pojavljuje u praksi manje je istražena nego ona simetrična. Simetrična inačica detaljno je razrađena u radovima kao što su [8], [9] i [10].

2.2.1. Primjena genetskog algoritma za rješavanje problema trgovačkog putnika

Genetski algoritam se primjenjuje na problem trgovačkog putnika tako da se svaki dio prethodno opisan prilagodi potrebama problema. Pojam jedinke u slučaju ovog problema odnosi se na permutaciju gradova koji se obilaze te duljinu puta između njih koja predstavlja kvalitetu jedinke, što je manji put to je kvaliteta jedinke veća. Populacija se sastoji od ovih jedinki te ima konstantnu veličinu. Algoritam započinje generiranjem nasumične populacije koja u sebi sadržava zadan broj jedinki. Funkcijom dobrote računa se kvaliteta permutacije preko izraza (2-1). Ova funkcija se koristi na dva mjesta u algoritmu, jednom u ocjenjivanju početne populacije i kasnije kod ocjenjivanja rješenja dobivenih kroz operatore križanja i mutacije. Glavna petlja algoritma ima zadani uvjet završetka koji je za ovaj problem zadan broj ponavljanja.

Sljedeći korak je odabir roditelja iz populacije. Jedan od mehanizama za odabir roditelje naziva se turnirska selekcija. Ovaj mehanizam, prema [2, str. 85], ne zahtijeva globalno znanje o populaciji, što znači da razliku od selekcija poput selekcije poretka koja sortira jedinke po kvaliteti i uzima one najbolje, turnirska selekcija ima znanje samo o nekolicini odabranih rješenja koja potom uspoređuje. Veličina turnira definirana je parametrom k koji određuje koji broj jedinki se odabire iz populacije. Bitna stavka kod odabira veličine turnira je što je veći parametar k to je veća vjerojatnost da će se odabrati jedinka čija je kvaliteta veća od prosječne vrijednosti kvalitete pa se time smanjuje vjerojatnost odabira jedinki niže kvalitete. Ukoliko se radi o selekciji bez zamjene, što znači da jedinka može samo jednim primjerkom sudjelovati u danom turniru, osigurava se da k-1 najgorih jedinki, odnosno one s najmanjom kvalitetom, nikad ne odaberu.

Na slici 2.2 prikazana je punjenje bazena za parenje (engl. *Mating pool*) pseudokodom [2, str. 85]. Sve jedinke odabrane su turnirskom selekcijom, kao što je vidljivo sa slike, te stavljaju se u *matingpool*. Ovdje se radi o selekciji sa zamjenom što znači da ista jedinka može s više primjeraka sudjelovati u turniru. U slučaju ovog tipa algoritma potrebno je odabrati broj roditelja jednak veličini populacije. Parametar k također se može podešavati prema potrebi. Parametar λ predstavlja broj roditelja koji se odabiru te je u ovom slučaju jednak veličini populacije odnosno parametru μ.

```
ZAPOČNI
        POSTAVI trenutni_član = 0
        PONAVLJANJE DOK (trenutni_član ≤λ)
                ODABERI k jedinki nasumično
                USPOREDI k jedinki i odaberi najbolju
                POSTAVI tu jedinku kao i
                POSTAVI mating_pool[trenutni_član] = i
                POSTAVI trenutni_član = trenutni_član + 1
```

Slika 2.2. Pseudokod punjenja bazena za parenje

```
ZAPOČNI
        ODABRATI nasumičan segment iz prvog roditelja i kopirati u dijete
        POČETKOM      u prvoj točci prelaska tražiti elemente u tom segmentu drugog roditelja koji nisu kopirani
        ZA SVAKI takav ( i ) potražiti u djetetu koji element ( j) je kopiran na to mjesto iz prvog roditelja
        POSTAVITI  element i na mjesto na kojemu je element j u drugom roditelju
        AKO je mjesto na kojem je j već ispunjeno elementom (k) postaviti i na mjesto na kojem je element k u drugom roditelju
        UČINITI ovo sa svim elementima u odabranom segmentu te preostala mjesta u djetetu popuniti preostalim elementima drugog roditelja
```

Slika 2.3. Pseudokod djelomično preslikanog križanja

Daljnji korak algoritma je križanje roditelja. Kako je redoslijed posjećivanja gradova u problemu trgovačkog putnika permutacija, jer se svaki grad smije posjetiti samo jednom, križanje ovo ne smije narušiti. Ovo znači da nakon križanja jedinka također mora biti permutacija. Iz ovog razloga

osmišljeni su posebni operatori križanja pa time i posebne varijante operatora križanja za permutacijsko kodiranje. Jedan od kojih je djelomično preslikano križanje koji je prikazan pseudokodom na slici 2.3 [2, str. 70]. Pseudokod na slici 2.3 opisuje stvaranje jednog djeteta, a drugo se stvara analogno zamjenom pozicije roditelja. Slika 2.4 prikazuje primjer rada ovog operatora [2, str. 71].

Sljedeći korak u algoritmu, prema [2], je mutacija rješenja dobivenih operacijom križanja. Vjerojatnost mutacije određuje koliko je izgledno da se operator mutacije primijeniti na potomke. Mutacija je unarni operator i djeluje samo na jedno rješenje. Ovaj operator također mora funkcionirati na specifičan način zbog permutacijskog kodiranja. Operator ne smije narušiti valjanost permutacije i zbog toga su osmišljene posebne varijante prikladnih mutacija i to su: mutacija zamjene, mutacija umetanja i mutacija premetanja. U ovom algoritmu koriste se prva dva spomenuta. Mutacija zamjene, čiji primjer je prikazan slikom 2.5, radi na takav načina da se nasumično odaberu dvije vrijednosti i zamjene [2, str. 69]. Mutacija umetanja, čiji primjer je prikazan slikom 2.6, također odabire dvije nasumične vrijednosti te se jedna ubacuje kraj druge pomičući cijeli poredak [2, str. 69].

