SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU  
ELEKTROTEHNIČKI FAKULTET

Diplomski studij računarstva

*Laboratorijska vježba 2*

*Genetski algoritmi  
Rješavanje problema trgovačkog putnika*

Denis Lazor, DRB

Osijek, 2020.

SADRŽAJ

[**1.** **UVOD** 1](#_Toc35885608)

[**2. PROBLEM TRGOVAČKOG PUTNIKA** 2](#_Toc35885609)

[**2.1.** **Opis rješenja klasičnim algoritmom** 2](#_Toc35885610)

[**2.2.** **Genetski algoritam** 2](#_Toc35885611)

[**2.3.** **Opis rješenja genetskim algoritmom** 3](#_Toc35885612)

[**2.3.1.** **Programski kod za automatsko izvođenje svih kombinacija** 3](#_Toc35885613)

[**3.** **REZULTATI** 7](#_Toc35885614)

[**3.1.** **Ovisnost o veličini populacije** 8](#_Toc35885615)

[**3.2.** **Ovisnost o postotku mutacije** 9](#_Toc35885616)

[**3.3.** **Ovisnost o broju elitnih članova** 10](#_Toc35885617)

[**4.** **ZAKLJUČAK** 12](#_Toc35885618)

# **UVOD**

Na drugoj laboratorijskoj vježbi se koristeći postupak genetskog algoritma rješava problem određivanja najkraćeg puta obilaska zadanih hrvatskih gradova (problem trgovačkog putnika) bez prelaska granice gdje svaki grad treba posjetiti točno jednom(permutacija).

Pri tome, mijenjaju se parametri populacije, mutacije i elitizma te se gleda ovisnost konvergiranja rješenja za promjenu svakog od parametra. Svaki set parametara se mjeri pet puta i odabire se ono rješenje s najboljom fitness vrijednošću (u ovom slučaju minimalnom).

# **2. PROBLEM TRGOVAČKOG PUTNIKA**

Problem trgovačkog predstavlja problem određivanja najkraćeg puta obilaska zadanih gradova. Uz primjenu genetskog algoritma se dalje analizira postupak rješenja problema tako da se na kraju odredi minimalni potrebni put da se obiđu ti gradovi.

## **Opis rješenja klasičnim algoritmom**

Problem je moguće riješiti generiranjem svih mogućih kombinacija obilaska gradova i ukupnu duljinu puta za svaku kombinaciju i izdvojiti onu s najmanjim putem. Ovaj algoritam bi bio vremenski i memorijski zahtjevan za veći broj gradova.

Moguće je problem riješiti tako da se određuju udaljenosti do svih gradova i uzme najmanja i da se prethodni grad izbaci kako ne bi došlo do ponovnog obilaska. I tako dok se ne dođe do konačnog grada. Često ovakav način pretrage neće dovesti do najboljeg rješenja jer puno ovisi o gustoći i rasporedu gradova.

## **Genetski algoritam**

Genetski algoritam heuristička je metoda optimiranja koja imitira prirodni evolucijski proces. Evolucija je robustan proces pretraživanja prostora rješenja. Po načinu djelovanja ubraja se u metode usmjerenog slučajnog pretraživanja prostora rješenja u potrazi za globalnim optimumom.

Genetski algoritam najlakše je opisati njegovim elementima koji su objašnjeni u nastavku.

**Populacija** predstavlja skup jedinki, odnosno rješenja u i-tom koraku algoritma. Svaka jedinka predstavlja kromosom, odnosno jedno moguće rješenje zadanog problema. Svaka jedinka sastoji se od gena koji je nositelj jedne informacije. Geni se mogu kodirati na više načina, a najčešće se koristi binarno (gen može poprimiti samo dvije vrijednosti 0/1), cjelobrojno (gen poprima vrijednosti iz zadanog intervala) te permutacijsko kodiranje (gen poprima cjelobrojne vrijednosti iz zadanog intervala takve da kromosom sadrži sve vrijednosti iz intervala)

