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

Diplomski studij

Rješavanje problema trgovačkog putnika koristeći genetski algoritam

Meko računarstvo

Laboratorijska vježba 2

Andrej Bošnjak

DRB

Osijek, 2023.

SADRŽAJ

1. UVOD

Na drugoj laboratorijskoj vježbi se koristeći postupak genetski algoritam rješava problem trgovačkog putnika. Cilj ove vježbe je usporediti dobivene rezultate kada se mijenjaju različiti parametri genetskog algoritma.

Budući da se za svaku konfiguraciju pronalazi pet rješenja, od njih se odabire generacija koja je najbliža srednjoj vrijednosti svih pet generacija (median), za koje se postiglo rješenje problema. Iteracije eksperimenta sa dobivenom median generacijom se potom prikazuju grafički i tablično.

2. PROBLEM TRGOVAČKOG PUTNIKA

Problem trgovačkog putnika opisuje problem obilaska određeni broj točaka točno jednom na način da je ukupni put najkraći.

2.1. Genetski algoritam

Genetski algoritam je heuristička metoda optimiranja koja imitira prirodni evolucijski proces. Evolucija je robustan proces pretraživanja prostora rješenja. Po načinu djelovanja ubrajaju se u metode usmjerenog slučajnog pretraživanja prostora rješenja (*guided random search techniques*) u potrazi za globalnim optimumom.

Populacija je skup jedinki odnosno rješenja u i-tom koraku algoritma. Kromosom je jedna jedinka rješenja odnosno jedno moguće rješenje zadanog problema. Dok gen predstavlja jediničnu informaciju odnosno nositelj je jedne informacije iz rješenja. Geni se mogu kodirati na razne načine koje odgovaraju pojedinim tipovima problema. Najčešći tipovi kodiranja su: binarni, vrijednosni, permutacijski i stablasti:

- Binarni način kodiranja: gen može poprimiti samo dvije vrijednosti: 0 ili 1
- Vrijednosno kodiranje: gen može poprimiti cjelobrojne/realne vrijednosti iz zadanog intervala
- Permutacijsko kodiranje: gen može poprimiti cjelobrojne vrijednosti tako da kromosom uvijek sadrži sve brojeve od 1 do N u različitom redoslijedu
- Stablasto kodiranje: gen je čvor stabla

Genetski algoritmi tijekom svog rada koriste genetske operator za stvaranje novih populacija. Koriste se slijedeći genetski operatori:

Rekombinacija: Kombiniranje gena dva roditelja u svrhu stvaranja novih i boljih potomaka. Najčešće rekombinacije koje se koriste su:

- Rekombinacija u jednoj točki
- Rekombinacija u dvije ili više točaka
- Uniformna rekombinacija

Mutacija: Mutacija mijenja vrijednost nasumično odabranog gena ili više gena i na taj način unosi nove informacije u populaciju i omogućuje izlazak iz lokalnog minimuma. Najčešće se baziraju na vjerojatnosti mutacije jednog gena. Postoji više tipova:

- Jednostavna mutacija
- Potpuna mutacija

Uloga mutacije je i također i u obnavljanju izgubljenog genetskog materijala. Dogodi li se, npr. da sve jedinke populacije imaju isti gen na određenom mjestu u kromosomu, samo križanjem se taj gen nikad ne bi mogao promijeniti.

Genetski algoritam prvo treba odabrati određene dobre roditelje za stvaranje nove populacije. To se vrši metodom selekcije. Svrha selekcije je održavanje i prenošenje dobrih svojstava na slijedeću generaciju. Metodu selekcije dijelimo na:

- Generacijske: Generacijski genetski algoritam u jednoj iteraciji raspolaže s dvije populacije
- Eliminacijske: Za razliku od generacijske selekcije, eliminacijska selekcija ne bira dobre kromosome za slijedeću populaciju, već loše koje treba eliminirati i reprodukcijom ih zamijeniti novima.

U svrhu očuvanja dobrih rješenja (jedinke) nakon puno iteracija algoritma se uvodi i pojam elitizma. Elitizam je mehanizam koji čuva najbolju jedinku od promjena kroz neki od genetskih operatora.

Najvažniji dio genetskog algoritma jest određivanje funkcije dobrote (fitness funkcije) koja će nam govoriti koliko je neko rješenje dobro. Kroz generacije se uz svaki kromosom dodjeljuje i njegova pripadajuća fitness vrijednost, koja se algoritmom pokušava minimizirati ili maksimizirati, ovisno o zadanom problemu i definiciji same fitness funkcije.

