SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3313

RJEŠAVANJE PROBLEMA RASPOREĐIVANJA RADNE SNAGE UPORABOM EVOLUCIJSKIH ALGORITAMA

Josip Feliks

Zagreb, lipanj 2013.

Sadržaj

1. Uvod

Raspoređivanje radne snage, odnosno izrada rasporeda za određen broj ljudi, problem je s kojim se susreće mnogo ljudi i koji im zadaje velike glavobolje. Potrebno je pronaći, odnosno izraditi, raspored koji mora zadovoljiti određene uvjete, a složenost izrade raste eksponencijalno s povećanjem broja osoba i duljine samog rasporeda. To je navelo ljude da, umjesto da izrađuju samostalno, zadaju taj kompleksan zadatak računalu. Potrebno je napomenuti da osim što je bitna kvaliteta rasporeda, bitno je i vrijeme njegovog stvaranja. Stoga se pokušava pronaći meki kompromis između kvalitete i trajanja stvaranja. Tehnika uporabljena u ovom radu su evolucijski algoritmi koji su se pokazali dostojnim da obavljaju ovaj vrlo složen zadatak.

U ovom radu pokušat će se stvoriti raspored radnih smjena za medicinsko osoblje u trajanju od četiri tjedna. Rad se sastoji od sedam poglavlja. Nakon ovog uvodnog poglavlja slijedi drugo poglavlje u kojem su opisani evolucijski odnosno genetski algoritmi te genetski operatori koji su korišteni kao tehnika za dobivanje konačnog rješenja. U trećem poglavlju je opisan konkretan problem, čvrsta i laka ograničenja te alati koji su korišteni prilikom implementacije. U četvrtom poglavlju su rezultati testiranja koji su dobiveni uporabom različitih parametara te prikaz konačnog rasporeda. U petom poglavlju je uspoređen ovaj rad s radom [9]. U šestom poglavlju se nalazi zaključak koji je proizašao nakon pisanja ovog rada. U sedmom poglavlju je navedena literatura.

2. Evolucijski algoritmi

2.1 Općenito o evolucijskim algoritmima

Unutar područja umjetne inteligencije nalaze se evolucijski algoritmi koji imitirajući proces evolucije u prirodi rješavaju kompleksne probleme [5]. Ovisno o problemu koriste se različite metode evolucijskih algoritama, a neke od njih su:

1. Genetski algoritmi
2. Genetsko programiranje
3. Evolucijsko programiranje
4. Evolucijske strategije
5. Diferencijalna evolucija
6. Itd.

2.2 Genetski algoritmi

Genetski algoritmi (GA) su heurističke metode pretraživanja i optimiranja koje simuliraju proces prirodne evolucije [7]. Postali su popularni kroz rad Johna Hollanda u ranim sedamdesetim godinama 20. stoljeća. Sve do sredine osamdesetih godina 20. stoljeća i prve internacionalne konferencije o GA u Pittsburgh, Pennsylvania, istraživali su se i proučavali velikim dijelom samo teoretski. Kako je interes rastao, u kasnim osamdesetim godinama 20. stoljeća General Electric je počeo prodavati prvi proizvod baziran na GA, računalni set alata za industrijske procese [4].

Da bi živa bića u prirodi opstala, neprestano moraju evoluirati. Procesom evolucije geni koji su nepotrebni izumiru, a ostaju oni koji su potrebni biću da preživi. Isto tako, kao što izumiru nepotrebni geni, nastaju i novi geni koji olakšavaju preživljavanje. Novi geni nastaju kroz selekciju, križanje i mutaciju starih gena.

Prema Hollandu genetski algoritam se sastoji od populacije jedinki. Svaka jedinka ima svoju sposobnost preživljavanja koja se u GA naziva dobrota i na temelju nje se obavlja selekcija jedinki. Jedinke sa manjom sposobnosti

preživljavanja izumiru, a one sa većom prelaze u novu generaciju pritom se križajući i stvarajući novu jedinku.

Genetski algoritmi su iterativni, to jest izvođenje se sastoji od određenog broja ponavljanja. Svakom iteracijom dolazi se sve bliže optimalnom rješenju. Genetski algoritmi se najčešće koriste na području učenja neuronskih mreža, izradi rasporeda, pronalaženju najkraćeg puta, itd.

2.2.1 Inicijalizacija

Inicijalizacija je postupak gdje se u početku izvođenja algoritma najčešće stvara slučajna populacija jedinki, premda, početna populacija može biti i rješenje nekog prethodnog procesa. Veličina populacije kao i ostali parametri (duljina stagnacije, vrijeme izvođenja, maksimalan broj generacija, itd.) se zadaju prije početka izvođenja.

2.2.2 Jedinka

Jedinka ili kromosom predstavlja potencijalno rješenje problema. Postoji više načina prikaza jedinke, a koristi se onaj koji najbolje odgovara zadanom problemu [1]. Jedinka može biti prikazana kao:

1. Prirodni binarni kod (Slika 2.1)
2. Permutacija (Slika 2.2)
3. Vektor s realnim komponentama (Slika 2.3)
4. Itd.

Jedinka	1001000111100010 10

Slika 2.1: Zapis jedinke binarnim kodom

Jedinka	3 4 1 2 5 8 0 9 6 8 7

Slika 2.2: Zapis jedinke permutacijom

Jedinka	3.14 2.7 1.4 5.23 7.8

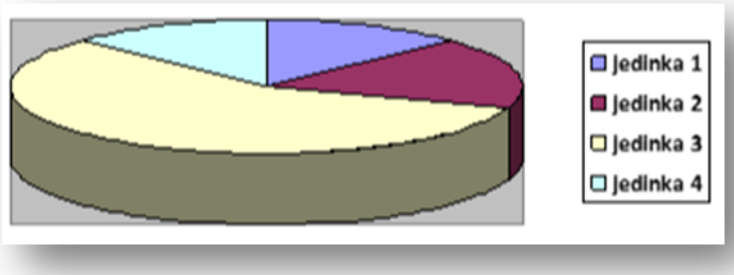
Slika 2.3: Zapis jedinke kao vektor s realnim komponentama

2.2.3 Selekcija

Svrha selekcije je očuvanje dobrih jedinki i njihovo prenošenje u sljedeću generaciju [6]. Selekcijom se odabiru jedinke roditelji od čijih će dijelova nastati jedinka dijete kao kombinacija roditelja. Postoji više vrsta selekcije, od kojih su neke:

1. Jednostavna selekcija (Slika 2.4)
2. Turnirska selekcija
3. Eliminacijska selekcija

Jednostavna selekcija (engl. *fitness proportionate selection* ili engl. *roulette wheel selection*) je genetski operator čiji se način odabira može prikazati kao kotač za *roulette*. Jedina razlika je što jedinka koja ima veću dobrotu ima veće šanse da bude odabrana od one koja ima manju dobrotu. Primjer jednostavne selekcije prikazan je na slici 2.4.



Slika 2.4: Jednostavna selekcija

Turnirska selekcija (engl. *tournament selection*) radi tako da se, ovisno o veličini turnira koju postavljamo prilikom inicijalizacije, odabire *k* jedinki iz populacije. Između *k* odabranih jedinki bira najbolju te je seli u novu populaciju nad kojom će se ti genetski operatori.

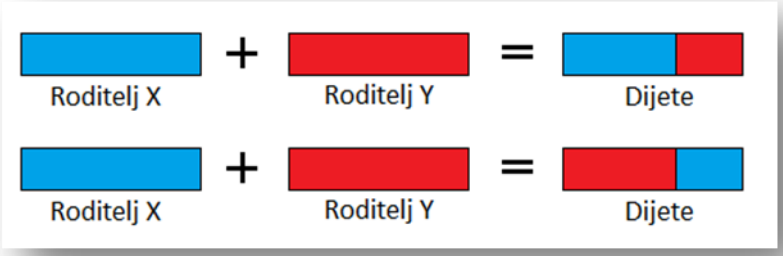
Eliminacijska selekcija (engl. *steady-state selection*) ima drukčiji princip rada od prethodne dvije selekcije. Za razliku od njih, ova selekcija odabire loše jedinke. U svakoj generaciji odabire se nekoliko najlošijih i nekoliko najboljih jedinki. Od najboljih jedinki se stvaraju nove jedinke, a najlošije jedinke izumiru.