**Rekombinacija** predstavlja kombiniranje dva kromosoma (roditelja) u svrhu stvaranja novih i boljih kromosoma (potomka) . Rekombinacija može biti u jednoj točki, u više točaka ili uniformna. Za stvaranje nove populacije se najprije vrši selekcije stare populacije koja može biti generacijska (u rekombinaciji sudjeluju dobre jedinke) ili eliminacijska (u rekombinaciji sudjeluju lošiji kromosomi koje želimo eliminirati). Kako se najbolje jedinke populacije ne bi izgubile, moguće je uvesti mehanizam zaštite najboljih jedinki od bilo kakve izmjene ili eliminacije. Takav mehanizam se naziva **elitizam.**

**Mutacija** predstavlja promjenu nasumično odabranog jednog ili više gena kromosoma s ciljem unošenja nove informacije u populaciju koja će možda omogućiti izlazak iz lokalnog minimuma. U slučaju permutacijskog kodiranja, slučajno se odabire kromosom i zatim se ispremještaju njegovi geni.

Najvažniji element genetskog algoritma jest definiranje sustava vrijednosti koji će za svaku jedinku pokazati koliko je njegovo rješenje dobro. Takav element zove se funkcija dobrote ili **fitness funkcija**. Što je vrijednost fitness funkcije jedinke veća, jedinka će imati veću vjerojatnost preživljavanja i križanja. Dobro definirana fitness funkcija ključna je za proces selekcije.

## **Opis rješenja genetskim algoritmom**

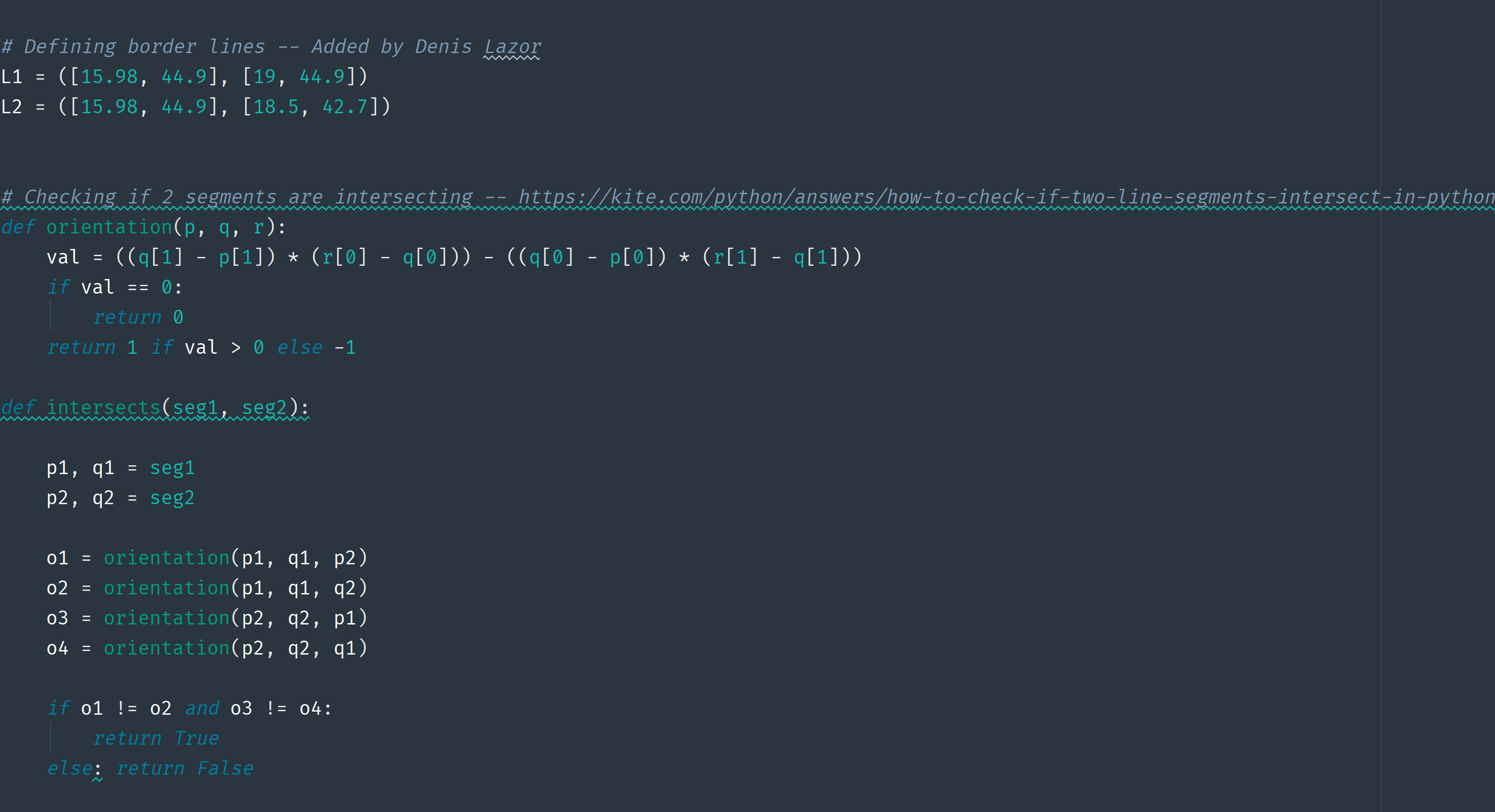
Problem trgovačkog putnika moguće je riješiti genetskim algoritmom definiranim na sljedeći način.