Slika 2.4. Primjer rada djelomično preslikanog križanja

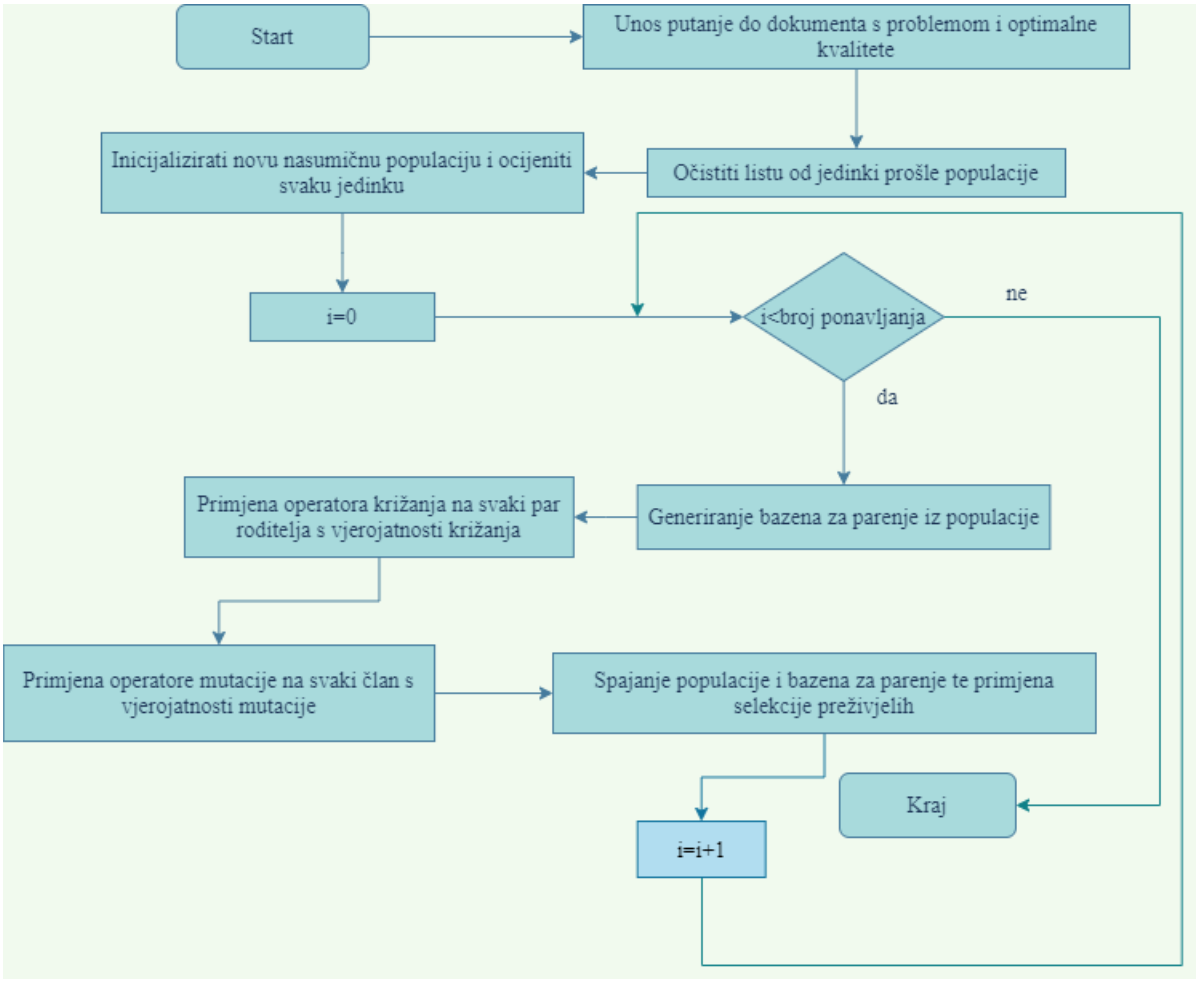
Slika 2.5. Primjer mutacije zamjene

Slika 2.6. Primjer mutacije umetanja

Posljednji korak u algoritmu je selekcija preživjelih. Postoje razni načini na koji se ovo postiže jedan od kojih je, prema [2], (μ + λ) selekcija. U ovom slučaju spajaju se multiskupovi populacije i djece prethodno nastale te se poredaju prema kvaliteti. μ predstavlja broj jedinki dopuštenih u populaciji, a λ broj djece. Selekcija nakon poretka svih jedinki prema kvaliteti uzima samo μ najboljih te se ovim putem osigurava da je svaka sljedeća populacije bolja ili jednaka onoj prethodnoj.

3. OSTVARENO PROGRAMSKO RJEŠENJE

Programsko rješenje ugrađuje genetski algoritam za rješavanje asimetričnog problem trgovačkog putnika te je odabran objektno-orijentirani dizajn. Program je napisan u programskom jeziku C# u integriranoj razvojnoj okolinu Visual Studio 2019. U rješavanju su također korišteni oblikovni obrasci kako bi se omogućila fleksibilnost i kako bi se lakše prikazao tok programskog rješenja. Na slici 3.1 prikazan je dijagram toka genetskog algoritma.