2.2.4 Križanje

Križanje je postupak kod kojeg se uzimaju dvije jedinke iz generacije i nastaju jedna ili dvije nove jedinke u sljedećoj generaciji [3]. Postoje različiti načini križanja, a neki od njih su:

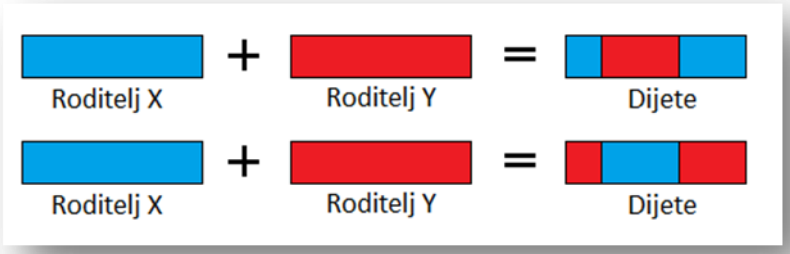
1. Križanje s jednom točkom prekida (Slika 2.5)
2. Križanje s dvije točke prekida (Slika 2.6)
3. Križanje s N točki prekida
4. Uniformno križanje (Slika 2.7)

Jedinka koja je nastala križanjem s jednom točkom prekida (Slika 2.5) nasljeđuje gene jednog roditelja sve do točke prekida, a nakon nje gene drugog roditelja.



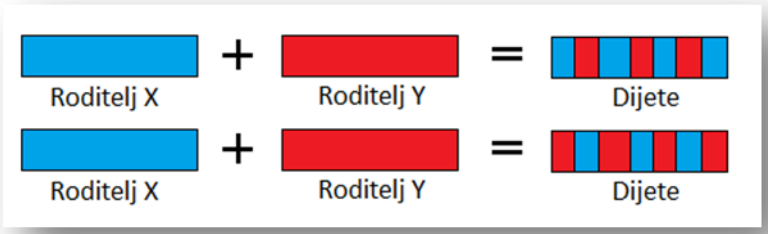
Slika 2.5: Križanje s jednom točkom prekida

Križanje s dvije točke prekida (Slika 2.6) je slično kao i križanje s jednom točkom prekida osim što dijete nasljeđuje gene jednog roditelja do prve točke prekida, zatim gene drugog roditelja do druge točke prekida i nakon toga opet gene prvog roditelja. Analogno je i za križanja s više točki prekida.



Slika 2.6: Križanje s dvije točke prekida

Kod uniformnog križanja (Slika 2.7) se osim broja točki prekida zadaje i vjerojatnost da dijete naslijedi gene roditelja. Ukoliko je vjerojatnost 50%, dijete će naslijediti 50% od jednog roditelja i 50% od drugog roditelja. Za svaki segment se može zadati različita vjerojatnost, tako da se može dogoditi da je vjerojatnost da dijete prvi segment naslijedi od jednog roditelja 80%, dok za idući segment može biti zadano 70% da naslijedi od drugog roditelja. Ako roditelji imaju neki određeni segment jednak, onda je svejedno od kojeg će roditelja dijete naslijediti taj segment.



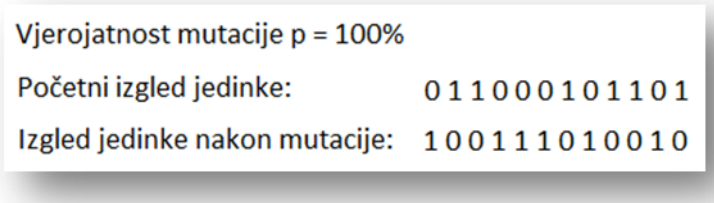
Slika 2.7: Uniformno križanje

2.2.5 Mutacija

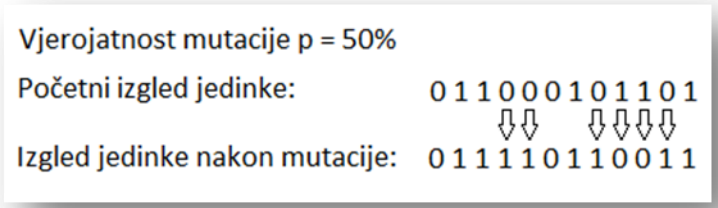
Mutacija je postupak kod kojeg se, ovisno o vjerojatnosti, događa slučajna promjena jednog ili više gena. To je unarna operacija jer je potrebna samo jedna jedinka kod koje se događa mutacija, za razliku od križanja gdje su potrebne dvije jedinke da bi nastala nova. Postoji više vrsta mutacija, a neke od njih su:

1. Jednostavna mutacija (Slika 2.8 i slika 2.9)
2. Invertirajuća mutacija
3. Rubna mutacija
4. Miješajuća mutacija

Jednostavna mutacija je takva da se u početku definira vjerojatnost mutacije gena, a onda svaki gen mutira s tom vjerojatnošću. Primjer jednostavne mutacije se nalazi na slici 2.8. i na slici 2.9.



Slika 2.8: Primjer jednostavne mutacije p=1



Slika 2.9: Primjer jednostavne mutacije p=0.5

Invertirajuća mutacija (engl. *flip bit*) radi na principu da invertira svaki bit jedinke. Ako je bit 1, nakon mutacije će biti 0 i obrnuto. Može se koristiti ako je jedinka prikazana u binarnom kodu. Invertirajuća mutacija je oblik jednostavne mutacije kod koje je vjerojatnost mutacije 100% kao na slici 2.8.

Rubna mutacija (engl. *boundary*) se može koristiti ako je zapis jedinke u obliku cijelih ili decimalnih brojeva. Radi tako da gen jedinke zamjenjuje s rubnim vrijednostima gdje je vjerojatnost odabira gornje ili donje granice 50%.
Miješajuća mutacija odabire slučajnu jedinku unutar populacije, zatim slučajno odabire prvu i drugu granicu, a gene između granica ili slučajno generira ili ih međusobno izmiješa.

Mutacija je dobra jer može pomoći prilikom traženja optimalnog rješenja, tako da, ako se u trenutku traženja generacija zaglavi u nekom lokalnom optimumu, uz pomoć mutacije nastaje nova jedinka koja pokreće pretragu u drugom smjeru [2].

3. Problem raspoređivanja radne snage

3.1 Definicija problema

Problem koji je razmatran u ovom radu odnosi se na izradu rasporeda smjena za medicinsko osoblje u trajanju od četiri tjedna. Medicinsko osoblje je podijeljeno na više i srednje sestre. Broj različitih smjena je tri. Postoje jutarnja, dnevna i noćna smjena. Potrebno je izraditi raspored koji će zadovoljavati određena ograničenja.

3.2 Ograničenja

Budući da se izrađuje raspored za određeni odjel u bolnici, konačno rješenje mora zadovoljavati određena ograničenja. Ograničenja se dijele na čvrsta i laka ovisno o važnosti njihovog ispunjenja. Za svako prekršeno ograničenje slijedi kazna koja je teška ovisno o kategoriji ograničenja i broju koliko puta je prekršeno.

3.2.1 Čvrsta ograničenja

Čvrsta (teška) ograničenja su ona koja ne smiju biti prekršena [8], to jest ako se prekrše dobiju vrlo veliku kaznu koja jedinci daje vrlo male izglede da će opstati i prijeći u novu generaciju. U praksi takve jedinke vrlo brzo izumru.
Čvrsta ograničenja su:

1. Ne smije postojati smjena u kojoj nitko ne radi
2. Osoba smije raditi najviše jednu smjenu u danu

Čvrsta ograničenja se dodatno dijele u dvije kategorije gdje svaka kategorija ima svoj koeficijent. Čvrsta ograničenja će se u daljnjem tekstu označavati s oznakom T{broj}. Tako će čvrsto ograničenje koje je navedeno pod 1. biti T1, a pod 2. T2.

3.2.2 Laka ograničenja

Laka ograničenja se smiju prekršiti, premda, za svako prekršeno ograničenje slijedi određena kazna koja je povezana s određenom kategorijom. Laka ograničenja se dijele u tri kategorije koje su prikazane u nastavku.

