SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU ELEKTROTEHNIČKI FAKULTET

Diplomski studij računarstva

Laboratorijska vježba 1 - Genetski algoritam

(rješavanja problema n-dama)

Ivan Budoš, DRB

Osijek, 2022.

SADRŽAJ

UVOD

Na prvoj se laboratorijskoj vježbi koristeći genetski algoritam rješava problem n-dama na šahovskoj ploči proizvoljnih dimenzija tako da niti jedna dama ne može napasti neku od ostalih dama.

Zadatak vježbe je da se na šahovskoj ploči promjenjivih dimenzija (12x12, 24x24, 48x48) mijenjaju parametri populacije, mutacije i elitizma te se promatra kakav utjecaj promjena parametara ima na brzinu konvergiranja prema rješenju. Zbog stohastičke naravi genetskog algoritma svaki eksperiment potrebno je ponoviti najmanje 5 puta (5 točnih rješenja) i zabilježiti srednji (medijan) čiju je promjenu fitnes vrijednosti potrebno prikazati i na grafu.

OPIS PROBLEMA N-DAMA

N-dame (*eng. N-queen*) je problem postavljanja dama na šahovsku ploču dimenzija NxN tako da niti jedna dama ne može napasti drugu damu.

Opis rješenja

Do rješenja ovog problema se može doći na više načina. Jedan od mogućih načina je rekurzija. Dame se postavljaju na početni redak ili stupac te se traži stupac ili redak u koji je moguće postaviti damu, a da ona nije u mogućnosti napasti drugu damu niti je u mogućnosti da bude napadnuta. Ukoliko takav stupac ili redak ne postoji, rekurzivno se vraća u prošli korak i pokušava se sa nekom drugom pozicijom dame.

Drugi način je generiranje svih mogućih rješenja danog problema i uzimanje samo onih točnih. Takav način rješavanja je dobar za manje dimenzije ploče, ali kod većih dimenzija postaje jako memorijski i vremenski zahtijevan.

Genetski algoritam

Genetski algoritam je metaheuristička 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 rada 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.

Binarno kodiranje – 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, ali različiti raspored.
Stablasto kodiranje – gen je čvor stabla.

Tijekom rada genetski algoritam koristi genetske operatore za stvaranje novih populacija odnosno novih rješenja. Koriste se sljedeći genetski operatori:

REKOMBINACIJA (križanje)- Kombiniranje gena dva roditelja u svrhu stvaranja novih i boljih potomaka. Najčešća rekombinacija može biti:

Rekombinacija u jednoj točki- Slučajnim odabirom točke rekombinacije kod roditelja vrši se zamjena gena od te točke na dalje.

Rekombinacija u dvije (više) točaka- Točke rekombinacije se biraju slučajno i nakon njih se naizmjence vrši zamjena gena.

Uniformna rekombinacija- Slučajno se odabire svaki gen da li će biti odabran iz jednog roditelja ili drugog. Odnosno svaki gen ima vjerojatnost odabira od jednog roditelja ili drugog.

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- slučajno se odabire gen unutar kromosona i mijenja ga se

Potpuna mutacija- slučajno se bira kromosom i zatim se ispremješta gene u njemu.

Ovisno o kojem kodiranju gena se radi, mutacija drugačije radi. Kod binarno kodiranih invertira bitove, kod vrijednosnih kodiranja mijenja vrijednost gena za određeni interval, kod permutacijskog kodiranja odabire dva gena i mijenja im mjesta, dok kod stablastih mijenja mjesto na stablu.

Da bi genetski algoritam provodio ove prethodno navedene genetske operatore prvo mora odabrati određene „dobre“ roditelje za stvaranje nove populacije. Odabir roditelja se vrši pomoću metoda selekcije. Stoga je svrha selekcije čuvanje i prenošenje dobrih svojstava na slijedeću generaciju jedinki. Genetske algoritme s obzirom na vrstu selekcije dijelimo na :

Generacijske – Generacijski genetski algoritam u jednoj iteraciji raspolaže sa dvije populacije (što je ujedno i nedostatak generacijskog genetskog algoritma), jer se odabiru dobre jedinke iz stare populacije koje cine novu populaciju i nakon selekcije sudjeluju u procesu reprodukcije. Vrijeme preživljavanja jedinki je točno jedna generacija.

Eliminacijske – za razliku od generacijske selekcije, eliminacijske selekcije ne bira dobre kromosone za sljedeću populaciju, već loše koje treba eliminirati i reprodukcijom ih zamijeniti sa novima. Dakle, loši kromosoni umiru, a njih nadomještaju djeca nastala reprodukcijom roditelja, tj. preživjelih kromosona. Nema stroge granice među generacijama, vrijeme preživljavanja jedinke ovisi o njenoj kvaliteti.

Koristeći bilo koje od ove dvije metode selekcije postoji opasnost da se dobro rješenje dobiveno nakon puno iteracija izgubi ukoliko ga genetski operatori izmjene. Stoga se javlja potreba za mehanizmom zaštite najbolje jedinke od bilo kakve izmjene ili eliminacije tijekom evolucijskog procesa. Takav mehanizam se naziva elitizam. Genetski algoritam s ugrađenim elitizmom iz generacije u generaciju, asimptotski teži ka globalnom optimumu, odnosno rješenju problema.

Funkcija dobrote ili funkcija ocjene kvalitete jedinke se u literaturi još naziva fitness funkcija, funkcija sposobnosti, funkcija cilja ili evaluacijska funkcija i u najjednostavnijoj interpretaciji ekvivalent je funkciji f koju treba optimizirati, odnosno ona predstavlja naš problem koji pokušavamo riješiti. Što je dobrota jedinke veća, jedinka ima veću vjerojatnost preživljavanja i križanja. Funkcija dobrote je ključ za proces selekcije.

Rješenje problema n-dama primjenom genetskog algoritma

Problem n-dama moguće je riješiti primjenom genetskog algoritma. Kod genetskog algoritma populaciju čine jedinke koje nose informacije o položaju dama na šahovskoj ploči. Dimenzija šahovske ploče odgovara veličini jedinke. Jedinka će se sastojati od N gena ukoliko je

dimenzija ploče NxN. Svaki element na i-tom mjestu predstavlja poziciju dame u i-tom retku, a vrijednost elementa predstavlja stupac u kojem se nalazi dama. Fitnes funkcija je definirana tako da broji koliko se dama međusobno napada.