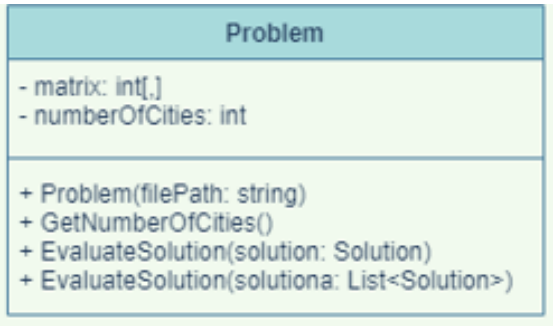


Slika 3.1. Dijagram toka implementiranog genetskog algoritma

3.1. Način rada programskog rješenja

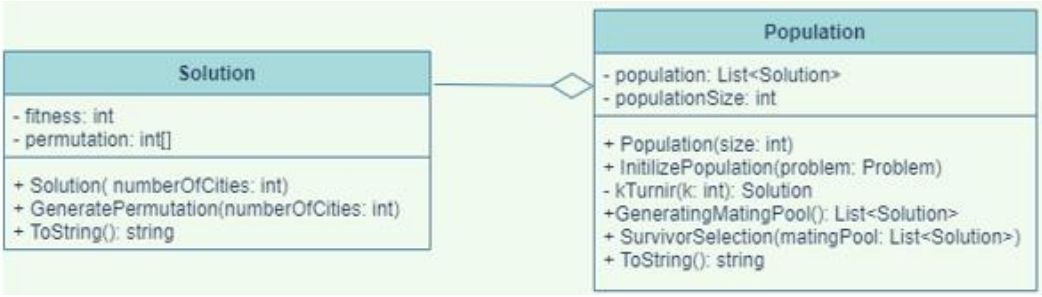
Asimetrični problem trgovačkog putnika predstavljen je klasom *Problem.* Klasa ima zadaću dohvaćanja podataka o problemu, koji uključuju matricu i broj gradova u problemu, također u sebi sadrži funkciju dobrote. Dijagram klase *Problem* prikazan je slikom 3.2. Permutacije gradova i njihova kvaliteta zapakirani su u zasebnu klasu *Solution* koja uz zadaću čuvanja ovih podataka

ima zadaću generiranja nasumičnih permutacija gradova koje je potrebno za inicijalizaciju početne populacije.



Slika 3.2. Dijagram klase *Problem*

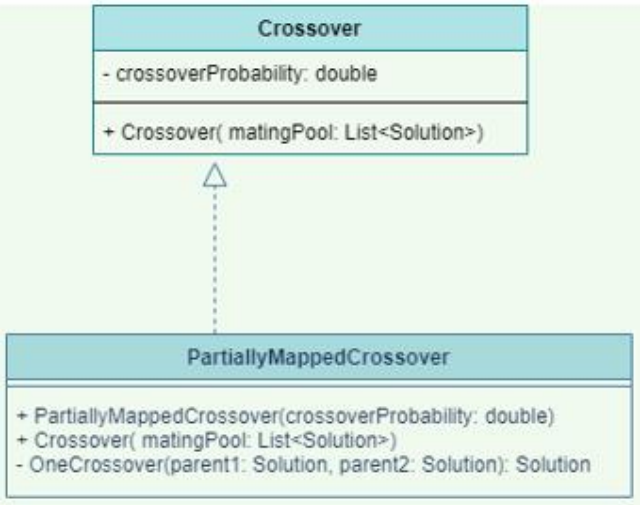
Nadalje, temeljna stavka genetskog algoritma, što je populacija, također je predstavljena u svojoj klasi *Population* te čuva podatke o svim potencijalnim rješenjima te o veličini populacije koja se zadaje od strane korisnika. Klasa populacija ima zadaću inicijalizirati početnu populaciju, oboje permutaciju i kvalitetu te je zato potrebno kao referencu poslati klasu *Problem*, koja kao što je već rečeno ocjenjuje kvalitetu permutacija. Generiranje permutacije, kao što je ranije rečeno obavlja se u klasi *Solution*. Klasa *Population* isto ima zadaću generiranja bazena za razmnožavanje koji se obavlja turnirskom selekcijom te selekciju preživjelih. Dijagram klasa *Solution* i *Population* prikazan je slikom 3.3.



Slika 3.3. Dijagram klasa *Solution* i *Population*

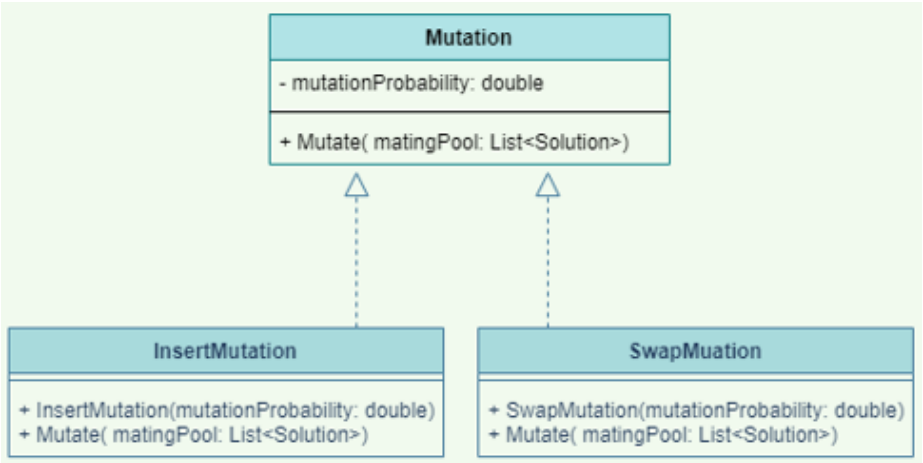
Sljedeći implementirani dio je operator križanja koji je ostvaren pomoću obrasca ponašanja strategije [11, str. 351]. Ovo je učinjeno jer, kao što je ranije navedeno, ima više vrsta operatora križanja te ovaj obrazac savršeno ugađa fleksibilnosti promjene različitih vrsta operatora. U slučaju ove implementacije ugrađen je samo jedan tip križanja, ali zbog mogućih budućih proširenja korišten je ovaj obrazac. Implementacija obrasca strategije učinjena je kroz apstraktnu