Populaciju čine jedinke koje nose informacije o rasporedu prolaska gradova. Veličina jedinke je uvijek jednaka broju gradova koje treba obići. Gen jedinke ne može sadržavati dva ista grada što je riješeno uvođenjem permutacije.

Fitness funkcija je definirana tako da zbraja udaljenosti između gradova i dodaje 5000 na fitness vrijednost za prelaženje granica (negativna nagrada).

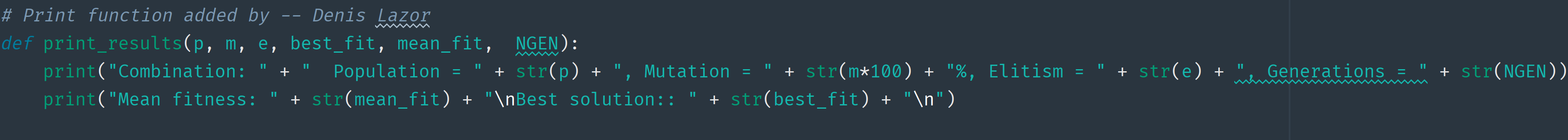
### **Programski kod za automatsko izvođenje svih kombinacija**

U postojeći kod dodane su različite funkcionalnosti koje su detaljno komentirane u kodu i sadrže ''Added by Denis Lazor'' komentare. Prvo su definirane 3 korisničke funkcije. Prva od njih je funkcija za provjeru da li se dva linijska segmenta sijeku. Prima 2 argumenta, to jest dvije definirane linije s koordinatama krajnjih točaka. Prije toga su definirane linije L1 i L2 koje predstavljaju sjevernu i zapadnu granicu BiH i Hrvatske. Funkcija je preuzeta sa internetske stranice prikazane u komentaru koda na slici 1.