Kategorija A:

1. Osoba ne smije tjedno raditi manje od pet smjena
2. Osoba ne smije tjedno raditi više od sedam smjena
3. U jutarnjoj smjeni mora raditi minimalno jedna viša sestra

Kategorija B:

1. Osoba nakon odrađene smjene mora imati slobodno minimalno sljedeće dvije smjene
2. Osoba smije imati maksimalno dva uzastopna slobodna dana
3. U jutarnjoj smjeni moraju raditi minimalno četiri osobe
4. U dnevnoj smjeni moraju raditi minimalno tri osobe
5. U noćnoj smjeni moraju raditi minimalno dvije osobe

Kategorija C:

1. Nakon odrađene noćne smjene, osoba mora imati minimalno jedan slobodan dan
2. Osoba smije raditi maksimalno dvije noćne smjene tjedno

U daljnjem tekstu se ograničenje broj 1 kategorije A označava kao A1. Analogno se odnosi za sve ostale kategorije i njihova ograničenja. Konačno rješenje će biti ono s najmanjom kaznom.

3.3 Opis metode procjene

Nakon što su definirani parametri rasporeda te ograničenja, potrebno je definirati i metodu procjene na temelju koje će se raspored razvijati kroz generacije algoritma da bi u konačnici bio što kvalitetniji. Svaki raspored se procjenjuje na temelju jednadžbe (3.1).

Rezultat jednadžbe:

$$rezultat = 10000 - \big(500 * (2 * T1 + T2) + 50 * (A1 + A2 + A3) + 20 * (B1 + B2 + B3 + B4 + B5) + 5 * (C1 + C2)\big) \quad (3.1)$$

je vrijednost dobrote pojedine jedinke. Simboli oblika {slovo}{broj} u jednadžbi (3.1) označavaju točno određeno ograničenje. Koeficijenti koji su zadani u jednadžbi, osim broja 10000, govore koliko je pojedino ograničenje bitno. Pokraj oznaka za čvrsta ograničenja su koeficijenti 1000 odnosno 500, a oni označavaju da ukoliko je prekršeno ograničenje T1, kazna je dvostruko veća nego da je prekršeno ograničenje T2. Ovisno o kategoriji lakih ograničenja pridijeljeni su različiti koeficijenti. Ukoliko je prekršeno ograničenje kategorije A, gleda se koliko puta je prekršeno i taj broj se množi sa koeficijentom 50. Analogno je i za ostala ograničenja.

3.4 Prikaz rasporeda u algoritmu

Ukoliko je raspored duljine četiri tjedna gdje se svaki dan sastoji od tri smjene, a veličina osoblja je 20, njegov izgled u algoritmu je niz nula i jedinica duljine 1680 (broj_tjedana * broj_dana_u_tjednu * broj_smjena * veličina_osoblja). Prve 84 znamenke (broj_tjedana * broj_dana_u_tjednu * broj_smjena) odnosi se na prvu medicinsku sestru, zatim sljedeće 84 znamenke na drugu i tako dalje. Jedan dan, koji se sastoji od tri smjene, je prikazan s tri znamenke. Ako medicinska sestra radi jutarnju smjenu, taj dan je prikazan kao “100“, ako radi dnevnu “010“ te ako radi noćnu smjenu “001“. Na taj način algoritam lako može promjenjivati genetske operatore s ciljem dobivanja kvalitetnijeg rasporeda.

3.5 Implementacija

Za potrebe implementacije korišten je programski jezik *Java* te *Eclipse* kao integrirano razvojno okruženje. Budući da konkretan problem nije implementacija alata potrebnih za evolucijsko računanje već njihovo korištenje u svrhu pronalaska

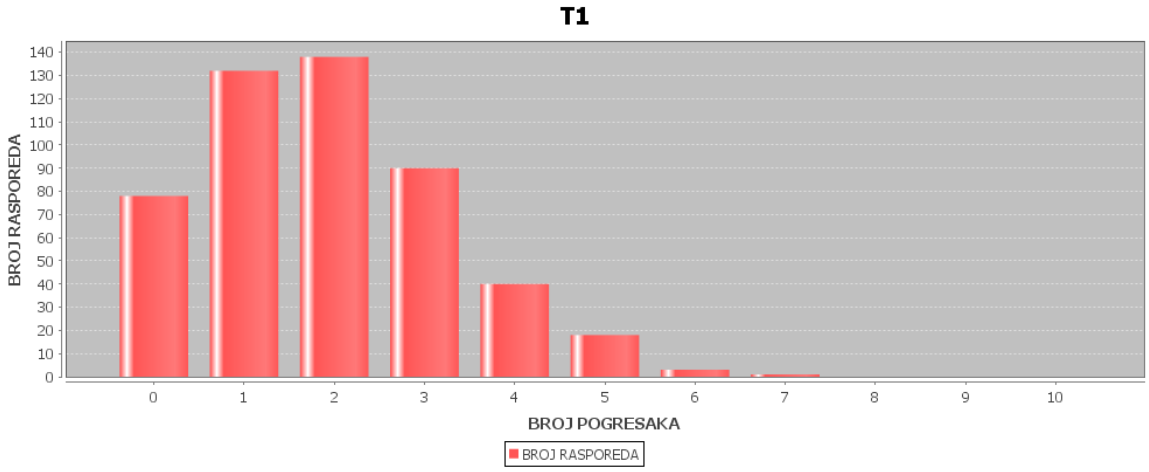
optimalnog rasporeda, pribjeglo se korištenju radnog okruženja ECF (*Evolutionary Computation Framework*) koji ima sve potrebno i na jednostavan način se uspjelo doći do rezultata. Kao oblik zapisa jedinke koristio se *BitString. BitString* je oblik zapisa jedinke kao niza nula i jedinica. Za svako rješenje, od inicijalnih do konačnog, se računa vrijednost dobrote (engl. *fitness value*) i na temelju te vrijednosti se rješenja razlikuju, od lošijih do boljih. Dobrota svakog potencijalnog rješenja se procjenjuje u metodi procjene.

4. Rezultati

4.1 Prikaz raspodjele inicijalnih rasporeda za čvrsta ograničenja

Za potrebe ocjene težine problema i procjene učinkovitosti predložene metode, provedena je analiza narušenosti ograničenja na skupu slučajno generiranih jedinki, kakav se dobiva na početku svakog pokretanja algoritma.

Nakon što je stvorena početna populacija jedinki, svaka od njih se ispituje i zapisuje se koliko puta je pojedino čvrsto ograničenje prekršeno te se na temelju tih podataka stvara graf. Na apscisi grafa su vrijednosti od 0 do 15 koje označavaju broj pogrešaka, a na ordinati su vrijednosti od 0 do broja veličine populacije. Graf služi da se vidi za svako ograničenje, kod koliko je jedinki i koliko puta prekršeno. Primjer takvog grafa je na slici 4.1.



Slika 4.1: Graf za čvrsto ograničenje T1

Ako se na primjer dogodi da je u grafu za ograničenje T1 stupac čiji vrh je na koordinatama x=5 i y=20, to znači da u inicijalnoj populaciji, broj rasporeda kod kojih je ograničenje T1 prekršeno pet puta je dvadeset. Ovakvim grafom može se vidjeti za svako pojedino ograničenje, koliko puta je prekršeno. Kod T2 postoji samo jedan stupac, u x=0, jer se prilikom kreiranja inicijalnog rasporeda odmah postavlja da to ograničenje bude zadovoljeno.

4.2 Izbor algoritma

Algoritam koji će biti korišten je turnirski sa eliminacijskom selekcijom (engl. *steady-state tournament*). Pseudo-kôd algoritma prikazan je na slici 4.3.

```
single generation {
 repeat(deme size times) {
  randomly add <nTournament_> individuals to the tournament;
  select the worst one in the tournament;
  randomly select two parents from remaining ones in the tournament;
  replace the worst with crossover child;
  perform mutation on child;
 }
}
```

Slika 4.2: Pseudo-kôd algoritma (preuzeto s [11])

Algoritam radi tako da za svaku generaciju pokreće određen broj turnira kojih ima koliko je veličina populacije. Svakom turniru pristupa broj jedinki, koliko je zadano prilikom postavljanja parametara (najčešće tri), između kojih se odabire najlošija i izbacuje, a na njeno mjesto dolazi nova jedinka koja je nastala kao dijete od slučajno odabranih roditelja iz skupine preostalih jedinki u tom turniru. Razlog zbog kojeg se kao veličina turnira najčešće odabire skupina od tri jedinke je taj kad bi se odabrao veći broj, lošije jedinke ne bi imale šanse proći u novu generaciju, a samim time bi se smanjila veličina prostora pretrage i rješenje bi vrlo brzo konvergiralo nekom lokalnom optimumu. Praksa je pokazala da ne valja imati u generaciji samo dobre jedinke, već da su konačna rješenja u velikom broju slučajeva bolja ukoliko se i loše jedinke prenose u nove generacije jer na taj način postoji veća raznolikost, a samim time može prilikom križanja nastati bolja nova jedinka.