klasu *Crossover* koje propisuje metodu križanja i atribut koji predstavlja vjerojatnost križanja. Klasa koja predstavlja željeni tip križanja *PartiallyMappedCrossover* koja tu metodu implementiran na način prethodno opisan u poglavlju 2.2.1. Na slici 3.4 ove klase su prikazane klasnim dijagramom.



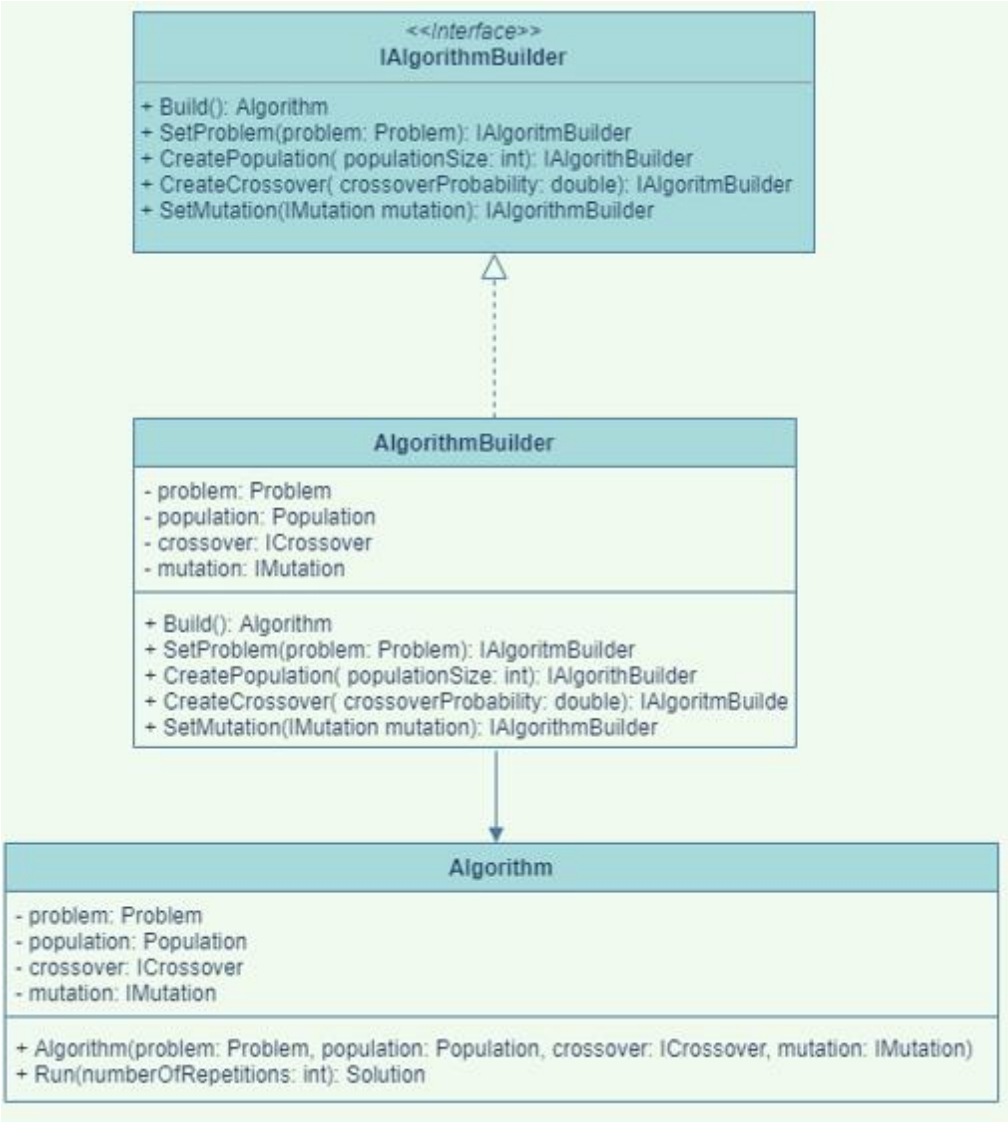
Slika 3.4. Dijagram klasa *Crossover* i *PartiallyMappedCrossover*

Operator mutacije također je implementiran pomoću obrasca strategije prema [11, str. 351] i ideja iza toga je ista kao i za križanje. U ovom slučaju čak je prikladnije jer su implementirana dva tipa mutacije. Apstraktna klasa također propisuje jednu metodu mutacije i sadrži atribut vjerojatnosti mutacije i implementirane su klase *SwapMutation* i *InsertMutation* čiji su načini rada opisani u poglavlju 2.2.1 te čiji su klasni dijagrami prikazani slikom 3.5.