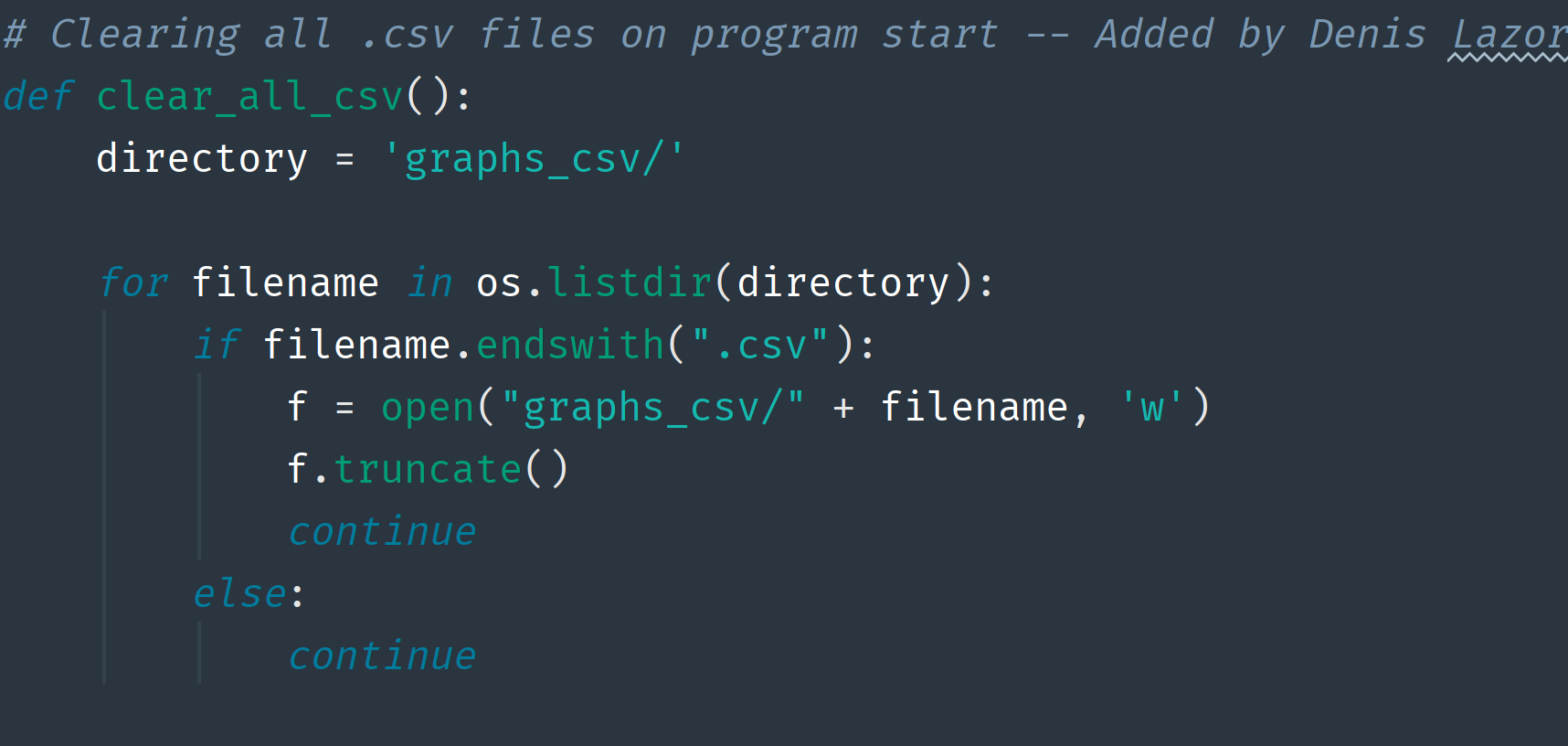
Slika 1 Definiranje funkcije presjeka

Sljedeća funkcija prikazana na slici 2. predstavlja funkciju za prikaz rezultata na ekranu.