4.3 Utjecaj veličine populacije

Ovisno o veličini populacije, konačno rješenje je “bolje“ ili “lošije“. Osim o vrijednosti dobrote potrebno je i razmotriti vremensko trajanje stvaranja rasporeda za svaku veličinu populacije. Veličine populacije koje su uzete u razmatranje su

100, 200, 300, 500 i 1000. Stvaranje rasporeda je pokrenuto dvadeset puta za svaku pojedinu veličinu populacije i vrijednosti dobrote su prikazane na slici 4.4.

Slika 4.3: Box plot za različite veličine populacije

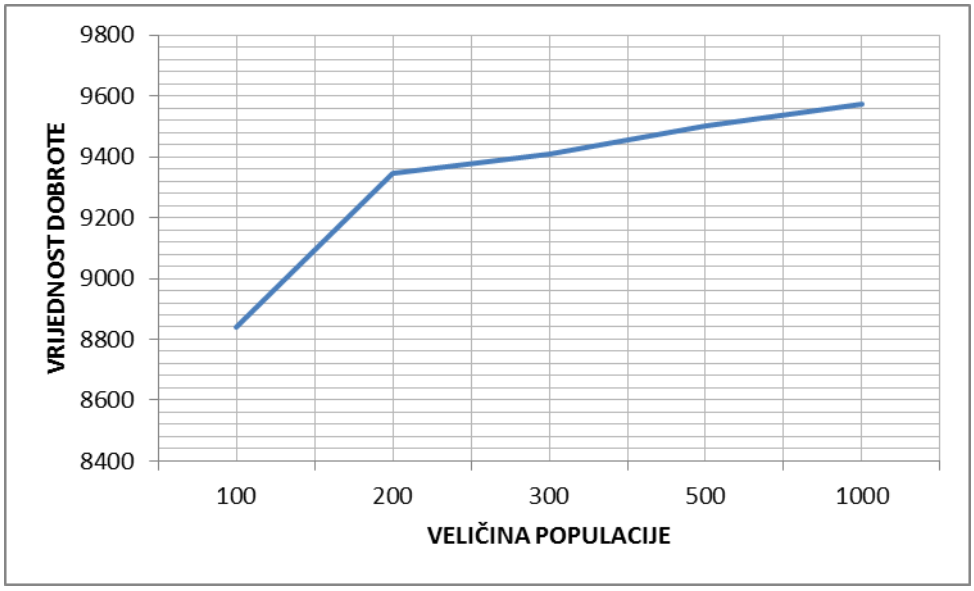
Prosjek za svaku veličinu populacije prikazan je u tablici 4.1 i na slici 4.5.

Tablica 4.1: Utjecaj veličine populacije

	VELIČINA POPULACIJE				
	100	200	300	500	1000
VRIJEDNOST DOBROTE	8840.5	9344	9406.75	9502.75	9574.25

Dodatni parametri koji su korišteni prilikom stvaranja:

- duljina stagnacije = 5000,
- broj generacija = 10 000,
- vjerojatnost mutacije = 0.3,
- maksimalan broj sekundi prije nego što se prekine generiranje je 3000.

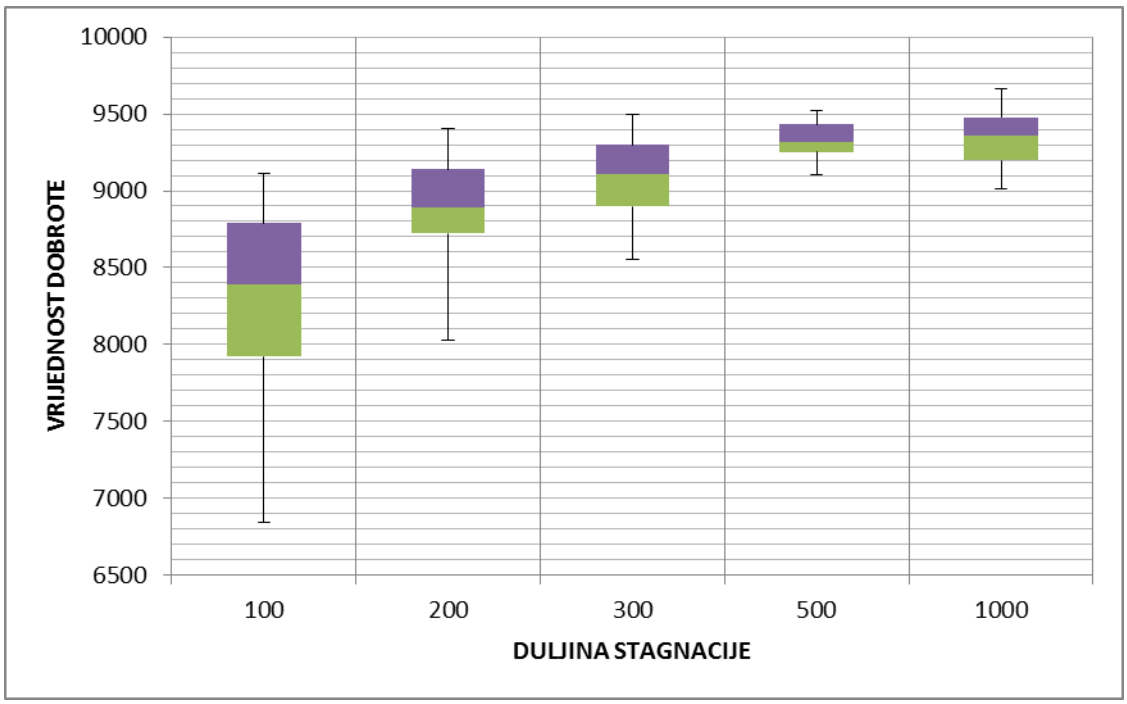


Slika 4.4: Grafički prikaz tablice 4.1

Iz slike 4.4 se može iščitati da je vrijednost dobrote najveća kod populacije od 1000 jedinki. Međutim, potrebno je znati da je generiranje prekinuto nakon što je vrijeme stvaranja premašilo maksimalan broj sekundi (3000). Osim optimalnosti rasporeda, bitno je i vrijeme potrebno da se raspored generira. Prosječno vrijeme izvođenja za populaciju od 500 jedinki je oko dvadeset minuta, što preračunato u sekunde iznosi oko 1200 sekundi. Razlika između vrijednosti dobrote za veličinu populacije od 500 i od 1000 jedinki mala je u odnosu na višestruko dulje trajanje stvaranja rasporeda od 1000 jedinki nego od 500, stoga je u daljnjim testiranjima korištena populacija od 500 jedinki.

4.4 Utjecaj duljine stagnacije

Duljina stagnacije, to jest broj generacija bez poboljšanja vrijednosti dobrote, uvelike utječe na kvalitetu konačnog rezultata, ali i duljinu izvođenja. S rastom duljine stagnacije, raste i vrijeme izvođenja, ali i vjerojatnost da konačan rezultat bude kvalitetniji. Stvaranje rasporeda je pokrenuto dvadeset puta za svaku pojedinu duljinu stagnacije i vrijednosti dobrote su prikazane na slici 4.6.