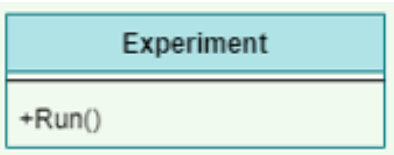
Slika 3.5. Dijagram klasa *Mutation, SwapMutation* i *InsertMutation*

Algoritam se sastoji od ovih komponenti pa je iz tog razloga prikladno odabran obrazac stvaranja graditelj kako bi se ovo ostvarilo. Obrazac se sastoji od sučelja *IAlgorithmBuilder*koje propisu od kojih se točno komponenti mora sastojati gotov proizvod. Nadalje se implementira klasa *AlgoritmBuilder* koja implementira metode propisane sučeljem na način koji odgovara danom problemu. U ovom slučaju objekti klase *Problem, SwapMutation* i *InserMutation* su kreirani izvan graditelja radi jednostavnosti obavljana eksperimenta spomenutog kasnije u radu. Krajnji proizvod koji daje graditelj je klasa *Algorithm* koja ima svoj konstruktor i metodu za pokretanje algoritma. Opisana struktura prikazana je na slici 3.6. Ova metoda kao rezultat vraća najbolji rezultat koji je algoritam pronašao te je dijagram toka algoritma prikazan slikom 3.1.



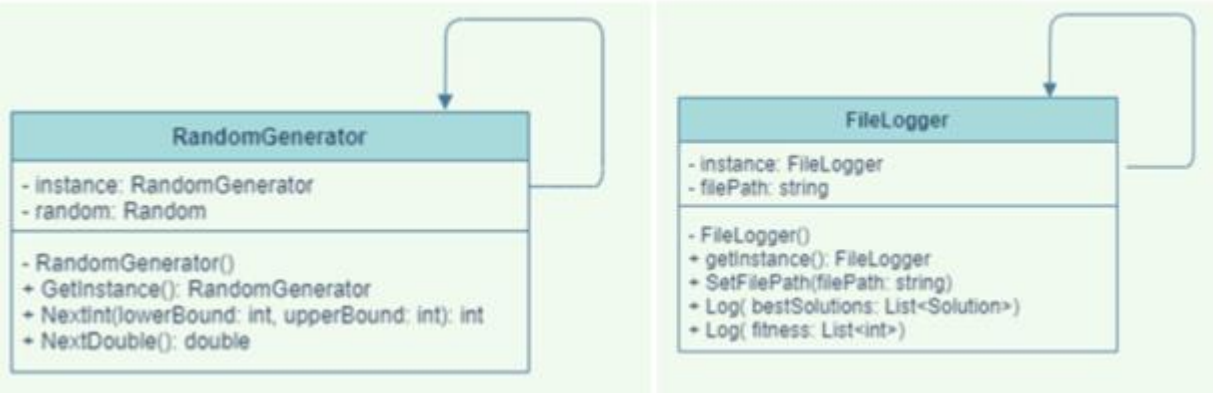
Slika 3.6. Dijagram obrasca graditelja za stvaranje klase *Algorithm*

Genetski algoritam je stohastičan te samo jedan najbolji pronađeni rezultat ništa ne govori zato je još stvorena klasa *Experimet* u kojoj se obavlja određen broj ponavljanja algoritma za specifične probleme i specifične promjene parametara. Klasa sadrži samo jednu metodu kojom se izvršava eksperiment i prikazana je klasnim dijagramom na slici 3.7. Za ovu klasu se može reći da je fasada. Prema [11, str. 208.], fasada je oblikovni obrazac strukture koji opisuje sučelje iza kojeg se skriva kompleksna struktura.



Slika 3.7. Dijagram klase *Experiment*

Uz sve ove komponente pojavljuju se i klase *RandomGenerator* i *FileLogger*koje su singeltoni. Klasa *RandomGenerator* se koriste da generira nasumične brojeve koji su potrebni na nekoliko mjesta u algoritmu i ovakva klasa standardno se implementira kao singelton kako bi se mogla dokučiti na bilo kojem mjestu u programu i za kod takvih klasa nema potrebe za više instanci. Ista stvar se može reći i za klasu *FileLogger*. Klasa *FileLogger* ima zadaću upisivati najbolja rješenja koja se pronađu u tekstualnu datoteku. Ove klase su prikazane dijagramom klasa na slici 3.8.



Slika 3.8. Dijagram klasa *RandomGenerator* i *FileLogger*

3.2. Način uporabe programskog rješenja

Korisnik programsko rješenje koristi kroz objekt klase *Experiment* u kojoj su već postavljene vrijednosti za eksperiment koji se obavlja u ovom radu. Pozivom funkcije *Run()* na objektu ove

klase pokreće se eksperiment te se ovo obavlja u glavnoj funkciji programa. Ovo je prikazano slikom 3.9. Korisnik je također u mogućnosti koristiti programsko rješenje iza klase *Experiment*, odnosno sam određivati parametre koje želi testirati.

```
static void Main(string[] args)
{
    Experiment experiment = new Experiment();
    experiment.Run();
}
```

Slika 3.9. Prikaz stvaranja i korištenja klase *Experiment*

Ako korisnik odluči sam stvarati svoj algoritam korištenjem klase *AlgorithmBuilder* potrebno je stvoriti objekte sljedećih klasa: *Problem* i *Mutation*. Ovakav dizajn je odabran jer se ove dvije stvari mijenjaju u eksperimentu obavljenom u poglavlju 4 te je ove stavke potrebno po potrebi mijenjati izvana. Sve druge klase, opisane u poglavlju 3.2, već su ugrađene u navedene klase. Objekt klase *Problem* stvara se pomoću konstruktora koji kao parametar prima *String* koji predstavlja putanju do tekstualne datoteke u kojoj je zadan problem kroz matricu u kojoj se nalaze međusobne udaljenosti te broju gradova u problemu. Prikaz stvaranja ovog objekta prikazan je slikom 3.10. Ovu stavku je također moguće mijenjati ovisno o tome na kojem problemu se želi izvesti eksperiment.

```
Problem problem_br17 = new Problem(@"C:\Users\Ja\source\repos\zavrsni\ALL_atsp\br17.atsp\br17.atsp");
```