2.2. Opis problema i njegovo rješenje

Za rješenje se koristi permutacijsko kodiranje jer problem zahtjeva obilazak svih gradova točno jednom. Kao prvi zadatak nismo uzimali u obzir granice država, a drugi zadatak kažnjava prelaženje granica na način da, ako se put između dva grada sječe sa već definiranim linijama koje predstavljaju granicu, fitness funkciji se dodaje veliki iznos. Kako je zadatak minimizirati fitness funkciju, na ovaj načni kažnjavamo algoritam za prelaženje granice. Fitness funkcija je definirana kao ukupna udaljenost koju je putnik prešao. Za svaku od kombinacija je potrebno ispitati ponašanje genetskog algoritma promjernom parametara:

- Populacija: 50, 100, 200, 400
- Mutacija: 5%, 10%, 15%, 20%
- Broj elitnih članova: 5, 10, 15, 20

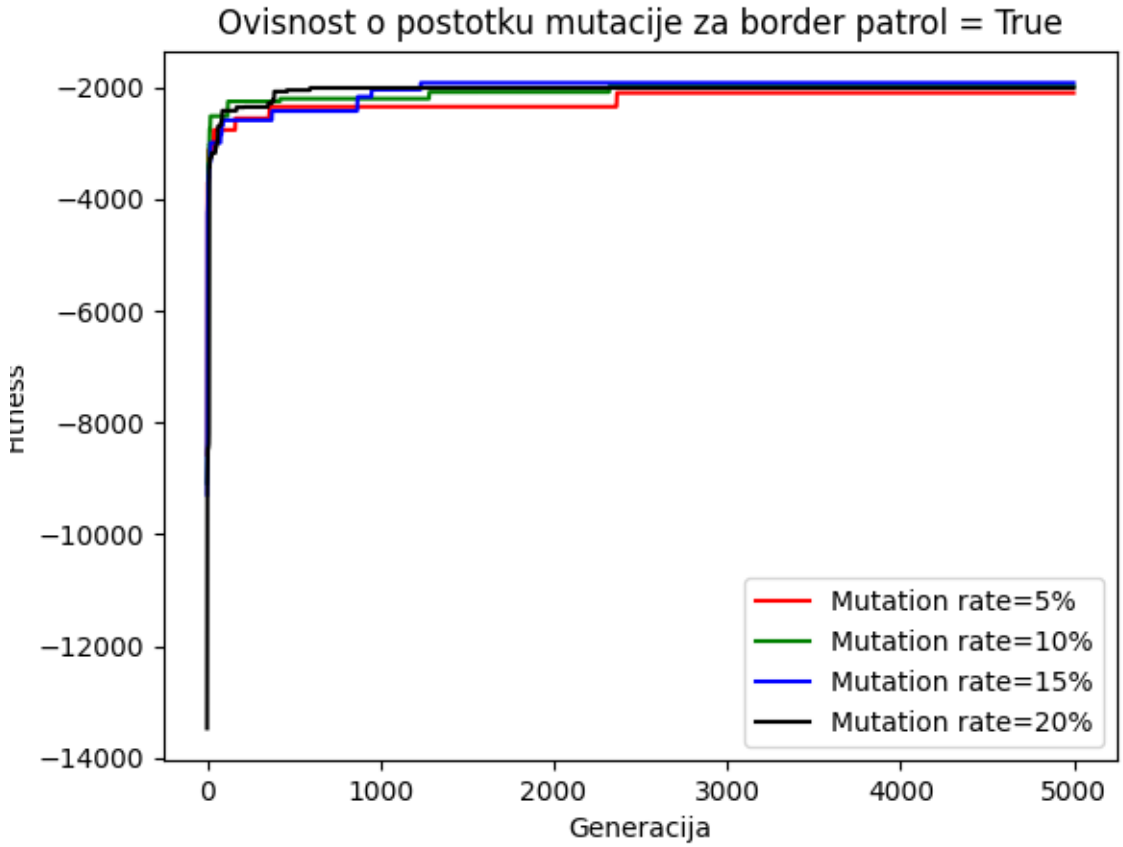
3. GENETSKI ALGORITAM REZULTATI

Svi rezultati su prikazani u idućim podnaslovima.