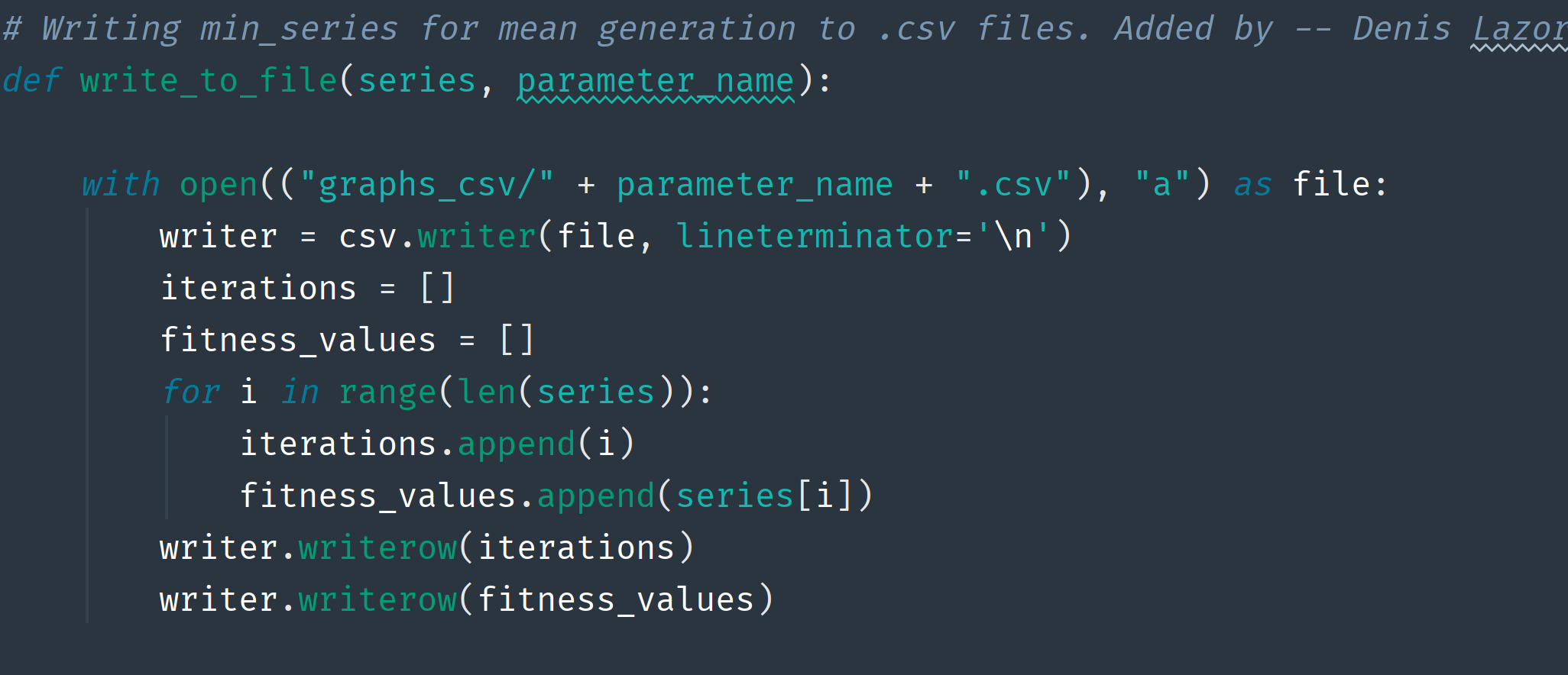
Slika 2 Prikaz rezultata

Funkcija prikazana na slici 3. ima ulogu brisanja sadržaja csv datoteka ukoliko se nakon ponovnog pokretanja programa pokušava upisati nešto u csv datoteku koja ime neki sadržaj.



Slika 3 Funkcija za brisanje sadržaja csv datoteka

Funkcija za pisanje u csv datoteke je prikazana na slici 4.



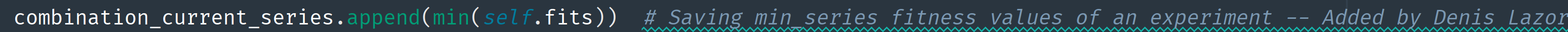
Slika 4 Pisanje u datoteku

Kod na slici 5. se izvodi na pritisak tipke 'start'. Prvo se provjerava da li su csv datoteke prazne i s obzirom na to zove funkcija za brisanje sadržaja. Nakon toga se definira broj generacija koji će biti fiksan. Nakon toga slijedi kod koji automatski prolazi sve potrebne kombinacije 5 puta, ispisuje potrebne informacije na ekran i upisuje liste koje sadrže informaciju o kretanju fitness funkcije u csv datoteke. '…' predstavlja originalni kod koji se ponavlja.



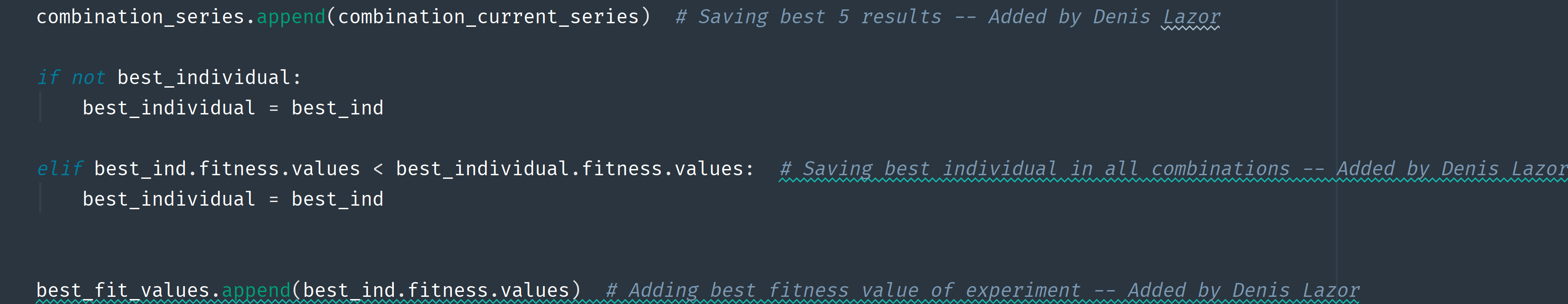
Slika 5 Funkcija za automatsko prolaženje kombinacija