Implementacija koda za automatsku izmjenu parametara

Uz već priloženi kod za rješavanje problema n-dama genetskim algoritmom bilo je potrebno omogućiti automatsko izvođenje programa na način da se parametri sami izmjenjuju te se izvršava algoritam 5 puta. To je implementirano unutar nove funkcije *Automatic_btn()* koja prvo kreira polja population, mutations i elites sa mogućim vrijednostima parametara. Funkcija se pokreće preko gumba *Automatic* koji je također dodan.

```
def Automatic_btn(self):

    global NO_QUEENS
    global NGEN
    population = [50, 100, 200]
    mutations = [0.04, 0.08, 0.16]
    elites = [4, 8, 16]
```

Slika 1. Predefinirani parametri unutar funkcije „*Automatic_btn*“

Automatska izmjena parametara omogućena je pomoću for petlje. Ona to radi na način da mijenja vrijednost jednog parametra dok ostali ostaju isti. While petljom je omogućeno pronalaženje 5 rješenja za svaku kombinaciju parametara. Ukoliko se pronađe rješenje prije nego algoritam završi s izvođenjem, ono se sprema u csv datoteku koja također sadrži broj generacija te vrijednost fitness funkcije najbolje jedinke u i-toj generaciji.

```
for i in range(3):
    print("\n" + str(i+1) + ". Queens on board:", NO_QUEENS)
    for j in range(3):
        print("        " + str(i+1) + "." + str(j+1), end="  ")
        k = 0
        while k < 5:
            k += 1
            found = self.findSolution(population[j], mutations[0], elites[0], k)
            if not found:
                k -= 1

    for j in range(1, 3):
        print("        " + str(i+1) + "." + str(j+3), end="  ")
        k = 0
        while k < 5:
            k += 1
            found = self.findSolution(population[0], mutations[j], elites[0], k)
            if not found:
                k -= 1

    for j in range(1, 3):
        print("        " + str(i+1) + "." + str(j+5), end="  ")
        k = 0
        while k < 5:
            k += 1
            found = self.findSolution(population[0], mutations[0], elites[j], k)
            if not found:
                k -= 1

    NO_QUEENS = NO_QUEENS * 2
    NGEN = NGEN * 2
```

Slika 2. Implementacija automatizma

REZULTATI

Potrebno je na šahovskoj ploči promjenjivih dimenzija (12x12,24x24) provesti mjerenja uz cjelobrojno i permutacijsko kodiranje. Izmjeren je utjecaj promjene parametara za svaku dimenziju ploče te je za svaki set parametara pronađeno 5 rješenja od kojih je zadržano ono rješenje koje je najbliže srednjoj vrijednosti po broju generacija potrebnih za pronalazak rješenja.

Parametri genetskog algoritma su poprimali sljedeće vrijednosti:

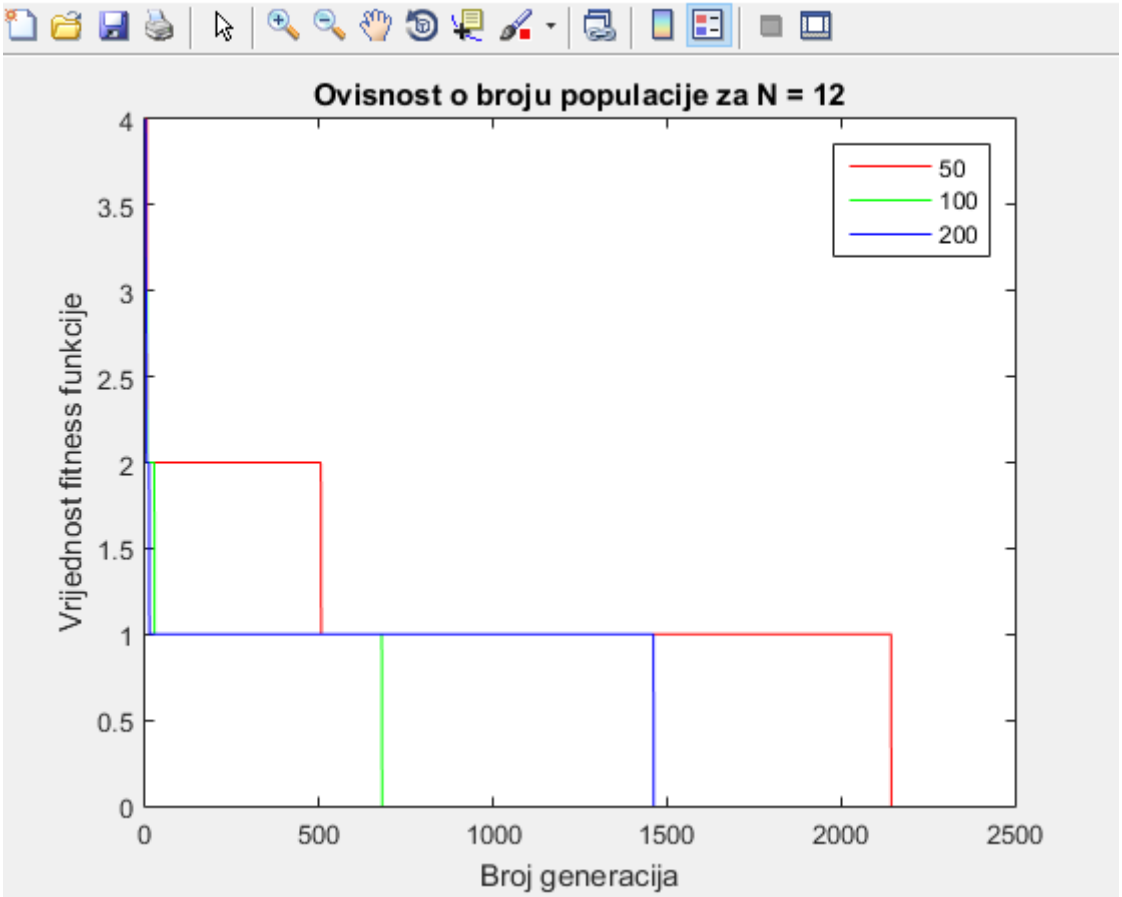
- Veličina populacije: 50,100,200
- Postotak mutacije: 4%, 8%, 16%
- Broj elitnih članova: 4, 8, 16

Cjelobrojno kodiranje

OVISNOST O BROJU POPULACIJE

Kodiranje	cjelobrojno		
Broj dama	12		
Broj generacija	10 000		
Broj elitnih članova	4		
Postotak mutacije[%]	4		
Veličina populacije	50	100	200
Broj generacija da bi se dobilo rješenje	4390, 325, 2146, 345, 504	144, 211, 682, 10, 5032	4547, 14, 5998, 1463, 37
Medijan broja generacija da bi se dobilo rješenje	2146	682	1463