3.1. Border patrol = True

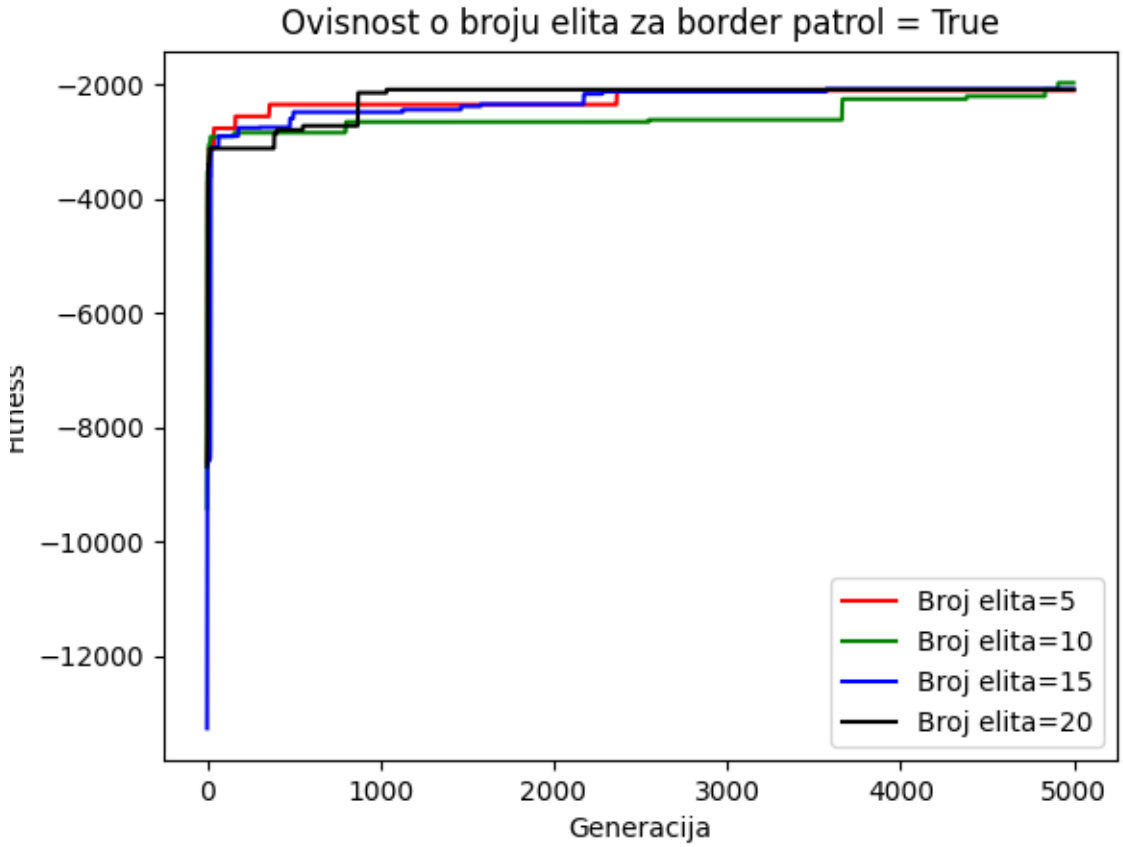
Border patrol	True			
Populacija	50			
Broj elitnih članova	5			
Mutacija	5%	10%	15%	20%
Dobivena rješenja	2097,2227,2145, 2193,2368	2162,1973,2102, 1973,2328	1961,2038,1917, 2125,2203	2007,2166,2020, 2056,2117
Prosječna vrijednost rješenja	2206.5	2108.2	2049.2	2073.7
Najbolje rješenje	[5, 16, 3, 2, 9, 8, 15, 6, 19, 20, 13, 10, 11, 14, 18, 7, 12, 17, 1, 0, 4]	[13, 10, 8, 15, 1, 0, 9, 6, 12, 17, 3, 2, 5, 4, 11, 16, 19, 20, 14, 18, 7]	[14, 9, 2, 5, 10, 8, 18, 7, 12, 19, 20, 13, 11, 16, 3, 4, 17, 1, 0, 15, 6]	[5, 4, 17, 8, 15, 9, 6, 12, 16, 3, 2, 18, 7, 13, 10, 19, 20, 14, 1, 0, 11]