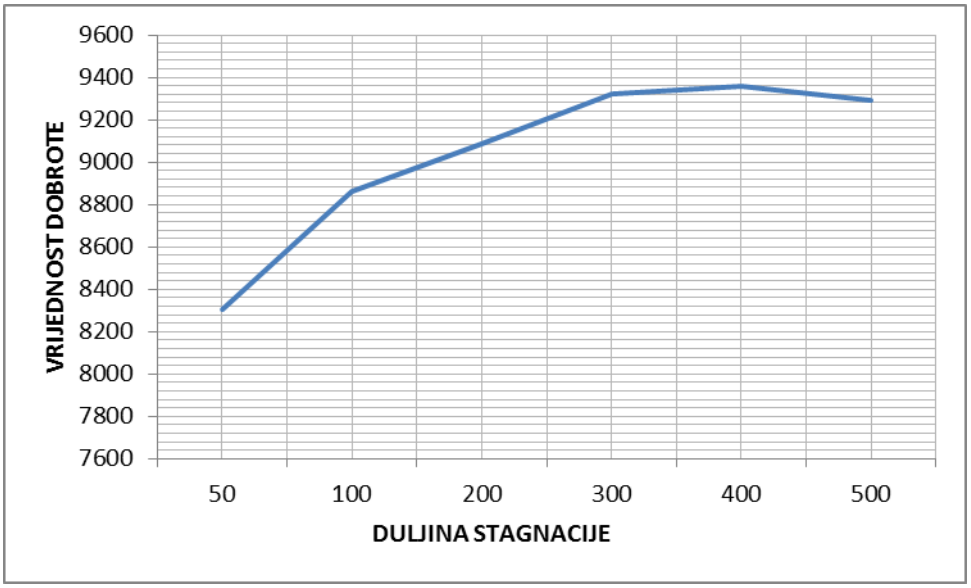


Slika 4.5: Box plot za različite duljine stagnacije

Prosjek za svaku pojedinu duljinu stagnacije prikazan je u tablici 4.2 i na slici 4.7.

Tablica 4.2: Utjecaj duljine stagnacije

	DULJINA STAGNACIJE					
	50	100	200	300	400	500
VRIJEDNOST DOBROTE	8303.5	8863.75	9086.5	9322.75	9356.5	9291.25

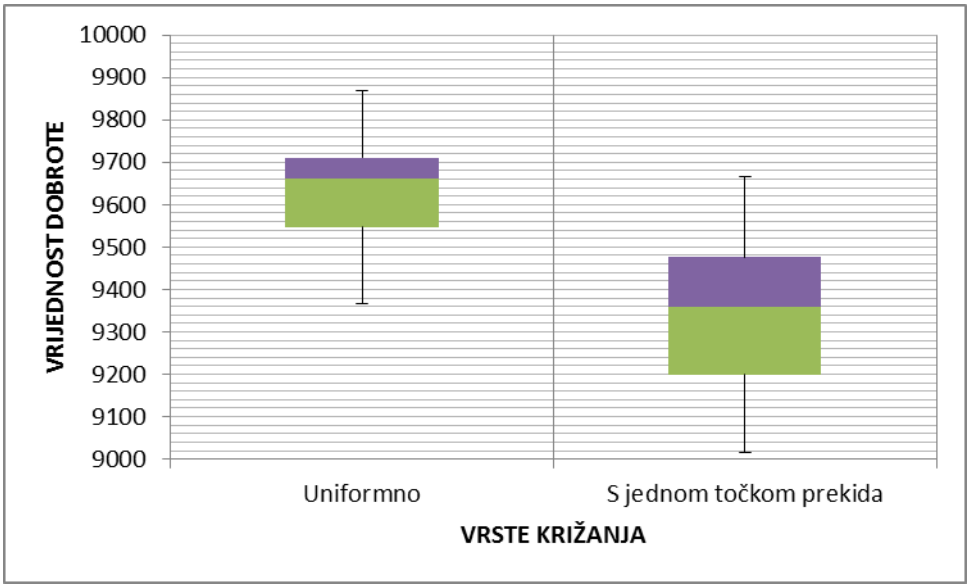


Slika 4.6: Grafički prikaz utjecaja stagnacije na rezultat

Kao i u tablici 4.2, i na slici 4.7 se vidi da je najbolji rezultat bio kad je duljina stagnacije bila podešena na 400 generacija. Iako je u pravilu sa većom duljinom stagnacije veća i dobrota, iz slike 4.7 se vidi da jedna dosta loša iteracija može poprilično utjecati na rezultat. Zbog toga ispada da je bolji rezultat za stagnaciju duljine 400 generacija nego za 500 generacija iako je to statistički gledano neznatna razlika. Potrebno je još jednom napomenuti da je prosjek za svaku pojedinu duljinu stagnacije računat na uzorku od dvadeset iteracija. Razlika između rezultata za duljinu stagnacije od 300 i 400 generacija nije prevelika, ali uzimajući u obzir da je trajanje izvođenja za stagnaciju od 400 u prosjeku samo 25% dulje nego kod 300, odlučeno je da će se kao optimalna, što zbog vremena izvođenja, što zbog vrijednosti dobrote, uzeti duljina stagnacije od 400 generacija.

4.5 Utjecaj različitih načina križanja

Postoje dva različita načina križanja koja se trenutno mogu odabrati da bi se obavilo ispitivanje. Sva mjerenja koja su do sad učinjena koristila su križanje s jednom točkom prekida, stoga će se dosadašnji rezultati preslikati u tablicu 4.3. Drugi način križanja je uniformno križanje kod kojeg dijete nasljeđuje bit roditelja ako su jednaki, a inače je bit nasumičan. Stvaranje rasporeda je pokrenuto dvadeset puta za uniformno križanje i vrijednosti dobrote su prikazane na slici 4.8.



Slika 4.7: Box plot za različite vrste križanja

Prosjeci za obje vrste križanja prikazani su u tablici 4.3.

Tablica 4.3: Utjecaj različitih vrsta križanja na rezultat

	KRIŽANJA	
	Križanje s jednom točkom prekida	Uniformno križanje
VRIJEDNOST DOBROTE	9356.5	9633

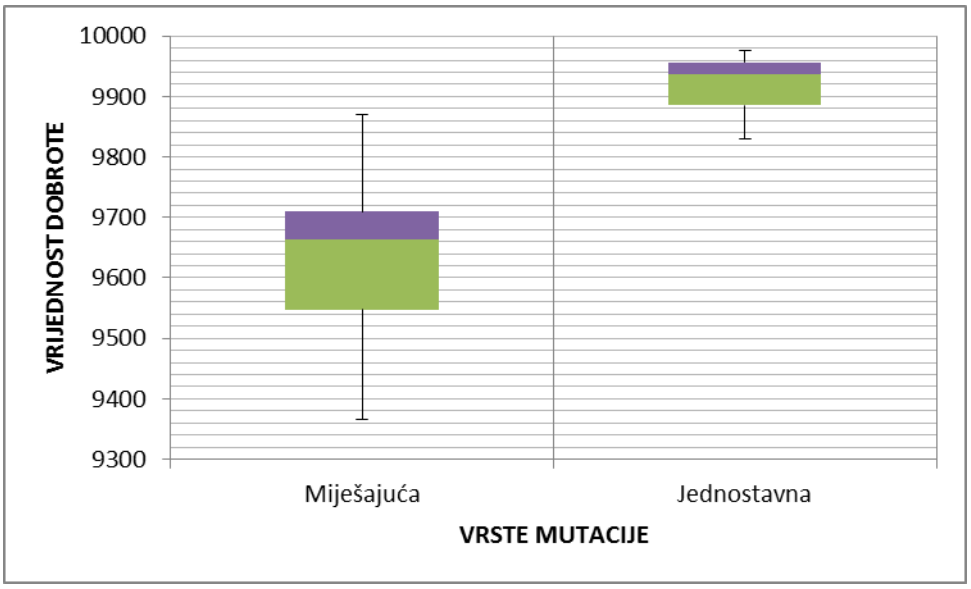
U tablici 4.3 se vidi da izbor načina križanja je veoma bitan. Uniformno križanje daje puno bolja rješenja i stoga će se u daljnjim testiranjima upravo to križanje primjenjivati.

4.6 Utjecaj različitih vrsta mutacija

Kad se dogodi da je u populaciji previše sličnih jedinki, vrlo brzo rješenje konvergira prema nekom neoptimalnom rješenju. Mutacija služi upravo zato da se uz pomoć mutirane jedinke izbjegne prebrzo konvergiranje i pretraga krene u drugom smjeru. Trenutno su na raspolaganju dvije vrste mutacije; jednostavna mutacija i miješajuća mutacija. Kod jednostavne mutacije se slučajnim odabirom uzima jedinka u kojoj bitovi mutiraju s određenom vjerojatnošću, a kod miješajuće

mutacije se odrede dvije točke u jedinci između kojih se bitovi preslaguju slučajnim odabirom, ali uz uvjet da poslije preslagivanja između te dvije točke postoji jednak broj nula i jednak broj jedinica kao i u roditeljskoj jedinci.

Rezultati koji su do sad dobiveni koristili su miješajuću mutaciju, stoga se rezultati za tu vrstu mutacije prepisuju, dok se za jednostavnu mutaciju vrijednost dobrote računa tako da se dvadeset puta pokreće izvođenje i računa prosjek. Vrijednosti su prikazane na slici 4.9.