Slika 3.10. Stvaranje objekta klase *Problem*

Sljedeća stavka koja se mijenja su tipovi mutacije te je stvaranje ovih objekata prikazano na slici 3.11. Konstruktor kao parametar prima vrijednost vjerojatnosti mutacije te je moguće koristiti dva tipa mutacije.

```
Mutation swapMutation100 = new SwapMutation(1);
Mutation insertMutation10 = new InsertMutation(0.1);
```

Slika 3.11. Stvaranje objekata klase *Mutation*

Nadalje, na slici 3.12 prikazano je stvaranje algoritma pomoću klase *AlgoritmBuilder*. Parametri algoritma vjerojatnost križanja i veličina populacije postavljaju se tijekom gradnje algoritma. Poziv za izvršavanje algoritma prikazuje slikom 3.13 te kao povratnu vrijednost ima objekt klase *Solution*, a kao parametar prima broj ponavljanja koji je naveden kao jedan od parametara algoritma. Objekti križanja i populacije stvaraju se u samom graditelju te je potrebno za parametre postaviti vjerojatnost križanja i veličinu populacije.

```
IAlgorithmBuilder algorithmBuilder = new AlgorithmBuilder();

Algorithm algorithm_swap10 = algorithmBuilder
    .CreateCrossover(1)
    .SetMutation(swapMutation10)
    .CreatePopulation(100)
    .SetProblem(problem_br17)
    .Build();
```

Slika 3.12. Stvaranje i korištenje objekta klase *AlgoritmBuilder*

```
bestSolutions.Add(algorithms[j].Run(1000));
```

Slika 3.13. Poziv za izvršavanje algoritma

4. EKSPERIMENTALNA ANALIZA

Eksperimentalnom analizom cilj je pokazati utjecaj odabira tipa operatora mutacije i veličine vjerojatnosti mutacije na sposobnost algoritma u pronalasku optimalnog rješenja. Eksperiment se provodi na pet različitih problema. Problemi zajedno s optimalnim rješenjima su preuzeti s [12], a njihove karakteristike su prikazane u Tablici 4.1.

Tablica 4.1. Karakteristike razmatranih problema

Naziv problema	**Broj gradova**	**Optimalno rješenje [x_o]**
Br17	17	39
Ftv33	34	1286
Ftv35	36	1473
Ftv38	39	1530
Ftv44	45	1613

4.1. Postavke eksperimenta

Genetski algoritam je stohastičke prirode te je iz tog razloga potrebno algoritam izvršiti više puta, u ovom slučaju se izvršava 30 puta. Kao što je spomenuto u poglavlju 2.2 parametri ovog algoritma su veličina populacije, broj ponavljanja, vjerojatnost križanja i vjerojatnost mutacije. Veličina populacije pri svakoj instanci algoritma postavljena je na 100, broj ponavljanja na 1000 i vjerojatnost križanja na 100%.

Parametar koji se mijenjao te čiji se utjecaj promatrao u analizi je vjerojatnost mutacije i postavljena je na 10%, 30%, 50% i 100%. Ovi parametri su odabrani po preporukama iz [2]. Također se promatrao i utjecaj simboličkog parametra tipa mutacije koje su u ovom slučaju mutacija zamjene i mutacija umetanja. Ovim putem dobije se osam primjerka algoritma koji se primjenjuju na pet navedenih problema.

4.2. Rezultati

Na temelju podataka o najkraćem pronađenom putu prikupljenih tijekom izvođenja programskog rješenja izračunate su vrijednosti aritmetičke sredine ($\bar{x}$), standardne devijacije (σ), minimalne (x_{min}) i maksimalne (x_{max}) optimalne vrijednosti puta, odstupanje aritmetičke sredine od

optimalnog rješenja (Δ)koje se računa prema (4-1) te broj pronalaska optimalnog rješenje za svaku varijaciju algoritma (n).

$$\Delta = \frac{\bar{x} - x_o}{x_o} * 100\% \quad (4\text{-}1)$$

Tablica 4.2. Rezultati za probleme korištenjem operatora mutacije zamjene

Problem		**Vjerojatnost mutacije**			
		10%	30%	50%	100%
Br17	$\bar{x}$	41,3	40,27	39,97	**39**
	σ	2,68	2,12	1,16	**0**
	x_{min}	39	39	39	39
	x_{max}	48	50	44	**39**
	Δ	5,9%	3,25%	2,48%	**0%**
	n	10	14	14	30
Ftv33	$\bar{x}$	1794,13	1737,73	1755,03	**1577**
	σ	168,38	126,39	122,29	**112,19**
	x_{min}	1558	1480	1537	**1420**
	x_{max}	2223	2023	2131	**1880**
	Δ	39,51%	35,13%	36,47%	**22,63%**
	n	0	0	0	0
Ftv35	$\bar{x}$	2031,23	2014,17	1962,27	**1772,97**
	σ	158,71	125,67	107,31	**101,01**
	x_{min}	1724	1732	1784	**1563**
	x_{max}	2315	2211	2235	**1969**
	Δ	37,9%	36,74%	33,22%	**20,36%**
	n	0	0	0	0
Ftv38	$\bar{x}$	2119,53	2077,6	2031	**1806,1**
	σ	141,44	100,57	116,46	**75,82**
	x_{min}	1910	1895	1840	**1683**
	x_{max}	2480	2279	2279	**1983**
	Δ	38,53%	35,79%	32,75%	**18,05%**
	n	0	0	0	0
Ftv44	$\bar{x}$	2329	2356,23	2340,93	**2054,77**
	σ	154,76	177,99	**109,85**	118,97
	x_{min}	2056	1982	2069	**1844**