3.1 Ovisnost o postotku mutacije za border patrol = True



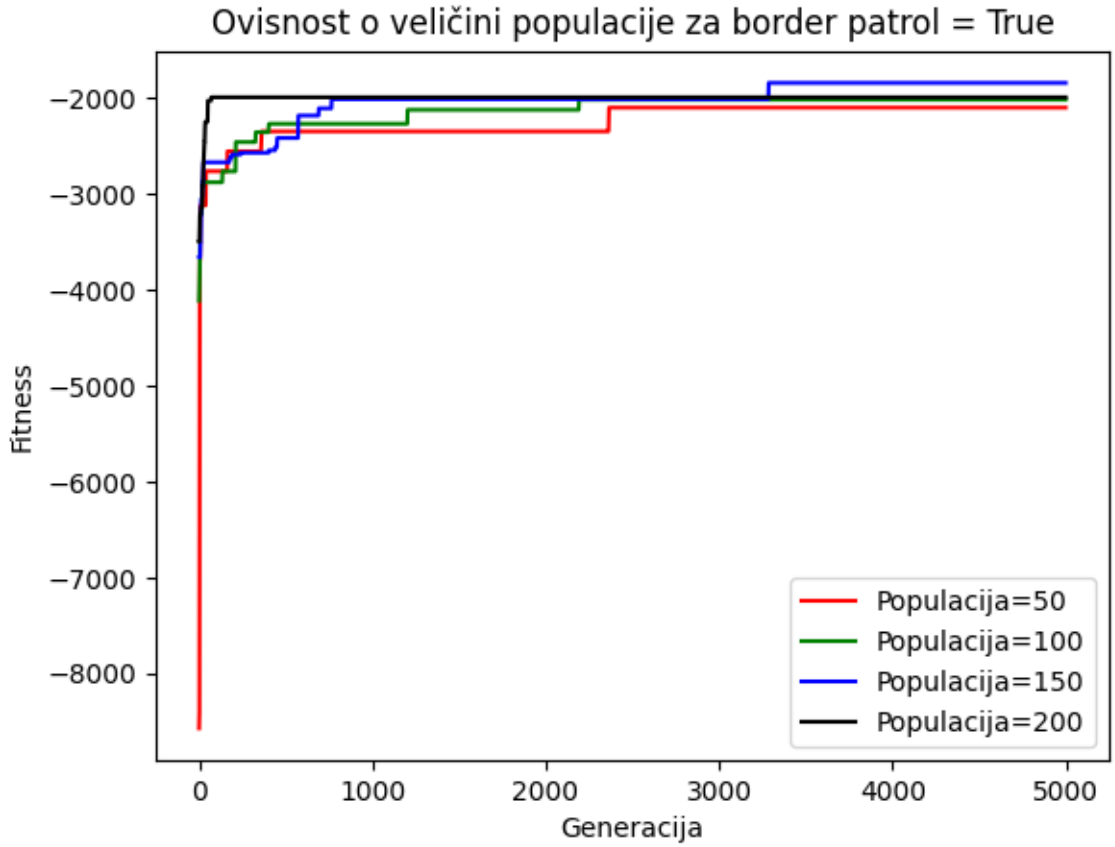
Border patrol	True			
Populacija	50			
Mutacija	5%			
Broj elitnih članova	5	10	15	20
Dobivena rješenja	2097,2227,2145, 2193,2368	1965,2260,2207, 2304,2301	2257,2070,2258, 2258,2026	2083,2158,2315, 2236,2327
Prosječna vrijednost rješenja	2206.5	2208.0	2174.2	2224.1
Najbolje rješenje	[5, 16, 3, 2, 9, 8, 15, 6, 19, 20, 13, 10, 11, 14, 18, 7, 12, 17, 1, 0, 4]	[15, 7, 12, 14, 9, 6, 1, 0, 11, 16, 3, 4, 8, 5, 19, 20, 13, 10, 2, 18, 17]	[13, 20, 14, 18, 4, 17, 15, 6, 7, 12, 19, 16, 3, 9, 2, 1, 0, 10, 8, 5, 11]	[15, 6, 14, 9, 8, 1, 0, 10, 19, 20, 13, 4, 7, 12, 17, 5, 11, 16, 3, 2, 18]

3.2 Ovisnost o broju elitnih članova za border patrol = True



Border patrol	True			
Mutacija	5%			
Broj elitnih članova	5			
Populacija	50	100	200	400
Dobivena rješenja	2097,2227,2145, 2193,2368	2180,1988,2047, 2037,1961	2014,2134,2211, 2055,1860	1965,1887,1839, 2055,1885
Prosječna vrijednost rješenja	2206.5	2043.1	2055.3	1926.5
Najbolje rješenje	[5, 16, 3, 2, 9, 8, 15, 6, 19, 20, 13, 10, 11, 14, 18, 7, 12, 17, 1, 0, 4]	[18, 7, 12, 14, 4, 17, 2, 1, 0, 5, 11, 16, 3, 19, 20, 13, 10, 8, 15, 6, 9]	[1, 0, 5, 10, 8, 15, 6, 20, 13, 4, 17, 7, 12, 19, 14, 9, 11, 16, 3, 2, 18]	[15, 6, 9, 2, 18, 16, 3, 7, 12, 17, 8, 5, 4, 13, 10, 11, 20, 19, 14, 1, 0]