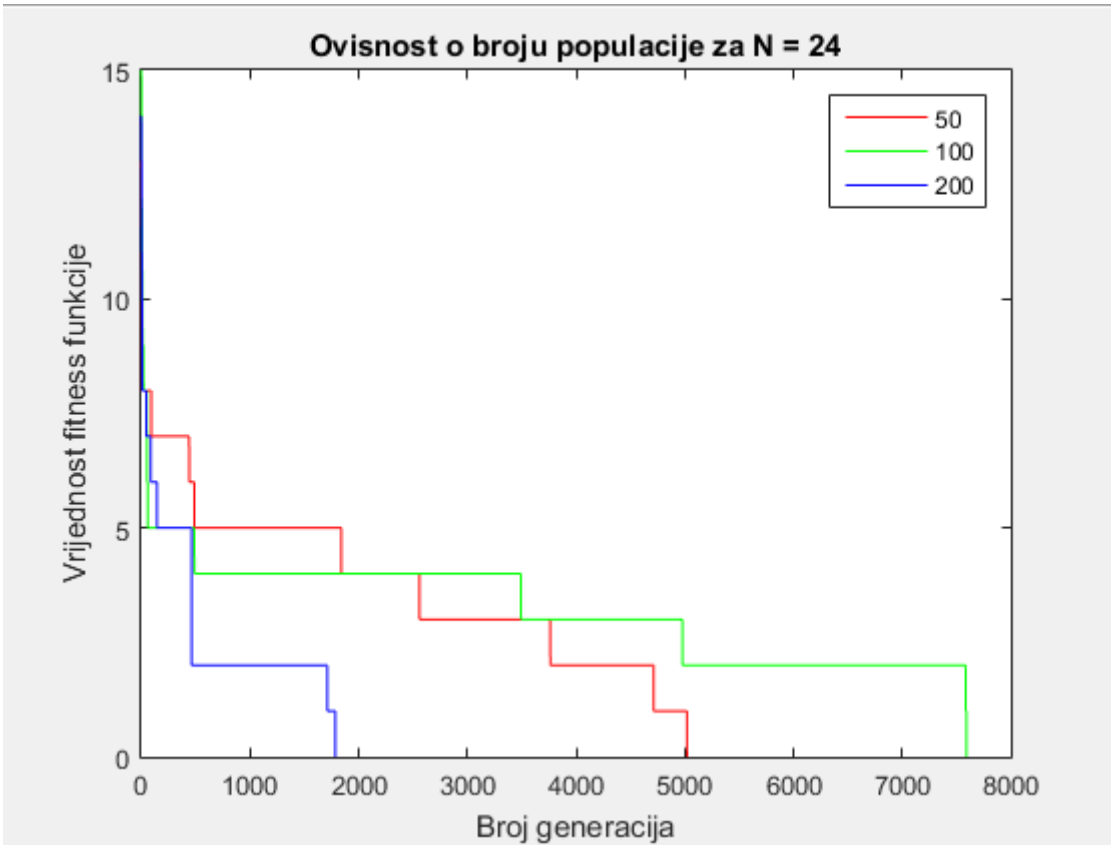
Tablica 1. Ovisnost o broju populacije za N = 12



Slika 4. Prikaz ovisnosti o veličini populacije za N = 12

Kodiranje	cjelobrojno		
Broj dama	24		
Broj generacija	20 000		
Broj elitnih članova	4		
Postotak mutacije[%]	4		
Veličina populacije	50	100	200
Broj generacija da bi se dobilo rješenje	5731, 4292, 5023, 8924, 1145	6399,9732,7593,9231,5010	1052,1787,3423,2003,703
Medijan broja generacija da bi se dobilo rješenje	5023	7593	1787

Tablica 2. Ovisnost o broju populacije za N = 24

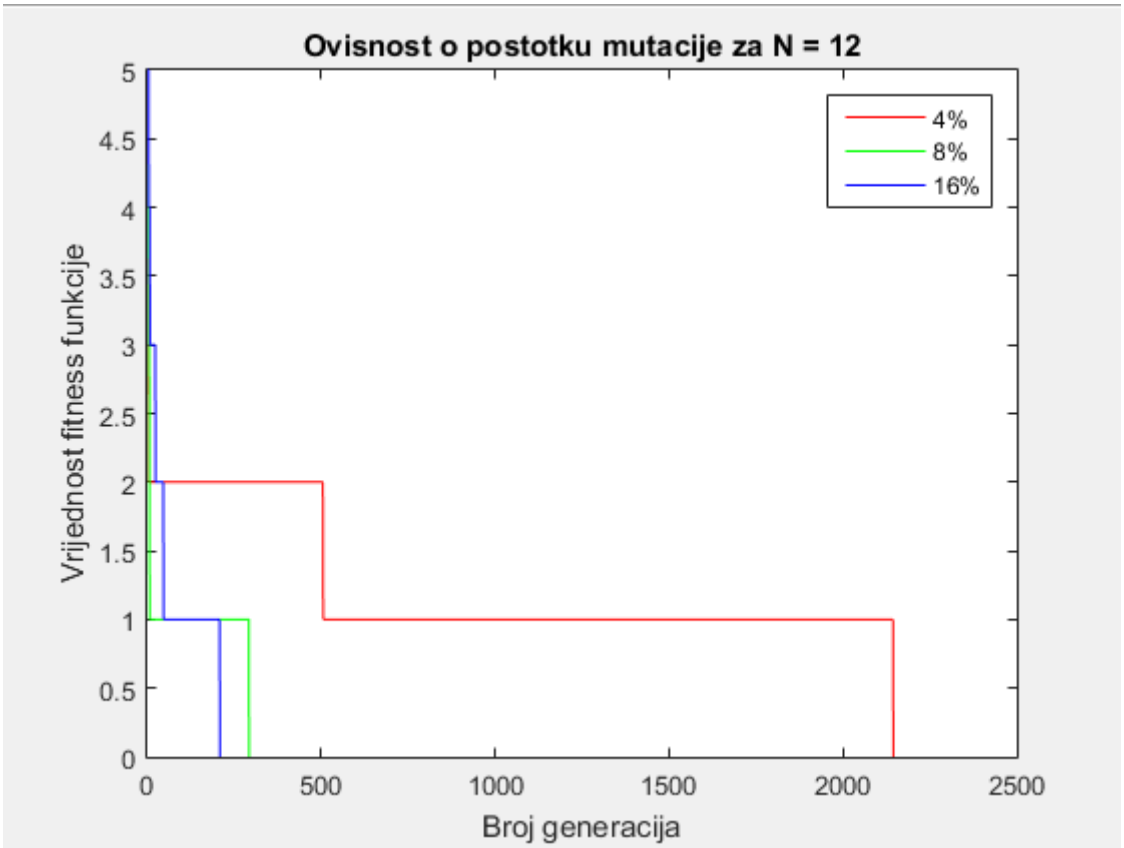


Slika 5. Prikaz ovisnosti o veličini populacije za N = 24

OVISNOST O POSTOTKU MUTACIJE

Kodiranje	cjelobrojno		
Broj dama	12		
Broj generacija	10 000		
Veličina populacije	50		
Broj elitnih članova	4		
Postotak mutacije[%]	4	8	16
Broj generacija da bi se dobilo rješenje	4390, 325, 2146, 345, 504	601, 294, 68, 177, 330	221,309,210,132,178
Medijan broja generacija da bi se dobilo rješenje	2146	294	210