Slika 4.8: Box plot za različite vrste mutacije

Prosjeci za određenu vrstu mutacije su prikazani u tablici 4.4.

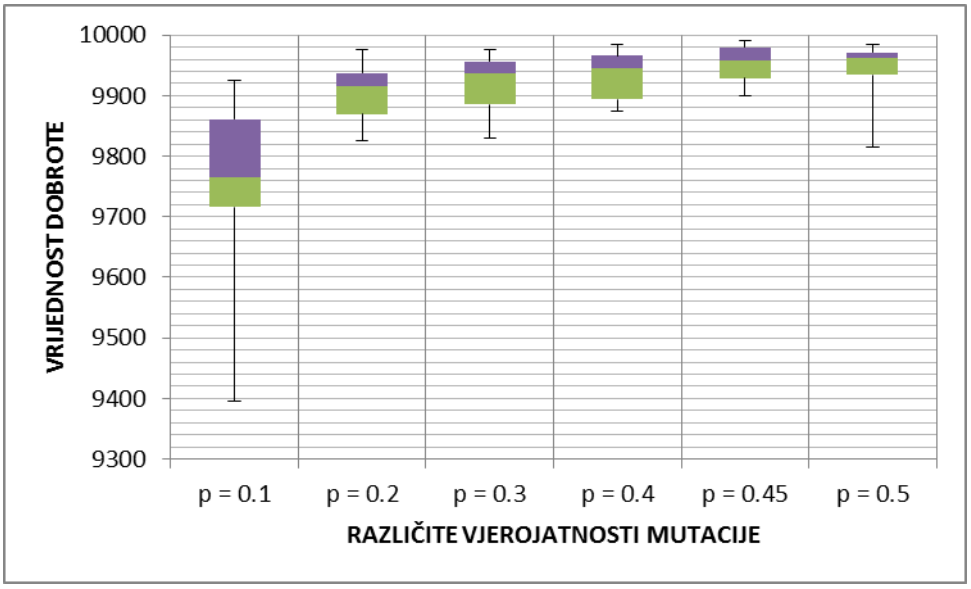
Tablica 4.4: Utjecaj različitih mutacija na rezultat

	MUTACIJA	
	Miješajuća mutacija	Jednostavna mutacija
VRIJEDNOST DOBROTE	9633	9920.25

Iz tablice 4.4 se vidi da jednostavna mutacija daje puno bolje rezultate nego miješajuća, stoga se u daljnjim računanjima primjenjuje upravo jednostavna mutacija.

4.7 Utjecaj različitih vjerojatnosti mutacija

Nakon što je odabran tip mutacije, potrebno je odrediti i vjerojatnost pri kojoj će se dogoditi mutacija. Do sad je računato sa vjerojatnošću mutacije od 0.3, no potrebno je provjeriti hoće li uz neku drugu vjerojatnost mutacije konačno rješenje biti bolje. Raspon vjerojatnosti i pripadajući rezultati vidljivi su na slici 4.10, a njihovi prosjeci u tablici 4.5.

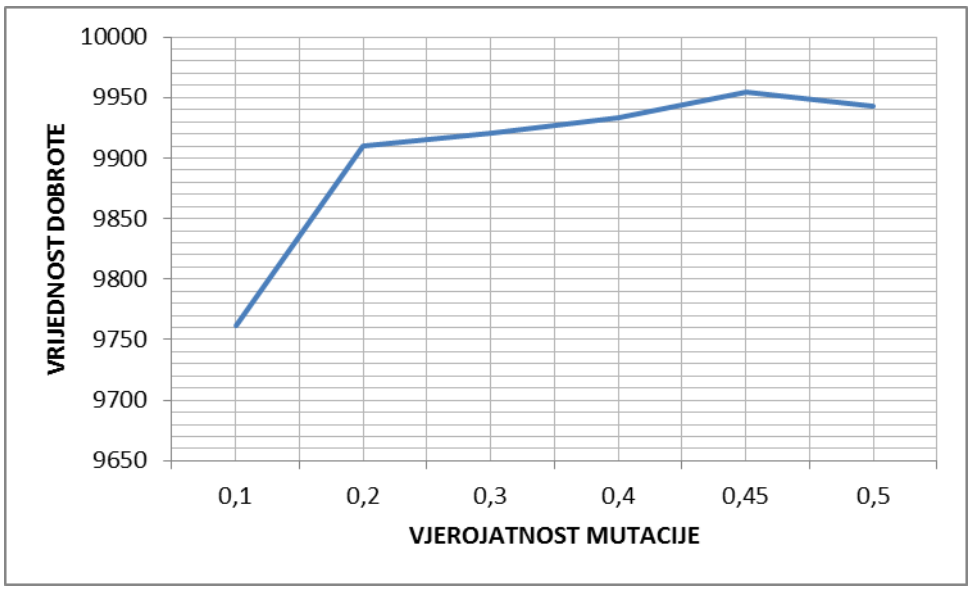


Slika 4.9: Box plot za različite vjerojatnosti mutacije

Tablica 4.5: Utjecaj vjerojatnosti mutacije na rezultat

	VJEROJATNOST MUTACIJE					
	0.1	0.2	0.3	0.4	0.45	0.5
VRIJEDNOST DOBROTE	9761.5	9909.75	9920.25	9933.75	9955	9942.5

Razlog što vjerojatnost mutacije ne prelazi 0.5 je taj što bi onda mutacija često nastupala i traženje optimalnog rješenja bi se svelo na generiranje nasumičnih jedinki. Mutacija kao takva bi izgubila svoju prvotnu namjenu. Za svaku vjerojatnost mutacije izvođenje je pokrenuto, kao i dosad, dvadeset puta i izračunat je prosjek. U tablici 4.5, kao i na slici 4.11, je vidljivo da su najbolji rezultati dobiveni ako se postavi vjerojatnost mutacije na 0.45.



Slika 4.10: Grafički prikaz utjecaja vjerojatnosti mutacije na rezultat

4.8 Utjecaj kombinacije više vrsta križanja i mutacije

Nakon što je odabrana vrsta križanja i mutacije, potrebno je provjeriti dobivaju li se bolji rezultati ako se koriste obje vrste križanja u kombinaciji s jednostavnom mutacijom te uniformno križanje i kombinacija obje vrste mutacije. Testiranje je provedeno dvadeset puta za svaki pokus, a rasponi vrijednosti su prikazani na slici 4.12. Prosjek vrijednosti dobrote prikazan je u tablici 4.6.

Slika 4.11: Box plot za različite kombinacije

Tablica 4.6: Vrijednosti dobrote za kombinacije križanja i mutacije

	KOMBINACIJE	
	Sve vrste križanja i jednostavna mutacija	Uniformno križanje i sve vrste mutacija
VRIJEDNOST DOBROTE	9926.5	9950.75

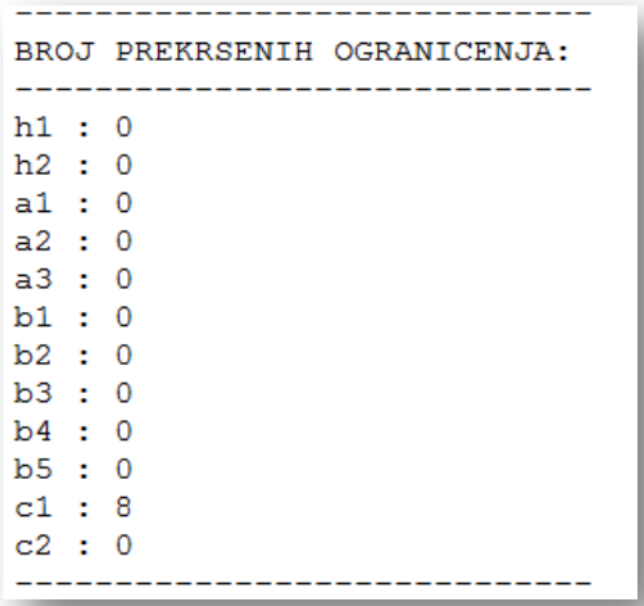
Nakon svih testiranja se vidi da jedna vrsta križanja, uniformno križanje, i jedna vrsta mutacije, jednostavna mutacija, daju bolje rezultate od raznih kombinacija križanja i mutacija.