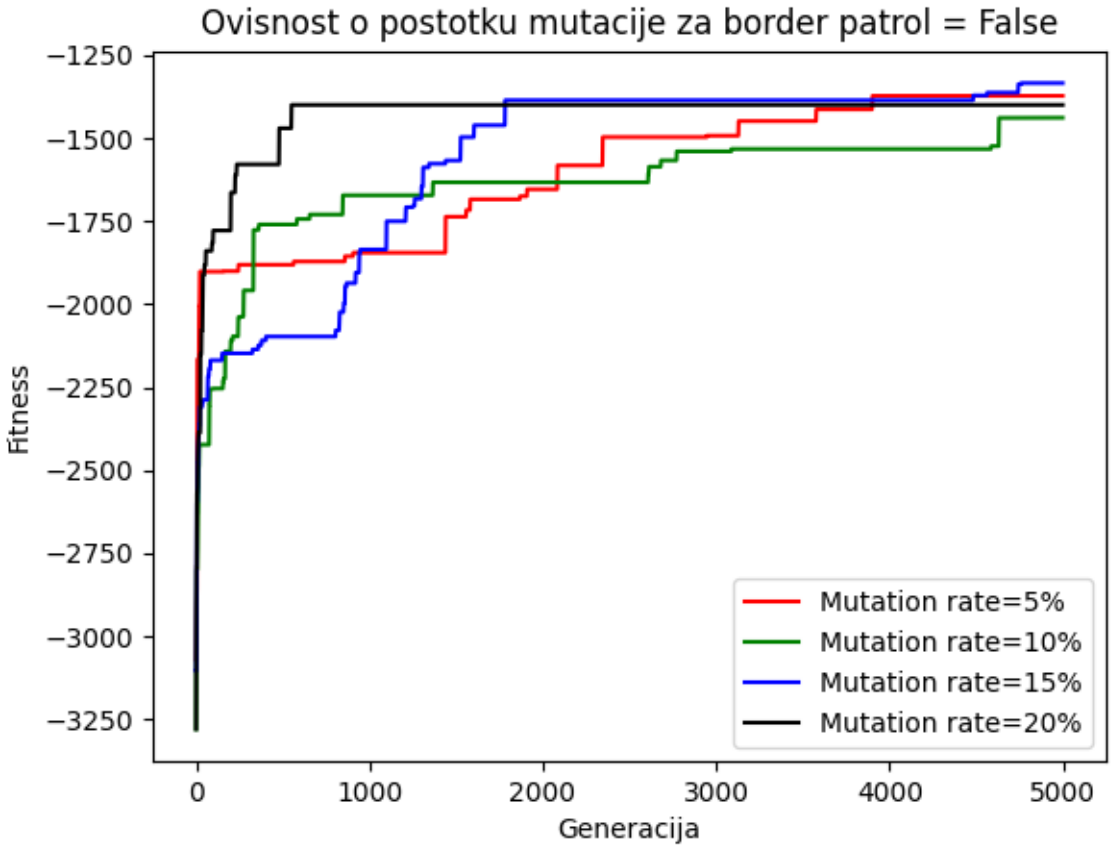
3.3 Ovisnost o veličini populacija za border patrol = True



3.2. Border patrol = False

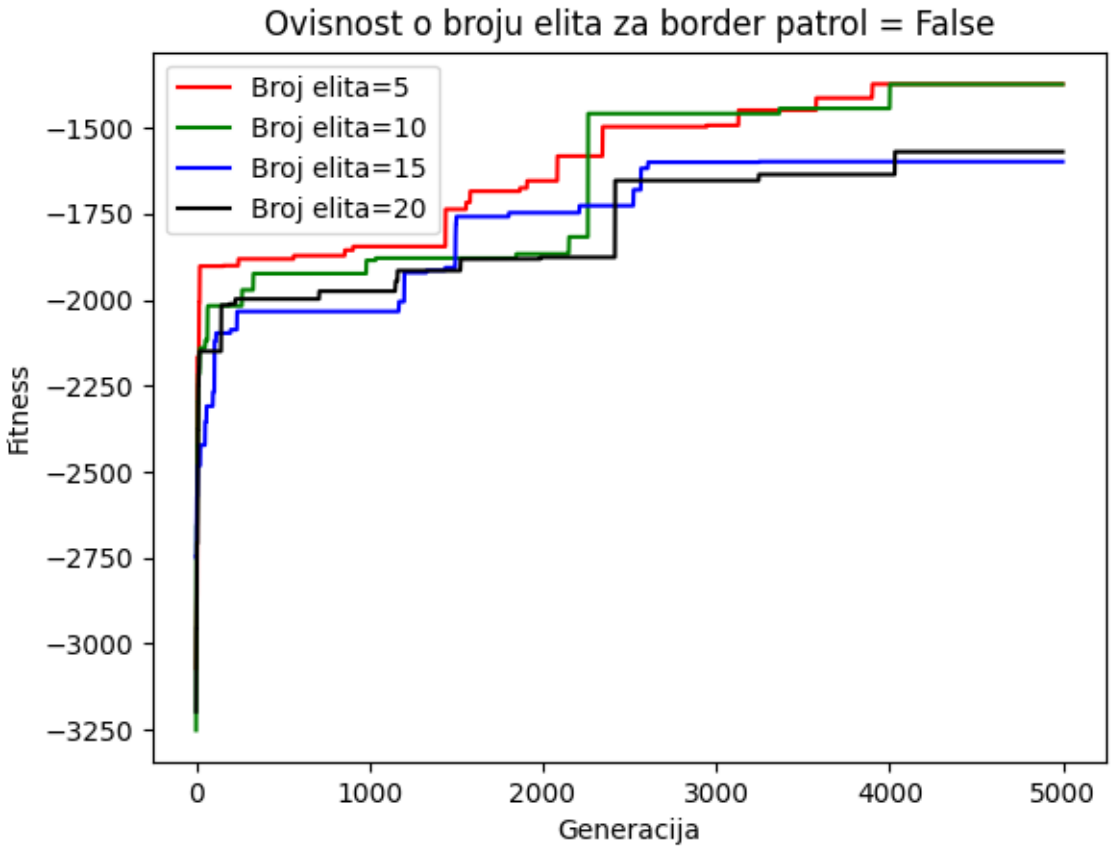
Border patrol	False			
Populacija	50			
Broj elitnih članova	5			
Mutacija	5%	10%	15%	20%
Dobivena rješenja	1372,1666,1658, 1496,1484	1438,1581,1680, 1548,1347	1461,1583,1333, 1581,1524	1741,1690,1604, 1399,1465
Prosječna vrijednost rješenja	1535.7	1519.1	1496.7	1580.2
Najbolje rješenje	[0, 10, 11, 4, 2, 6, 12, 7, 17, 13, 19, 14, 5, 15, 18, 1, 9, 20, 16, 3, 8]	[12, 6, 7, 8, 3, 16, 17, 13, 19, 14, 5, 15, 18, 1, 9, 20, 2, 4, 11, 10, 0]	[12, 6, 7, 16, 20, 9, 1, 5, 15, 18, 14, 19, 13, 17, 3, 8, 2, 4, 11, 10, 0]	[12, 6, 7, 2, 8, 3, 17, 13, 14, 5, 15, 18, 1, 9, 20, 19, 16, 4, 11, 10, 0]