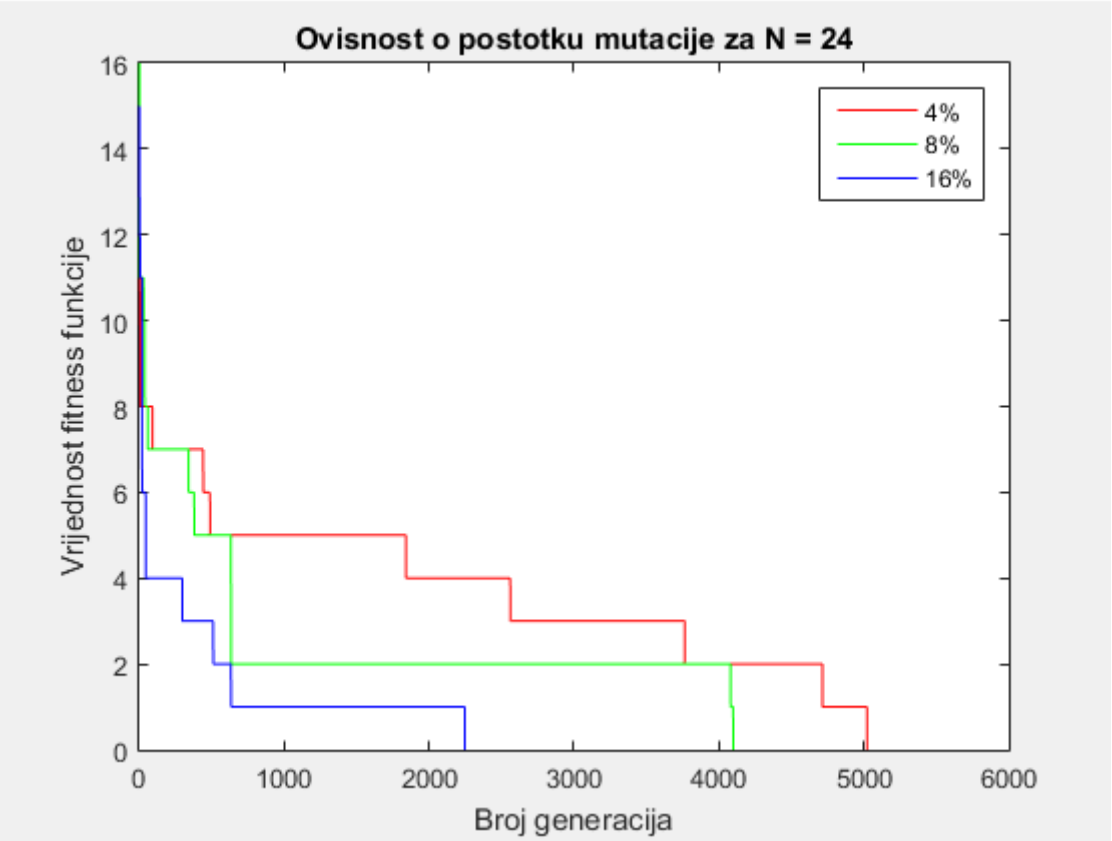
Tablica 3. Ovisnost o postotku mutacije za N = 12



Slika 6. Prikaz ovisnosti o postotku mutacije za N = 12

Kodiranje	cjelobrojno		
Broj dama	24		
Broj generacija	20 000		
Veličina populacije	50		
Broj elitnih članova	4		
Postotak mutacije[%]	4	8	16
Broj generacija da bi se dobilo rješenje	5731, 4292, 5023, 8924, 1145	2749,4672,4100,6899,2080	1425,2248,3623,1632,2312
Medijan broja generacija da bi se dobilo rješenje	5023	4100	2248

Tablica 4. Ovisnost o postotku mutacije za N = 24

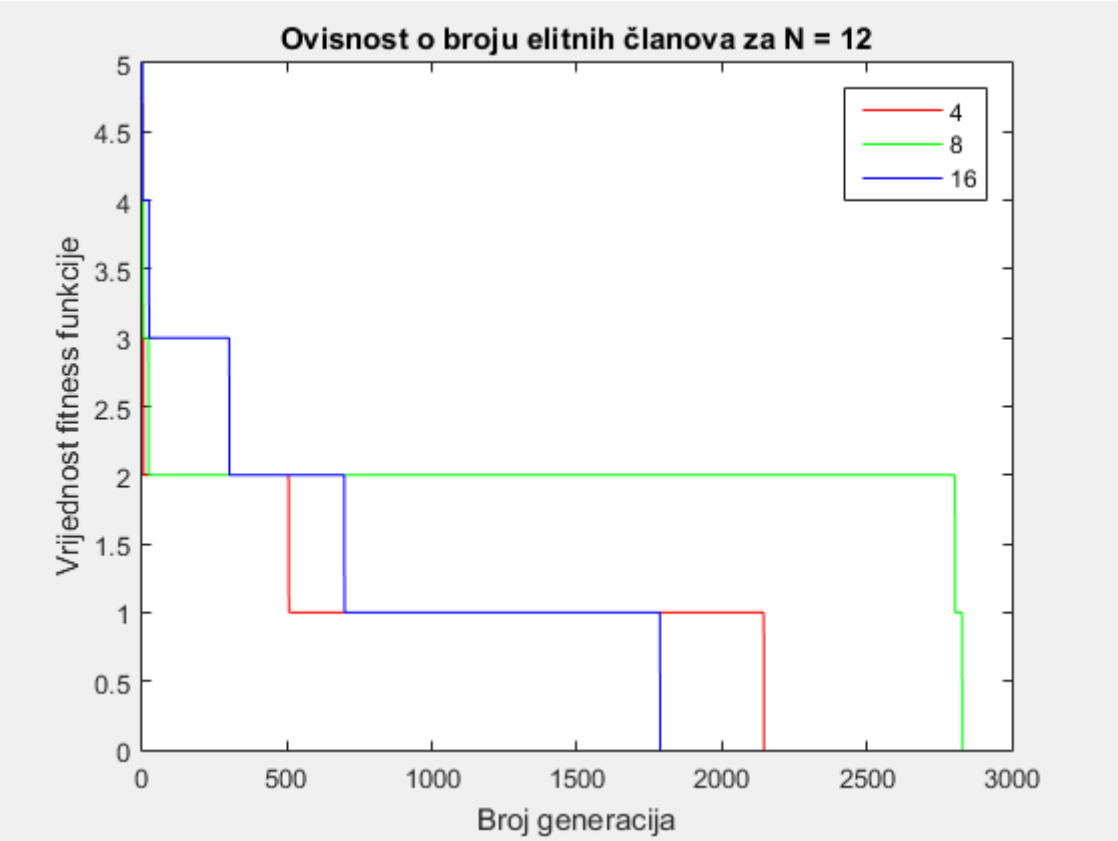


Slika 7. Prikaz ovisnosti o postotku mutacije za N = 24

OVISNOST O BROJU ELITNIH ČLANOVA

Kodiranje	cjelobrojno		
Broj dama	12		
Broj generacija	10 000		
Veličina populacije	50		
Postotak mutacije[%]	4		
Broj elitnih članova	4	8	16
Broj generacija da bi se dobilo rješenje	4390, 325, 2146, 345, 504	2830,3427,4731,1742,1420	221,309,210,132,178
Medijan broja generacija da bi se dobilo rješenje	2146	2830	1788