Popunjavanje određenih listi je riješeno u *Evolve()* funkciji. Prvo se upisuje serija minimalnih fitness vrijednosti kombinacije u privremeno polje prikazano na slici 6.

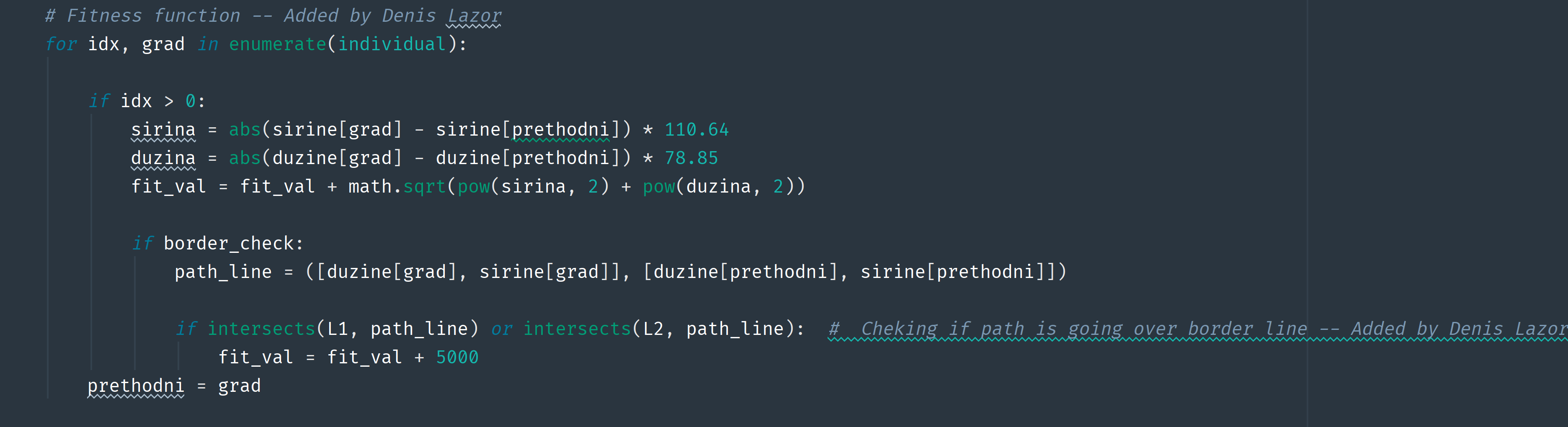


Slika 6 Popunjavanje privremene liste

Nakon toga se privremena lista dodaje na konačnu listu koja sadrži 5 eksperimenata( briše se sadržaj nakon 5 iteracija). Nakon toga se određuje jedinka s najboljom fitness vrijednošću od svih kombinacija te se vrijednosti fitness funkcije za tih 5 eksperimenata zapisuje u polje koje se koristi za nalaženje srednje fitness vrijednosti za svaku kombinaciju i odabir serije kretanja fitness funkcije koja će biti zapisana u csv datoteku za određenu kombinaciju.

 Slika 7 Popunjavanje potrebnih lista

Na slici 8. je prikazana fitness funkcija. Na ukupnu fitness funkciju se nadodaje 5000 ukoliko se linija koja povezuje dva susjedna grada unutar jedinke sječe sa nekom od linija koje predstavljaju granicu.



Slika 8 Fitness funkcija

# **REZULTATI**

Mjerenje je za svaku kombinaciju provedeno 5 puta sa sljedećim parametrima:

• Populacija : 50, 100, 200, 400

• Mutacija: 5%, 10%, 15%, 20%

• Broj elitnih članova: 5, 10, 15, 20

Dok se jedan parametar mijenja ostali su na minimalnim fiksnim vrijednostima.