3.4 Ovisnost o postotku mutacije za border patrol = False



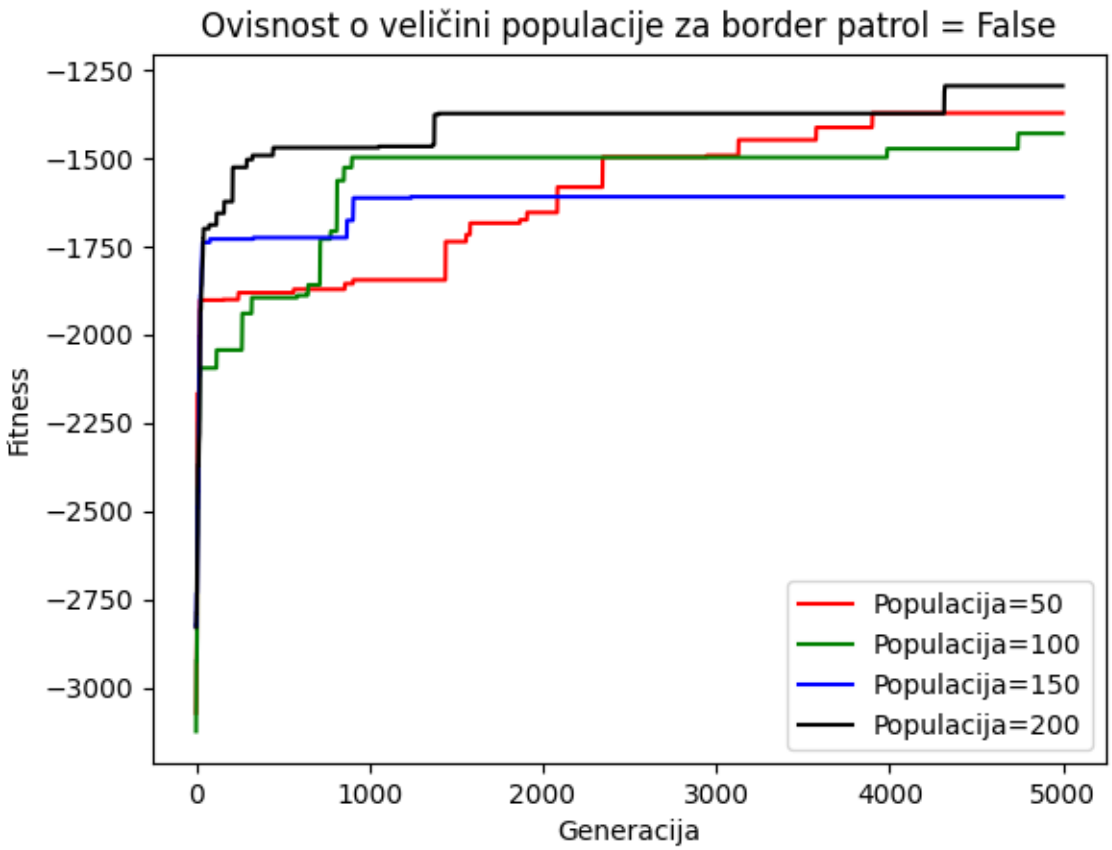
Border patrol	False			
Populacija	50			
Mutacija	5			
Broj elitnih članova	5	10	15	20
Dobivena rješenja	1372,1666,1658, 1496,1484	1714,1557,1372, 1767,1744	1788,1597,1710, 1726,1471	1582,1569,1849, 1740,1721
Prosječna vrijednost rješenja	1535.7	1631.3	1659.0	1692.6
Najbolje rješenje	[0, 10, 11, 4, 2, 6, 12, 7, 17, 13, 19, 14, 5, 15, 18, 1, 9, 20, 16, 3, 8]	[15, 18, 5, 1, 9, 20, 14, 19, 13, 17, 3, 7, 12, 6, 2, 8, 16, 4, 11, 10, 0]	[0, 10, 11, 4, 2, 16, 19, 14, 13, 17, 3, 7, 12, 6, 8, 20, 9, 1, 18, 15, 5]	[12, 6, 7, 17, 13, 19, 20, 14, 5, 15, 18, 1, 9, 4, 2, 8, 3, 16, 11, 10, 0]