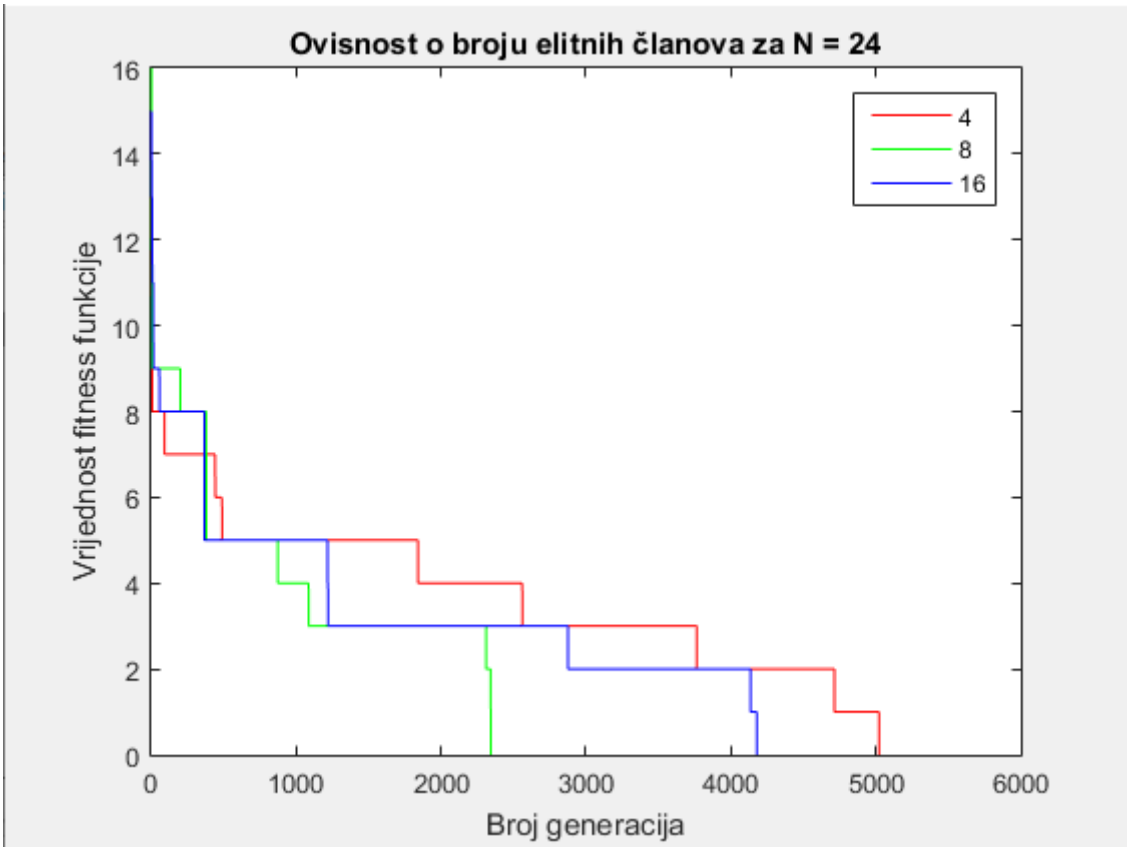
Tablica 5. Ovisnost o broju elitnih članova za N = 12



Slika 7. Prikaz ovisnosti o broju elitnih članova za N = 12

Kodiranje	cjelobrojno		
Broj dama	24		
Broj generacija	20 000		
Veličina populacije	50		
Postotak mutacije[%]	4		
Broj elitnih članova	4	8	16
Broj generacija da bi se dobilo rješenje	5731, 4292, 5023, 8924, 1145	2830,3427,4731,1742,1420	221,309,210,132,178
Medijan broja generacija da bi se dobilo rješenje	5023	2434	4180

Tablica 6. Ovisnost o broju elitnih članova za N = 24



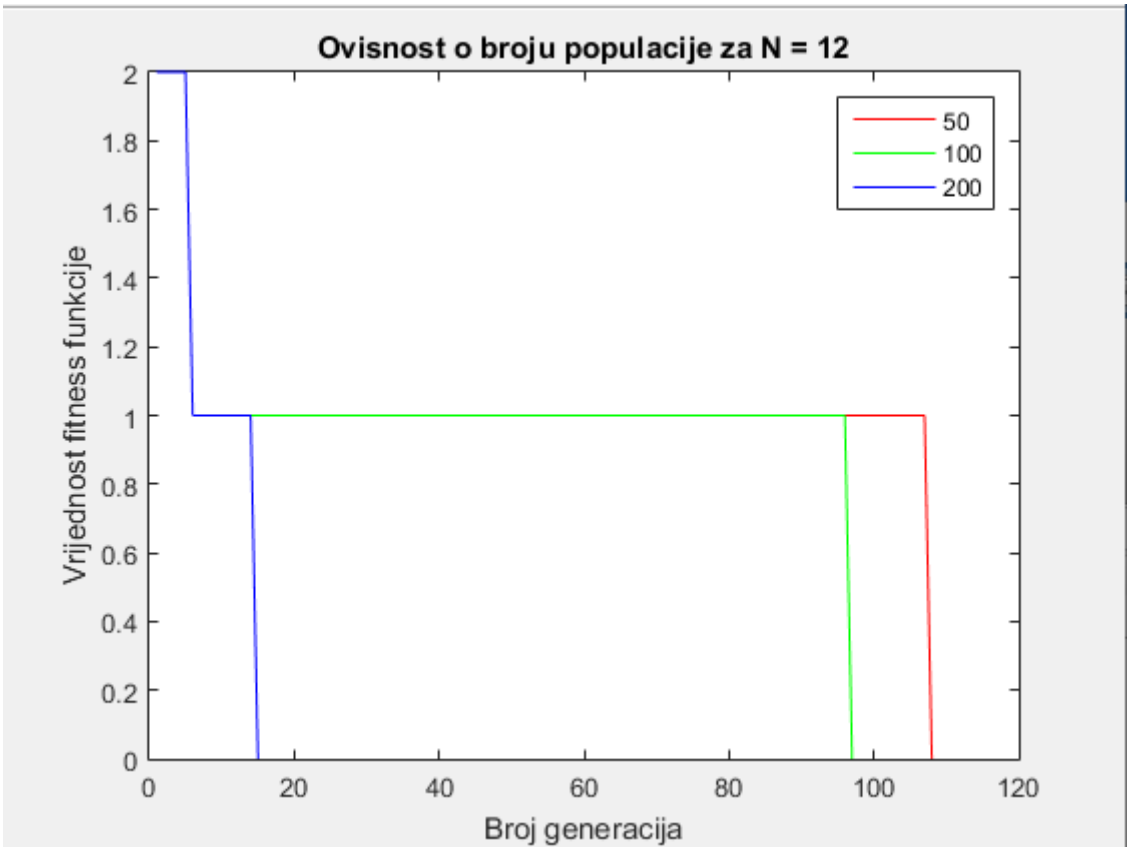
Slika 8. Prikaz ovisnosti o broju elitnih članova za N = 24