### **Ovisnost o veličini populacije**

Tablica 1 Ovisnost o veličini populacije

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Broj generacija** | 5000 | | | |
| **Broj elitnih članova** | 5 | | | |
| **% Postotak mutacije** | 5 | | | |
| **Veličina populacije** | 50 | 100 | 200 | 400 |
| **Srednja vrijednost fitness funkcije** | 1796.60524 | 1590.63957 | 1463.39586 | 1348.53754 |
| **Najbolje rješenje** | 1358.86052 | 1344.62835 | 1261.75570 | 1261.75570 |

Slika na kojoj se prikazuje muškarac

Opis je automatski generiran

Slika 3.1.1. Prikaz ovisnosti o veličini populacije

### **Ovisnost o postotku mutacije**

Tablica 2 Ovisnost o postotku mutacije

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Broj generacija** | 5000 | | | |
| **Veličina populacije** | 50 | | | |
| **Broj elitnih članova** | 5 | | | |
| **Postotak mutacije** | 5 | 10 | 15 | 20 |
| **Srednja vrijednost fitness funkcije** | 1796.60524 | 1573.38229 | 1621.51221 | 1423.10129 |
| **Najbolje rješenje** | 1358.86052 | 1429.13533 | 1490.29425 | 1297.33682 |

Slika na kojoj se prikazuje tekst, karta, obično, zrakoplov

Opis je automatski generiran

Slika 3.2.1. Prikaz ovisnosti o postotku mutacije

### **Ovisnost o broju elitnih članova**

Tablica 3 Ovisnost o broju elitnih članova

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Broj generacija** | 5000 | | | |
| **Veličina populacije** | 50 | | | |
| **% Postotak mutacije** | 5 | | | |
| **Broj elitnih članova** | 5 | 10 | 15 | 20 |
| **Srednja vrijednost fitness funkcije** | 1796.60524 | 1561.25685 | 1580.55128 | 1603.97572 |
| **Najbolje rješenje** | 1358.86052 | 1318.05972 | 1307.00849 | 1365.70989 |

Slika na kojoj se prikazuje tekst, karta, čamac, letenje

Opis je automatski generiran

Slika 3.3.1. Prikaz ovisnosti o broju elitnih članova

* 1. **Ukupni rezultati**

Najmanja postignuta fitness funkcija je s jedinkom : [15, 18, 5, 1, 9, 20, 14, 19, 13, 17, 16, 3, 8, 7, 12, 6, 2, 4, 11, 10, 0]

Slika na kojoj se prikazuje tekst, karta

Opis je automatski generiran

*Slika 3.4.1. Rezultat najbolje jedinke prikazan na mapi*

# **ZAKLJUČAK**

Na temelju provedenih mjerenja može se izvesti zaključak o ovisnosti pojedinog parametra na konvergenciju rješenja genetskog algoritma.

Za veličinu populacije može se vidjeti da fitness vrijednost brže konvergira minimumu i daje bolju vrijednost fitness funkcije (u ovom slučaju manju) kada je populacija veća. Povećanjem broja jedinki se povećava mogućnost dobivanja zadovoljavajuće jedinke.

Za postotak mutacije konvergiranje fitness funkcije varira. Dok za najveću vrijednost daje najbolje rezultate za najmanje daje bolje rezultate od srednjih vrijednosti. Konvergencija dosta ovisi o jedinkama koje će mutirati tako da je dosta toga prepušteno sreći. Za velike vrijednosti je veća mogućnost da će mutirati i dobre jedinke i da će konvergencija zapeti. Ukoliko se to ne desi rezultati će biti prilično dobri.

Za broj elitnih članova konvergiranje fitness funkcije varira, ali se može uočiti da je optimalan broj elitnih članova 10-15. Za veliki i mali broj elitnih članova fitness funkcije u početku brže opada, ali zatim zapinje na određenoj vrijednosti dok kod srednjih vrijednosti sporije opada li dolazi do boljeg rješenja. Zato jer prenošenje prevelikog broja jedinki s vremenom smanjuje mogućnost pronalaženja novih dobrih jedinki. Dok kod prenošenja malog broja jedinki se javlja problem gubljenja dobrih jedinki.

Važno je napomenuti da zbog stohasitčke naravi potrebno provesti što više eksperimenata.