3.5 Ovisnost o broju elitnih članova za border patrol = False



Border patrol	False			
Broj elitnih članova	5			
Mutacija	5%			
Populacija	50	100	200	400
Dobivena rješenja	1372,1666,1658, 1496,1484	1698,1612,1812, 1429,1445	1608,1645,1648, 1613,1571	1472,1570,1295, 1485,1526
Prosječna vrijednost rješenja	1535.7	1599.8	1617.6	1470.0
Najbolje rješenje	[0, 10, 11, 4, 2, 6, 12, 7, 17, 13, 19, 14, 5, 15, 18, 1, 9, 20, 16, 3, 8]	[12, 6, 7, 3, 8, 2, 4, 11, 10, 0, 18, 15, 5, 1, 9, 20, 14, 19, 13, 17, 16]	[15, 18, 5, 19, 13, 17, 7, 12, 6, 2, 8, 3, 16, 20, 14, 1, 9, 4, 11, 10, 0]	[0, 10, 11, 4, 2, 8, 6, 12, 7, 3, 16, 17, 13, 19, 14, 20, 9, 1, 5, 18, 15]

3.6 Ovisnost o veličini populacije za border patrol = False


Ovisnost o veličini populacije za border patrol = False
Fitness
−1250
−1500
−1750
−2000
−2250
−2500
−2750
−3000
0
1000
2000
3000
4000
5000
Generacija
Populacija=50
Populacija=100
Populacija=150
Populacija=200

4. Fitness funkcija

Kod s kojim je definirana fitness funkcija dan je u nastavku:

```
for i in range(IND_SIZE):
    distance_row=[]
    for j in range(IND_SIZE):
        sirina=(sirine[i] - sirine[j])*110.64
        duzina=(duzine[i] - duzine[j])*78.85
        distance_row.append(math.sqrt(sirina**2 + duzina**2))
    distance.append(distance_row)

#Define evaluation (fitness) function for individual (cromosome)
def evaluateInd(individual):
    fit_val = 0.0 #starting fitness is 0
    #Implement Your own fitness function!
    for i in range(1,len(individual)):
        fit_val += distance[individual[i]][individual[i-1]]
    return fit_val,#returning must be a tuple becos of posibility of optimization via multiple goal values (objectives)
```

Prvo se definira matrica udaljenosti između pojedinih gradova. Udaljenost se računa prema Pitagorinom poučku $Udaljenost = \sqrt{širina^2 + dužina^2}$, gdje je udaljenost između dva meridijana 78.85 km i između dvije paralele 110.64 km. Na taj način se ispuni tablica udaljenosti gdje i-ti redak i j-ti stupac predstavlja udaljenost između i-tog i j-tog grada.

```
62  ###Border patrol
63
64  s1=45.474865, 13.625031
65  s2=45.488345, 15.347138
66  s3=46.473144, 16.239777
67
68  b1=44.869479, 19.288317
69  b2=45.204253, 15.833898
70  b3=42.637992, 18.322982
71
72  sBorder1 = LineString([s1, s2])
73  sBorder2 = LineString([s2, s3])
74  bBorder1 = LineString([b1, b2])
75  bBorder2 = LineString([b2, b3])
76
77
78  def checkForSlovenia(width1,width2,length1,length2):
79      intersection = False
80      citiesLine = LineString([(width1,length1),(width2,length2)])
81      if(citiesLine.intersects(sBorder1) or citiesLine.intersects(sBorder2)):
82          intersection = True
83      return intersection
84
85
86  def checkForBosnia(width1,width2,length1,length2):
87      intersection = False
88      citiesLine = LineString([(width1,length1),(width2,length2)])
89      if(citiesLine.intersects(bBorder1) or citiesLine.intersects(bBorder2)):
90          intersection = True
91      return intersection
92
93  for i in range(IND_SIZE):
94      distance_row=[]
95      for j in range(IND_SIZE):
96          sirina=(sirine[i] - sirine[j])*110.64
97          duzina=(duzine[i] - duzine[j])*78.85
98          distance_row.append(math.sqrt(sirina**2 + duzina**2))
99          if(checkForSlovenia(sirine[i],sirine[j],duzine[i],duzine[j])):
100             distance_row.append(5000)
101         if(checkForBosnia(sirine[i],sirine[j],duzine[i],duzine[j])):
102             distance_row.append(5000)
103     distance.append(distance_row)
```