Permutacijsko kodiranje

OVISNOST O BROJU POPULACIJE

Kodiranje	permutacijsko		
Broj dama	12		
Broj generacija	10 000		
Broj elitnih članova	4		
Postotak mutacije[%]	4		
Veličina populacije	50	100	200
Broj generacija da bi se dobilo rješenje	28, 109, 2, 911, 13	15, 1599, 8, 98, 11	14, 13, 16, 15, 178
Medijan broja generacija da bi se dobilo rješenje	109	98	16

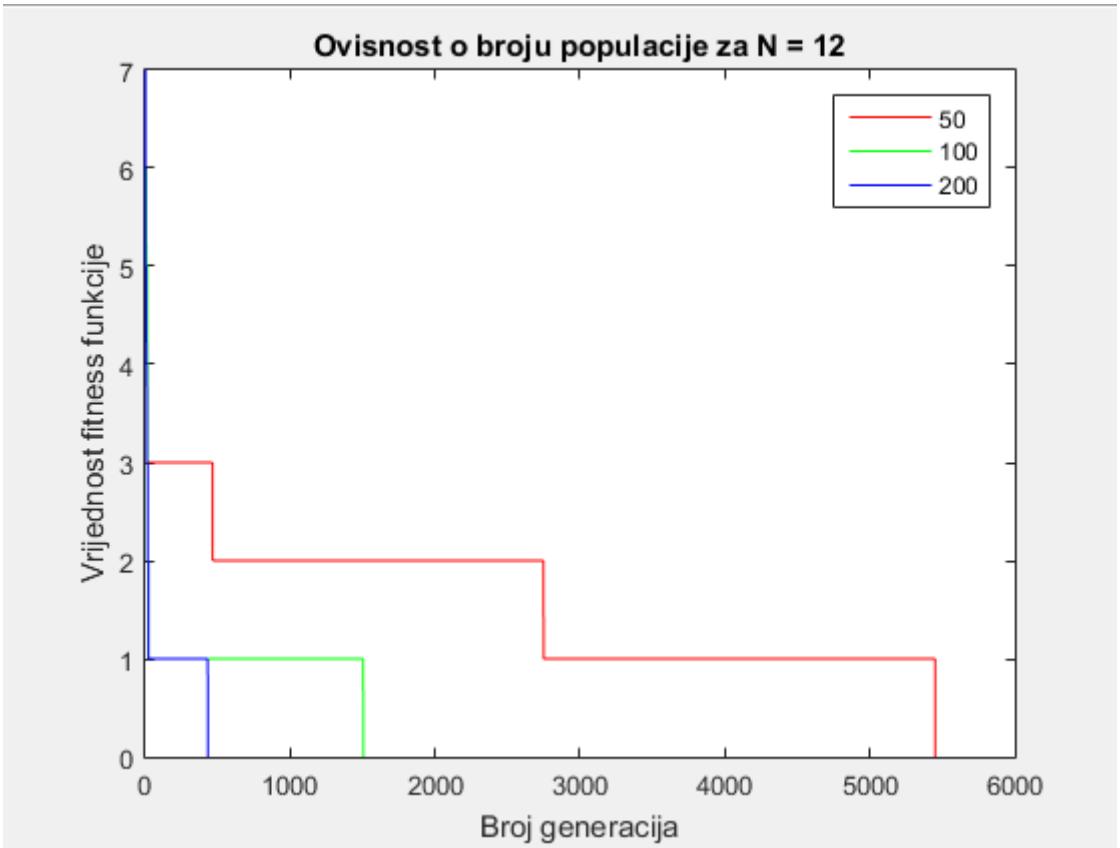
Tablica 7. Ovisnost o broju populacije za N = 12



Slika 9. Prikaz ovisnosti o broju elitnih članova za N = 12

Kodiranje	permutacijsko		
Broj dama	24		
Broj generacija	20 000		
Broj elitnih članova	4		
Postotak mutacije[%]	4		
Veličina populacije	50	100	200
Broj generacija da bi se dobilo rješenje	5452, 2605, 6275, 3243, 6357	1506, 9296, 458, 929, 837	433, 32, 3389, 32, 4811
Medijan broja generacija da bi se dobilo rješenje	5452	1506	433