	x_{max}	2657	2619	2556	**2354**
	Δ	44,39%	46,08%	45,13%	**27,39%**
	n	0	0	0	0

Tablica 4.2 prikazuje rezultate dobivene korištenjem operatora mutacije zamjene na svim prethodno spomenutim problemima. Primjećuje se kako aritmetička sredina najmanje odskače od optimalnog rješenja u slučaju kada je vjerojatnost mutacije postavljena na 100% te se odstupanje povećava smanjenjem vjerojatnosti mutacije. Standardne devijacije se kod skoro svih problem smanjuju povećanjem vjerojatnosti mutacije, iznimka problem Ftv44. Također se može reći i u slučaju minimalnog i maksimalnog pronađenog rješenja, jedino odstupanje je također problem Ftv44. Optimalna rješenja pronađena su samo kod problem Br17 te je broj pojava također raste porastom vjerojatnosti mutacije. Također je bitno za reći da povećanjem broja gradova u problemu raste odstupanje aritmetičke sredine od optimalnog rješenja.

Tablica 4.3. Rezultati za probleme korištenjem operatora mutacije umetanja

Problem		**Vjerojatnost mutacije**			
		10%	30%	50%	100%
Br17	$\bar{x}$	40,1	39,57	39,57	**39**
	σ	1,29	1,19	2,05	**0**
	x_{min}	39	39	39	39
	x_{max}	44	44	50	**39**
	Δ	2,82%	1,45%	1,45%	**0%**
	n	14	23	26	30
Ftv33	$\bar{x}$	1681,43	1620,67	1593,1	**1464,37**
	σ	119,24	133,01	97,59	**72,94**
	x_{min}	1454	1428	1400	**1347**
	x_{max}	1941	1868	1771	**1603**
	Δ	30,75%	26,02%	23,88%	**13,87%**
	n	0	0	0	0
Ftv35	$\bar{x}$	1878,97	1823,63	1811,87	**1646,37**
	σ	137,89	134,09	100,97	**73,02**
	x_{min}	1599	1575	1607	**1513**
	x_{max}	2100	2077	2059	**1816**
	Δ	27,56%	23,8%	23,01%	**11,77%**

	n	0	0	0	0
Ftv38	$\bar{x}$	1966,7	1958,33	1905,33	**1753,57**
	σ	134,03	134,53	127,12	**89,39**
	x_{min}	1707	1718	1663	**1617**
	x_{max}	2213	2213	2143	**1923**
	Δ	28,54%	27,99%	24,56%	**14,61%**
	n	0	0	0	0
Ftv44	$\bar{x}$	2205,87	2200,97	2165,37	**1952,3**
	σ	191,54	152,37	137,76	**122,48**
	x_{min}	1873	1869	1871	**1753**
	x_{max}	2644	2508	2463	**2204**
	Δ	36,76%	36,45%	34,24%	**21,04%**
	n	0	0	0	0

Tablica 4.3 prikazuje rezultate dobivene korištenjem operatora mutacije umetanja na istim prethodno spomenutim problemima. Aritmetička sredina kod svih problema se sve više približava optimalnoj vrijednosti povećanjem vjerojatnosti mutacije. Standardna devijacija se smanjuje na isti način s iznimkom kod problema Br17. Kod minimalnih i maksimalnih rješenja svih problema dolazi do varijacija, odnosno ne smanjuju se povećanjem vjerojatnosti mutacije u svim slučajevima. Odstupanje od aritmetičke sredine također se povećava povećanjem broja gradova, s iznimkom problem Ftv33, i smanjuje se povećanjem vjerojatnosti mutacije. Optimalno rješenje ponovno je pronađeno samo u slučaju problema Br17 te se broj ponavljanja povećava povećanjem vjerojatnosti mutacije.

Usporedbom rezultata u Tablicama 4.2 i 4.3 mogu se usporediti učinkovitosti korištenja različitih operatora mutacije. Aritmetičke sredine kod operatora mutacije umetanja u slučajevima svih problema su manje. Također su manje standardne devijacije, minimalne i maksimalne vrijednosti te odstupanja aritmetičke sredine od optimalnog rješenja. U jedinom slučaju gdje su pronađena optimalna rješenja, problem Br17, češće su pronađena kod operatora mutacije umetanja.

5. ZAKLJUČAK

Asimetrični problem trgovačkog putnika spada pod NP-teške kombinatorne probleme te ovaj problem pronalazi svoje primjene u raznim područjima. Iz ovog razloga bilo je potrebno razviti učinkovite algoritme koji služe u pronalasku rješenja problema, jedan od kojih je genetski algoritam. Genetski algoritam ima svoje specifične parametre koji su bitni za rad i to su: veličina populacije, broj ponavljanja, vjerojatnost mutacije i vjerojatnost križanja. Izborom različitih vrijednosti ovih parametara utječe se na brzinu i kvalitetu pronalaska rješenja za dani problem.