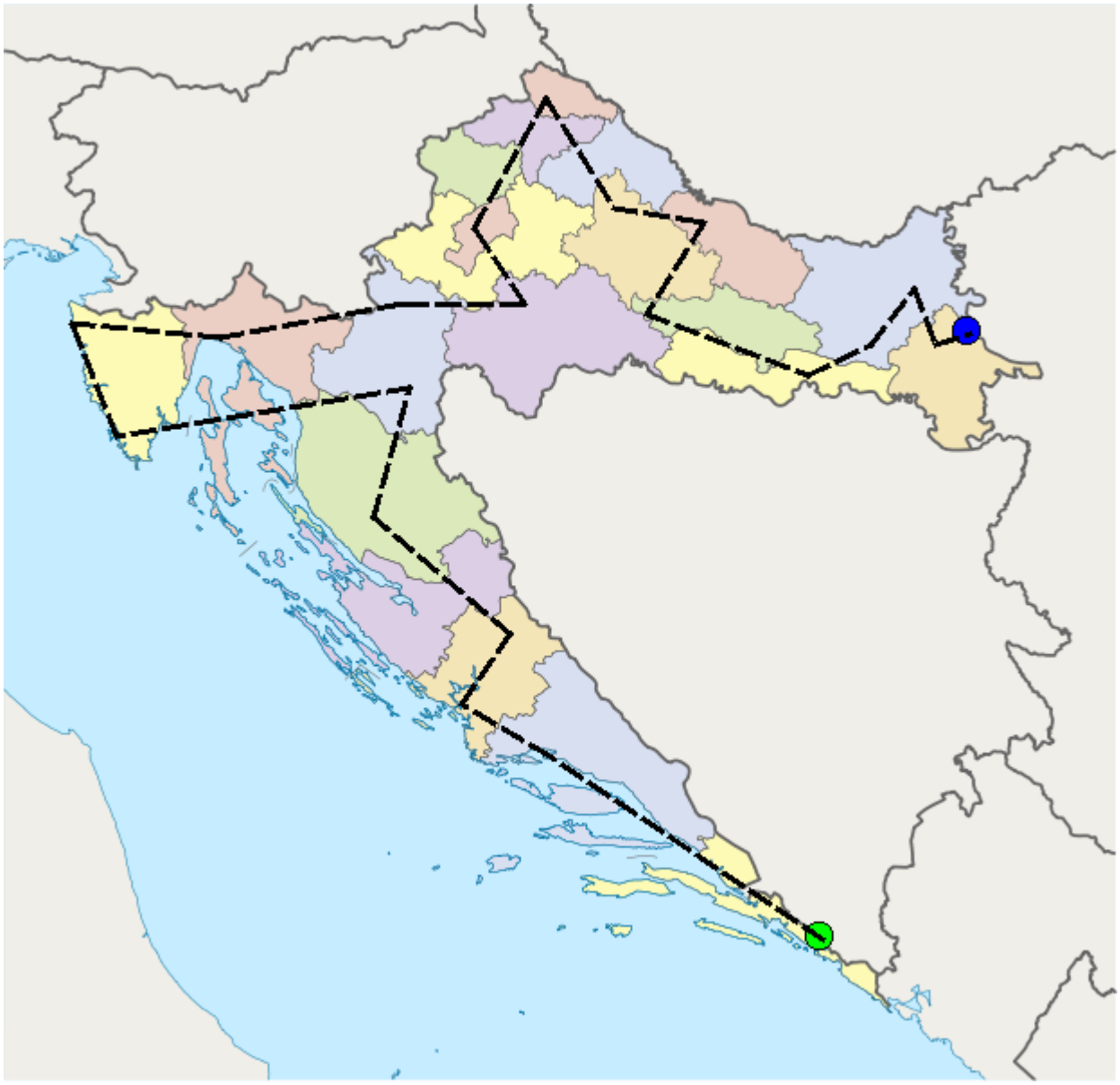
Border patrol je definiran na način da se uzmu 3 točke u blizini granica sa Slovenijom i 3 točke u blizini granice sa Bosnom i Hercegovinom te povuku zamišljene linije između tih točaka. Ako bi rješenje sjeklo neku od tih linija, fitness funkciji se dodaje vrijednost od 5000 kako bi se kaznilo to rješenje.

4.1. Najbolje rješenje

Najbolje rješenje je postignuto za kombinaciju:

- Border patrol = False
- Populacija = 400
- Mutacija = 5%
- Broj elitnih članova = 5

Iznos fitness funkcije je 1295



5. Zaključak

Nakon izvršenih kombinacija parametara za navedeni algoritam računanja problema trgovačkog putnika možemo zaključiti da promjenom bilo kojeg parametra direktno utječemo na rezultate rješenja. Ta ovisnost se najbolje može vidjeti na prikazanim grafovima.

Najbolja rješenja su dobivena povećanjem veličine populacije, gdje u oba slučaja (sa i bez granice) što je veća populacija to, u prosjeku, ukupni put bude manji.

U oba slučaja (sa i bez granice) uočavamo da povećanjem postotka mutacije se dobivaju bolji rezultati, no preveliki postotak (20%) daje lošije rezultate.

Povećanjem broja elitnih članova u slučaju sa granicom rezultati se ne mijenjaju previše, no povećanjem istog parametra za slučaj bez granice se dobivaju lošiji rezultati.