Tablica 8. Ovisnost o broju populacije za N = 24

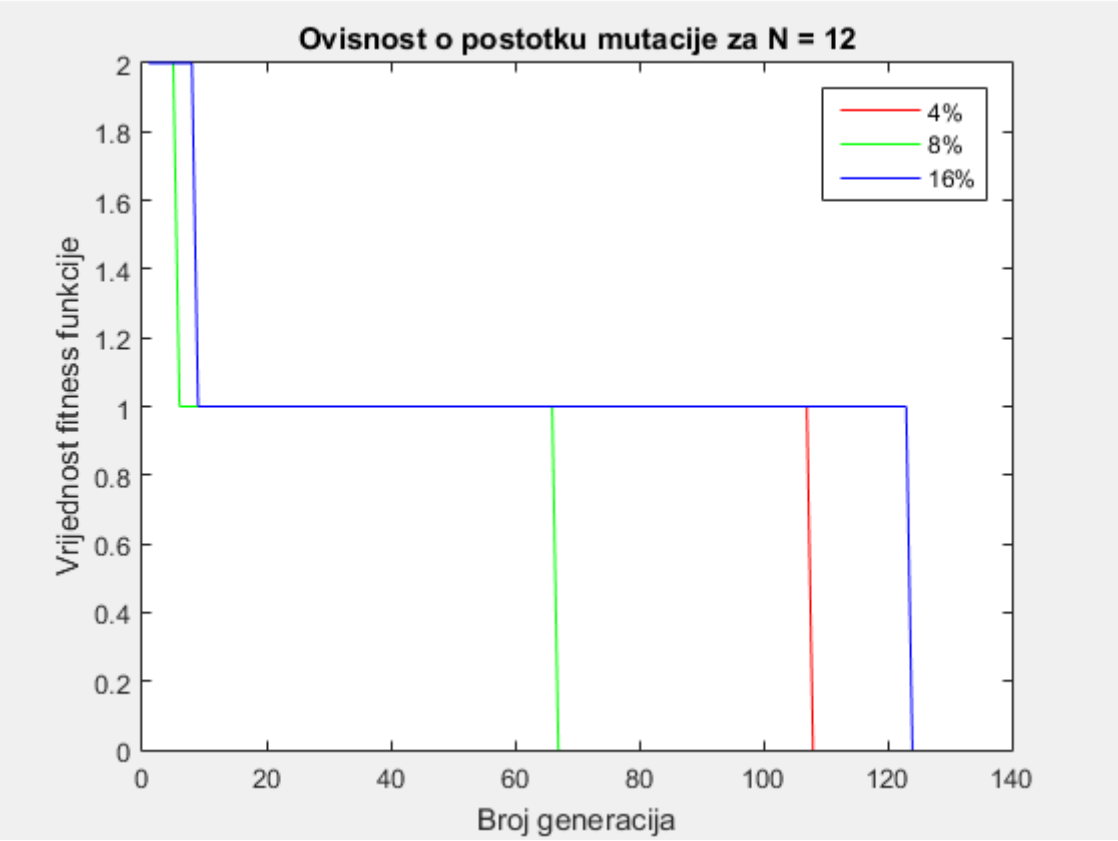


Slika 10. Prikaz ovisnosti o broju elitnih članova za N = 24

OVISNOST O POSTOTKU MUTACIJE

Kodiranje	permutacijsko		
Broj dama	12		
Broj generacija	10 000		
Veličina populacije	50		
Broj elitnih članova	4		
Postotak mutacije[%]	4	8	16
Broj generacija da bi se dobilo rješenje	28, 109, 2, 911, 13	16, 68, 75, 4, 100	125, 29, 1211, 10, 41
Medijan broja generacija da bi se dobilo rješenje	109	68	125

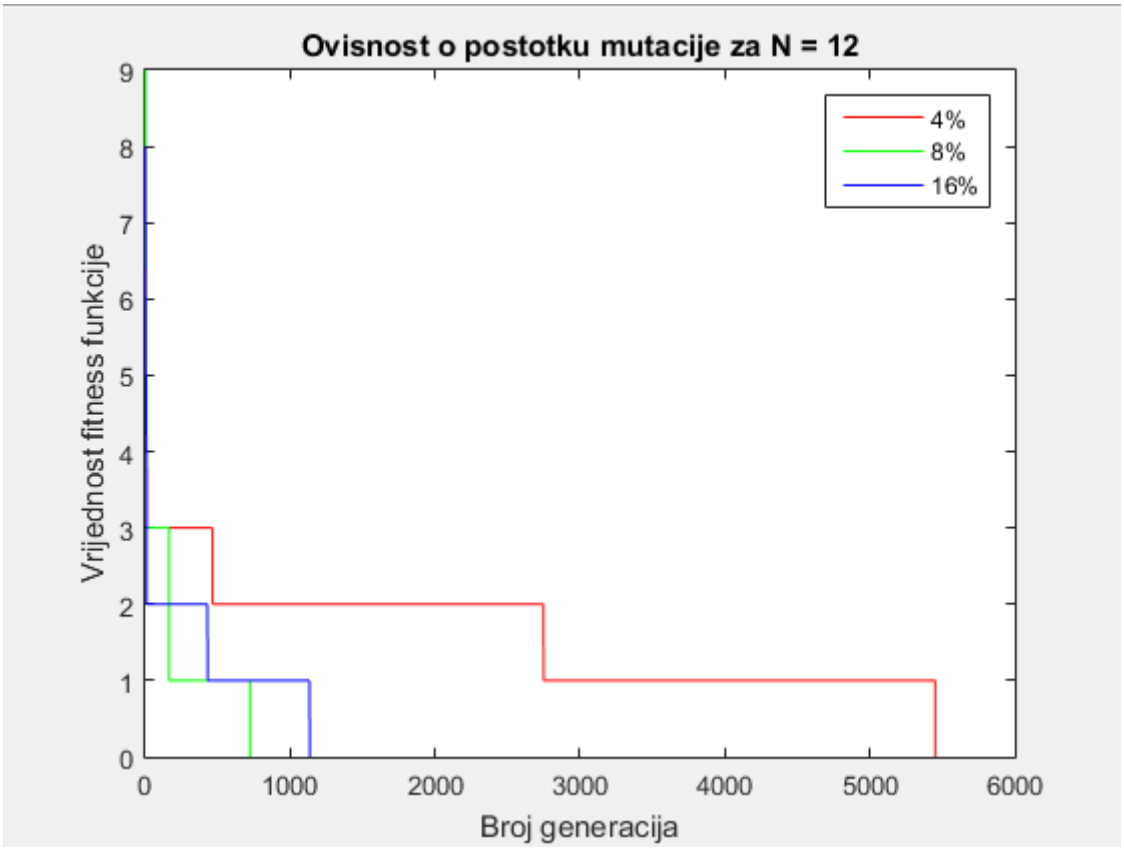
Tablica 9. Ovisnost o postotku mutacije za N = 12



Slika 11. Prikaz ovisnosti o postotku mutacije za N = 12

Kodiranje	permutacijsko		
Broj dama	24		
Broj generacija	20 000		
Veličina populacije	50		
Broj elitnih članova	4		
Postotak mutacije[%]	4	8	16
Broj generacija da bi se dobilo rješenje	5452, 2605, 6275, 3243, 6357	163, 2302, 1166, 724, 309	1136, 690, 4935, 673, 118
Medijan broja generacija da bi se dobilo rješenje	5452	724	1136