4.8 Prikaz konačnog rasporeda

Nakon što su odabrani svi parametri uz koje se postiže najbolja vrijednost dobrote, red je da se još jednom pokrene izvođenje i prikaže konačan rezultat, no najprije pregled svih parametara koji su korišteni:

- Duljina rasporeda = 4 tjedna
- Broj smjena = 3
- Broj srednjih sestara = 15
- Broj viših sestara = 5
- Maksimalan broj generacija = 10000
- Veličina populacije = 500
- Duljina stagnacije = 400
- Vrsta algoritma = turnirski s eliminacijskom selekcijom
- Vrsta križanja = uniformno križanje
- Vrsta mutacije = jednostavna mutacija
- Vjerojatnost mutacije = 0.45
- Maksimalan broj sekundi izvođenja programa = 3000

Nakon dvije i pol minute izvođenja, stvoren je raspored. Osim samog rasporeda, dobiven je i prikaz koji pokazuje koliko je svako ograničenje puta prekršeno. Prikaz se nalazi na slici 4.13.



Slika 4.12: Prikaz liste ograničenja

Vrijednost dobrote koju ovaj raspored ima je 9960, što je blizu idealnom. Prosjek koji je dobiven nakon dvadeset pokretanja je 9955, dok je minimalna, odnosno maksimalna vrijednost 9900, odnosno 9990. Na slici 4.13 se vidi da je ograničenje C1 prekršeno osam puta. Ograničenje C1 kaže da osoba smije raditi najviše dvije noćne smjene tjedno. Ako se pogleda veličina rasporeda, može se vidjeti da to nije toliki problem, uzimajući u obzir da je to jedino ograničenje koje je prekršeno. Kompletan raspored prikazan je na slici 4.14.

OSOBA	PON	UTO	SRI	CET	PET	SUB	NED
srednja0	jutarnja	dnevna	nocna	slobodno	nocna	slobodno	jutarnja
	nocna	slobodno	nocna	slobodno	jutarnja	jutarnja	jutarnja
	jutarnja	slobodno	jutarnja	nocna	nocna	slobodno	dnevna
	dnevna	slobodno	dnevna	nocna	slobodno	jutarnja	dnevna
srednja1	dnevna	nocna	slobodno	jutarnja	jutarnja	jutarnja	nocna
	slobodno	jutarnja	nocna	slobodno	nocna	slobodno	jutarnja
	slobodno	jutarnja	dnevna	slobodno	jutarnja	jutarnja	jutarnja
	jutarnja	nocna	slobodno	jutarnja	jutarnja	jutarnja	nocna
srednja2	jutarnja	jutarnja	jutarnja	jutarnja	slobodno	jutarnja	jutarnja
	dnevna	dnevna	nocna	slobodno	nocna	nocna	slobodno
	nocna	slobodno	jutarnja	jutarnja	slobodno	jutarnja	dnevna
	slobodno	jutarnja	dnevna	dnevna	nocna	slobodno	nocna
srednja3	jutarnja	dnevna	dnevna	dnevna	dnevna	dnevna	slobodno
	jutarnja	slobodno	jutarnja	jutarnja	slobodno	dnevna	slobodno
	jutarnja	jutarnja	dnevna	dnevna	dnevna	slobodno	jutarnja
	slobodno	jutarnja	jutarnja	nocna	slobodno	slobodno	jutarnja
srednja4	jutarnja	dnevna	dnevna	dnevna	dnevna	nocna	slobodno
	dnevna	dnevna	slobodno	jutarnja	slobodno	jutarnja	jutarnja
	dnevna	dnevna	slobodno	nocna	slobodno	jutarnja	nocna
	slobodno	dnevna	dnevna	dnevna	dnevna	nocna	slobodno
srednja5	nocna	slobodno	dnevna	nocna	slobodno	jutarnja	jutarnja
	jutarnja	jutarnja	dnevna	dnevna	slobodno	nocna	slobodno
	dnevna	nocna	slobodno	dnevna	dnevna	nocna	slobodno
	dnevna	nocna	slobodno	dnevna	nocna	slobodno	jutarnja
srednja6	jutarnja	jutarnja	dnevna	dnevna	dnevna	dnevna	nocna
	slobodno	nocna	nocna	slobodno	dnevna	dnevna	slobodno
	jutarnja	dnevna	dnevna	dnevna	dnevna	dnevna	slobodno
	nocna	slobodno	jutarnja	jutarnja	jutarnja	jutarnja	jutarnja
srednja7	jutarnja	jutarnja	dnevna	dnevna	dnevna	nocna	slobodno
	slobodno	jutarnja	nocna	slobodno	nocna	slobodno	jutarnja
	jutarnja	dnevna	nocna	slobodno	dnevna	dnevna	dnevna
	nocna	slobodno	dnevna	dnevna	nocna	slobodno	slobodno
srednja8	slobodno	jutarnja	dnevna	nocna	slobodno	dnevna	nocna
	slobodno	jutarnja	jutarnja	dnevna	dnevna	dnevna	nocna
	nocna	slobodno	dnevna	dnevna	nocna	slobodno	jutarnja
	jutarnja	jutarnja	nocna	slobodno	jutarnja	nocna	slobodno
srednja9	dnevna	dnevna	dnevna	slobodno	jutarnja	dnevna	nocna
	slobodno	nocna	slobodno	dnevna	dnevna	dnevna	dnevna
	dnevna	dnevna	dnevna	nocna	slobodno	nocna	slobodno
	nocna	slobodno	nocna	slobodno	nocna	slobodno	dnevna
srednja10	jutarnja	jutarnja	nocna	nocna	slobodno	dnevna	dnevna
	dnevna	nocna	slobodno	jutarnja	jutarnja	dnevna	slobodno
	jutarnja	jutarnja	nocna	slobodno	dnevna	dnevna	dnevna
	dnevna	slobodno	nocna	slobodno	nocna	slobodno	nocna
srednja11	dnevna	slobodno	jutarnja	jutarnja	dnevna	slobodno	jutarnja
	jutarnja	dnevna	slobodno	jutarnja	slobodno	dnevna	nocna
	slobodno	jutarnja	nocna	slobodno	slobodno	jutarnja	jutarnja
	jutarnja	jutarnja	jutarnja	dnevna	dnevna	dnevna	dnevna
srednja12	dnevna	slobodno	jutarnja	jutarnja	dnevna	slobodno	jutarnja
	slobodno	jutarnja	jutarnja	dnevna	dnevna	nocna	slobodno
	dnevna	dnevna	slobodno	jutarnja	jutarnja	dnevna	dnevna
	slobodno	jutarnja	dnevna	slobodno	jutarnja	jutarnja	jutarnja
srednja13	nocna	slobodno	jutarnja	jutarnja	nocna	slobodno	slobodno
	jutarnja	jutarnja	dnevna	nocna	slobodno	jutarnja	dnevna
	nocna	slobodno	jutarnja	jutarnja	jutarnja	dnevna	slobodno
	jutarnja	jutarnja	nocna	nocna	slobodno	dnevna	nocna
srednja14	slobodno	nocna	slobodno	jutarnja	dnevna	dnevna	dnevna
	dnevna	dnevna	dnevna	slobodno	nocna	slobodno	slobodno
	jutarnja	jutarnja	jutarnja	dnevna	slobodno	jutarnja	jutarnja
	dnevna	dnevna	dnevna	dnevna	dnevna	dnevna	dnevna
visa15	jutarnja	dnevna	nocna	slobodno	dnevna	slobodno	jutarnja
	nocna	slobodno	slobodno	jutarnja	jutarnja	jutarnja	dnevna
	nocna	slobodno	nocna	slobodno	jutarnja	jutarnja	dnevna
	dnevna	slobodno	jutarnja	jutarnja	dnevna	dnevna	dnevna
visa16	nocna	slobodno	dnevna	dnevna	slobodno	jutarnja	dnevna
	slobodno	jutarnja	jutarnja	dnevna	dnevna	nocna	slobodno
	nocna	nocna	nocna	slobodno	dnevna	dnevna	slobodno
	jutarnja	dnevna	dnevna	slobodno	jutarnja	dnevna	dnevna
visa17	jutarnja	slobodno	jutarnja	jutarnja	jutarnja	dnevna	slobodno
	jutarnja	nocna	slobodno	dnevna	dnevna	dnevna	slobodno
	jutarnja	jutarnja	jutarnja	jutarnja	jutarnja	jutarnja	nocna
	slobodno	dnevna	dnevna	slobodno	dnevna	slobodno	jutarnja
visa18	nocna	slobodno	dnevna	slobodno	nocna	slobodno	jutarnja
	slobodno	jutarnja	jutarnja	nocna	slobodno	dnevna	dnevna
	dnevna	slobodno	jutarnja	jutarnja	jutarnja	dnevna	nocna
	slobodno	jutarnja	jutarnja	nocna	slobodno	jutarnja	dnevna
visa19	jutarnja	jutarnja	slobodno	jutarnja	jutarnja	jutarnja	slobodno
	dnevna	slobodno	jutarnja	slobodno	jutarnja	jutarnja	jutarnja
	jutarnja	jutarnja	dnevna	dnevna	dnevna	slobodno	jutarnja
	jutarnja	slobodno	jutarnja	jutarnja	dnevna	slobodno	jutarnja