U ovom radu ispitan je utjecaj odabira tipa operatora mutacije te različitih vjerojatnosti mutacije na pronalazak optimalnih rješenja za zadane probleme. Ostali parametri su imali konstantne vrijednosti. Eksperimentalnom analizom pokazalo se da od dva odabrana tipa mutacije, mutacije zamjene i mutacije umetanja, za zadane probleme bolja rješenja se pronalaze korištenjem mutacije umetanja. Također se pokazalo da se povećanjem parametra vjerojatnosti mutacije pronalaze rješenja bliža onom optimalnom. Isto tako se pokazalo da se povećanjem broja gradova u problemu pronađena rješenja sve više odstupaju od optimalnog rješenja.

Budućim nadogradnjama na programsko rješenje moguće je prilagodbom ostalih parametara još više pospješiti pronalazak rješenja bliskih optimalnom. Također je osim toga moguće korištenje drugih operatora križanja koja bi moguće pospješila rad algoritma, zajedno s drugim vjerojatnostima križanja. Ujedno, moguća je primjena drugih poznatih mehanizama selekcije roditelja i preživjelih.

Literatura

[1] J. Cirasella, D. Johnson, L. McGeoch, W. Zhang, “The asymmetric traveling salesman problem: Algorithms, instance generators, and tests,” in Lecture Notes in Computer Science, vol. 2153, Springer, Berlin, Heidelberg, 2001.

[2] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, Springer Publishing Company, Incorporated, Berlin,2015.

[3] G. Gutin, A.Punnen, The Traveling Salesman Problem and Its Variations, Springer, Boston, MA, 2007.

[4] K. Ilavarasi, K. S. Joseph, "Variants of travelling salesman problem: A survey," International Conference on Information Communication and Embedded Systems (ICICES2014),str. 1-7,Indija,2014.

[5] A. P. Engelbrecht, Computational Intelligence: An Introduction, Second Edition. Wiley Publishing, Južna Afrika, 2007.

[6] Y. Nagata, D. Soler, “A new genetic algorithm for the asymmetric traveling salesman problem,” Expert Systems with Applications, vol. 39, no. 10, str. 8947–8953, kolovoz2012.

[7] H. D. Nguyen, Y. Ikuo, Y. Kunihito, Y. Moritoshi, “Greedy genetic algorithms for symmetric and asymmetric tsps,” IPSJ Transactions on Mathematical Modeling and Its Applications, vol. 43, no. 10, str. 165–175,studeni 2002.

[8] G. Martinovic, D. Bajer, “Impact of double operators on the performance of a genetic algorithm for solving the traveling salesman problem,” in Proceedings of the Second International Conference on Swarm, Evolutionary, and Memetic Computing - Volume Part I, (Berlin, Heidelberg), str. 290–298, prosinac2011.

[9] A. Hussain, Y. S. Muhammad, M. N. Sajid, I. Hussain, A. M. Shoukry, S. Gani, “Genetic algorithm for traveling salesman problem with modified cycle crossover operator,” Computational Intelligence and Neuroscience, listopad2017.

[10] J. Kaabi, Y. Harrath, “Permutation rules and genetic algorithm to solve the traveling salesman problem,” Arab Journal of Basic and Applied Sciences, vol. 26, no. 1, str. 283– 291, siječanj2019.

[11] E. Gamma, R. Helm, R. Johnson, J.Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, Sjedinjene Američke Države, 1994.

[12] *TSPLIB*. [Online]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html. [Accessed: 27-Aug-2021].

Sažetak

Asimetrični problem trgovačkog putnika je problem u kojem trgovački putnik mora obići N gradova gdje put od jednog grada do drugog može biti različit od onoga u obrnutom smjeru. Cilj je pronaći što manji put u prihvatljivom vremenskom razdoblju. U ovom radu korišten je genetskog algoritam za pronalazak rješenja problema. U radu je opisan rad genetskog algoritma i pripadajućih dijelova selekcije roditelja, operatora varijacije te selekcije preživjelih. U eksperimentalnom djelu rada ispitan je utjecajoperatora mutacije zamjene i mutacije umetanja, te različitih vjerojatnosti mutacije. Eksperiment je proveden na pet različitih problema te rezultati pokazuju da u ovim slučajevima bolje radi operator mutacije umetanja te se najbolji rezultati postižu pri vjerojatnosti mutacije od 100%.

Ključne riječi: asimetrični problem trgovačkog putnika, genetski algoritam, operator mutacije, vjerojatnost mutacije

Summary

Asymmetrical traveling salesman problem is a problem in which a traveling salesman must visit N cities but the distance between city A to city B can be different from the one in reverse. The goal of the solution is to find the minimal length tour in an acceptable time frame. In this paper, a genetic algorithm was used to find the solution to the problem. The genetic algorithm was also described in the paper as well as parts of it which are parent selection, variation operators and, survivor selection. In the experimental part of the paper mutation operators and mutation probability were tested. The experiment was conducted on five different problems and the results showed that the operator of the insert mutation worked better in these cases and the best results were achieved with mutation probability at 100%.

Key words: asymmetric traveling salesman problem, genetic algorithm, mutation operator, mutation probability

Životopis

Lukrecia Vulić, rođena je u Našicama 4. prosinca 1999.Završila je Osnovnu školu kralj Tomislava 2014. godine. Iste godine upisuje Srednju školu Isidora Kršnjavoga u Našicama te završava u 2018. sa uspješno položenom maturom. Nastavlja školovanje na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na smjeru računarstva. Autorica je trenutno u završetku preddiplomskog studija računarstva.

Prilozi /na CD-u/

1. Završni rad u doc, docx i PDF formatu
2. Projekt programskog rješenja
3. Podaci korišteni za eksperimentalnu analizu