Tablica 10. Ovisnost o postotku mutacije za N = 24

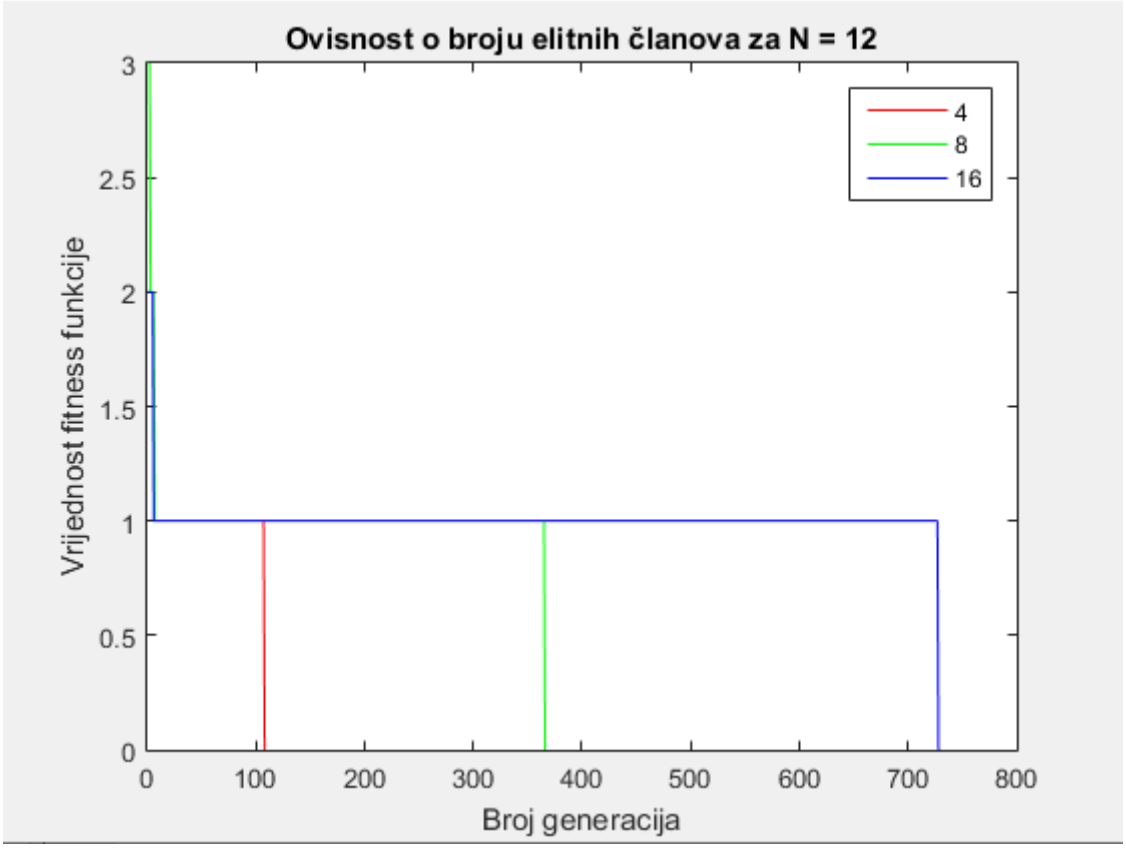


Slika 12. Prikaz ovisnosti o postotku mutacije za N = 24

OVISNOST O BROJU ELITNIH ČLANOVA

Kodiranje	permutacijsko		
Broj dama	12		
Broj generacija	10 000		
Veličina populacije	50		
Postotak mutacije[%]	4		
Broj elitnih članova	4	8	16
Broj generacija da bi se dobilo rješenje	28, 109, 2, 911, 13	3855, 9, 34, 367, 4	12, 386, 729, 4461, 137
Medijan broja generacija da bi se dobilo rješenje	109	367	729

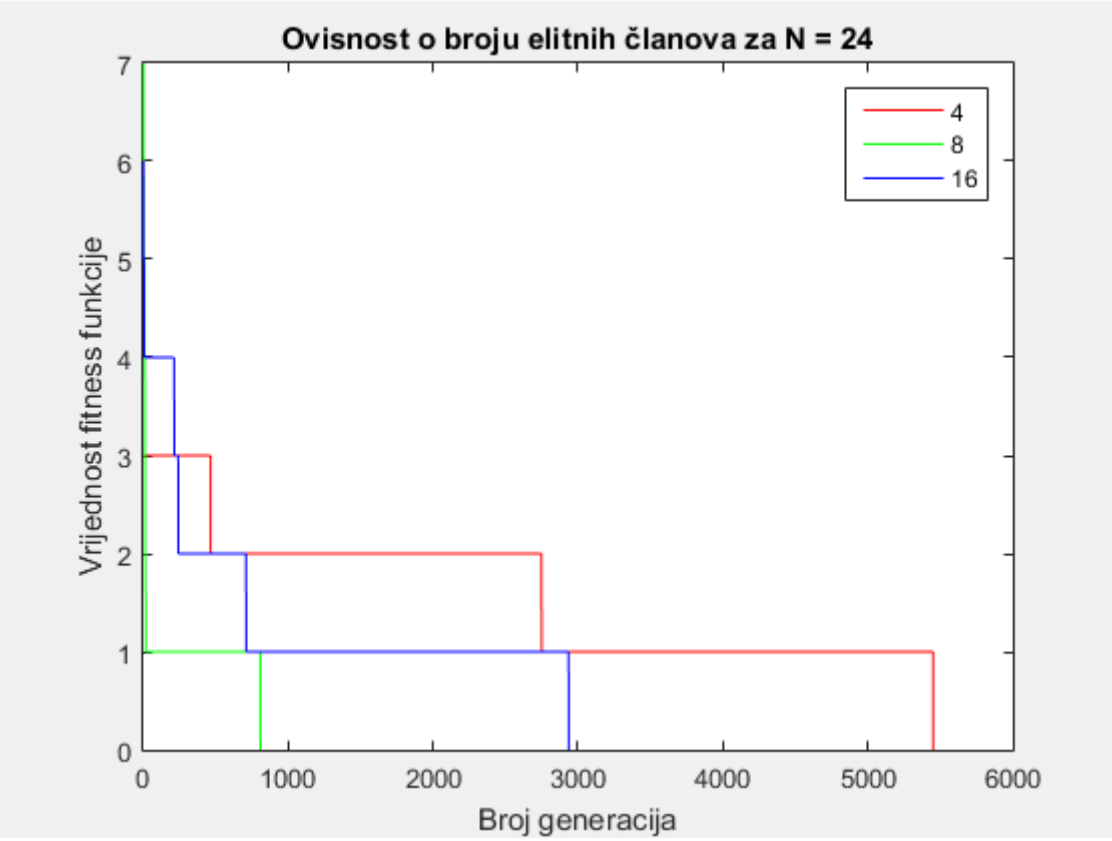
Tablica 11. Ovisnost o broju elitnih članova za N = 12



Slika 13. Prikaz ovisnosti o broju elitnih članova za N = 12

Kodiranje	cjelobrojno		
Broj dama	24		
Broj generacija	20 000		
Veličina populacije	50		
Postotak mutacije[%]	4		
Broj elitnih članova	4	8	16
Broj generacija da bi se dobilo rješenje	5452, 2605, 6275, 3243, 6357	809,704,972,830,730	2937,3672,2958,3582,1536
Medijan broja generacija da bi se dobilo rješenje	5452	809	2937

Tablica 12. Ovisnost o broju elitnih članova za N = 24



Slika 14. Prikaz ovisnosti o broju elitnih članova za N = 24

ZAKLJUČAK

Dorada i korištenje genetskog algoritma naučila nas je više o utjecaju parametara na brzinu pronalaska točnog rješenja. Za bolje predočavanje i lakše uočavanje njihovog utjecaja koristio se logaritamski grafički prikaz.

Iz vježbe se može zaključiti da je permutacijsko kodiranje puno brže od cjelobrojnog te da brzina pronalaženja rješenja uvelike ovisi o populaciji.

Povećavanjem populacije povećava se i brzina pronalaženja rješenja jer je veća šansa da neka od jedinki sadrži rješenje.

Mutacija može pozitivno i negativno utjecati na jedinke. Negativno utječe ukoliko neka od jedinki sadrži rješenje, a ona ga promjeni. Manje mutacije brže dovode do rješenja.

Broj elitnih članova utječe na brzinu rješavanja tako što ukoliko je on prevelik, previše jedinki prenosi se dalje i vrijeme rješavanja raste. Ukoliko je broj elitnih članova premali, velika je mogućnost da će se i one dobre jedinke izgubiti.

Cijeli postupak se morao ponoviti barem 5 puta kako bi dobivena rješenja bila što točnija.