Slika 4.13: Konačan raspored

5. Usporedba rezultata s rezultatima drugih radova

Nakon pronalaska rasporeda koji zadovoljava sva čvrsta i većinu lakih ograničenja te koji je vrlo blizu optimalnom, potrebno je taj raspored procijeniti nekom drugom metodom procjenjivanja koju koristi neki drugi istraživački rad. Kao kandidat je uzet rad [9] koji koristi [10]. Rad također koristi genetske algoritme za stvaranje rasporeda i također je implementiran u programskom jeziku Java.

5.1 Opis rada "A Brief Study of the Nurse Scheduling Problem (NSP)"

Rad [9] se također bavi problemom izrade rasporeda medicinskih sestara, međutim prilikom stvaranja rasporeda korištena su drukčija ograničenja, koja se ovdje neće razmatrati, te drukčija metoda procjene. Što se ograničenja tiče, potrebno je napomenuti da čvrsta ograničenja kod konačnog rasporeda ne smiju biti prekršena, a da laka ograničenja služe za optimizaciju. Funkcija koja procjenjuje vrijednost dobrote rasporeda opisana je izrazom 5.1.

$$\lambda * \mathcal{F}\left(\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{s=1}^{S}\mathcal{D}_{i,k,s}\right) + (1-\lambda) * \mathcal{G}\left(\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{s=1}^{S}\mathcal{D}_{i,k,s}\right) \tag{5.1}$$

Izraz 5.1 se može podijeliti na zbroj dviju funkcija gdje je svaka funkcija pomnožena s određenim faktorom λ, odnosno (1-λ). Faktor λ je broj iz intervala [0,1] i njime se određuje koja od funkcija izraza se više uzima u obzir. $\mathcal{D}_{i,k,s}$ je iz skupa {0,1} i označuje je li sestra *i* radi na dan *k* smjenu *s*. Funkcija $\mathcal{F}$ je linearna funkcija, a odnosi se na želje osoblja. Kao ulazni parametar se učitava matrica želja koja je dimenzija $(broj\ osoblja)\ x\ (broj\ dana * broj\ smjena)$. Brojevi u toj matrici su iz skupa {1, 2, 3, 4} gdje broj 1 "najviše odgovara", a broj 4 "najmanje odgovara". Umnožak matrice osobnih želja i rasporeda koji se procjenjuje se vraća kao rezultat funkcije $\mathcal{F}$. Osim matrice želja, kao ulazni parametar se učitava i matrica minimalnih potreba, to jest, matrica čije vrijednosti označuju koliko je

minimalno potrebno sestara za određeni dan i određenu smjenu. Matrica minimalnih potreba je dimenzija $(broj\ dana)\ x\ (broj\ smjena)$. Ta matrica se koristi u izrazu 5.1 prilikom računanja funkcije $\mathcal{G}$. $\mathcal{G}$ je eksponencijalna funkcija. Za svaku smjenu u svakom danu, rasporeda čija vrijednost dobrote se računa, se zbroji koliko sestara radi te se od tog broja oduzme broj koliko je minimalno potrebno sestara za tu smjenu na taj dan. Zatim se ta razlika kvadrira i rezultat se pamti. Nakon što se izračunaju rezultati za sve smjene u svim danima, oni se zbroje i šalju kao rezultat funkcije $\mathcal{G}$. Izraz 5.1 se koristi u metodi procjene i cilj je minimizirati njen rezultat. Na taj način se određuje koliko je raspored kvalitetan.

5.2 Prilagođavanje algoritma ovog rada

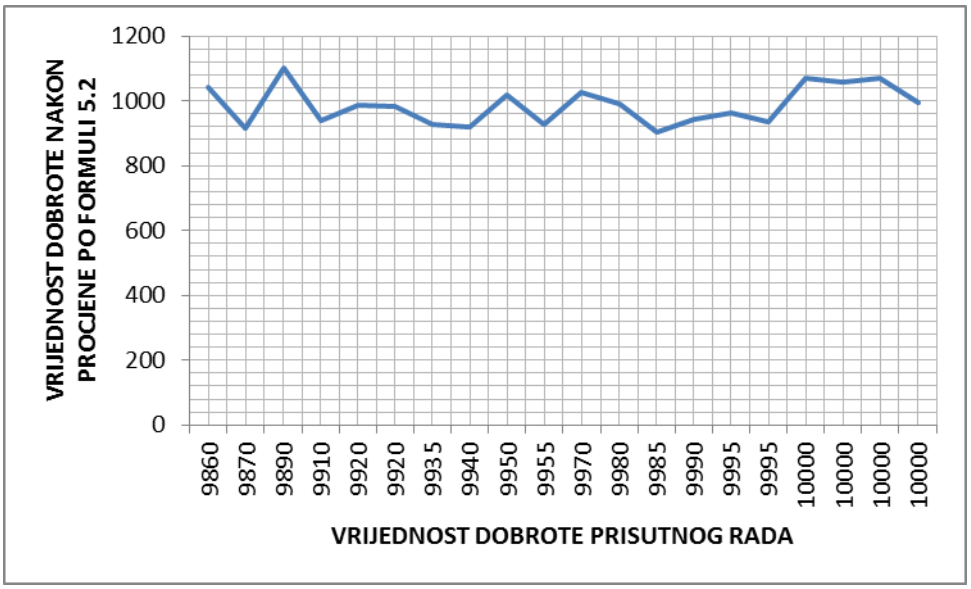
Zbog različitosti ovog rada i rada [9], potrebno je prilagoditi ovaj rad da bi se rezultati mogli usporediti. Metode procjene oba rada su međusobno različite, a kako ovaj rad ne uzima u obzir želje osoblja nego se fokusira na zadovoljavanje čvrstih i lakih ograničenja, i na taj način dolazi do rješenja, odlučeno je da se modificira metoda za procjenu rada [9] tako da se umjesto izraza 5.1 koristi modificirani izraz 5.2 koja je nastao tako da je vrijednost faktora λ jednaka nula.

$$\mathcal{G}\left(\sum_{i=1}^{\mathcal{I}}\sum_{k=1}^{\mathcal{K}}\sum_{s=1}^{\mathcal{S}}\mathcal{D}_{i,k,s}\right) \tag{5.2}$$

5.3 Rezultati

U radu [9] korištena su dva skupa podataka od kojih je jedan oblika $(\mathcal{I},\mathcal{K},\mathcal{S})$ = (25,7,4), a drugi oblika $(\mathcal{I},\mathcal{K},\mathcal{S})$ = (60,28,4). U tom radu se kao broj smjena uzima četiri jer se osim tri radne smjene broji i jedna slobodna smjena. Prilikom usporedbe rada [9] i ovog rada će se koristiti skup podataka $(\mathcal{I},\mathcal{K},\mathcal{S})$ = (25,7,4), odnosno veličina osoblja je 25, duljina intervala je sedam dana, a broj smjena je četiri. Vrijednost dobrote koja je dobivena u radu [9] je 214.50.

Nakon što se uz pomoć metode procjene ovog rada stvori, raspored se podvrgava procjeni po izrazu 5.2. Prosjek koji je dobiven na temelju dvadeset pokretanja programa ovog rada i zatim podvrgnut procjeni po izrazu 5.2 je 940.25. Minimalna vrijednost koja je dobivena je 905, a maksimalna 1100. U odnosu na vrijednost dobrote rada [9], ove vrijednosti su velike i ispada da se ne dobiva optimalan raspored. Međutim, u radu [9] se raspored stvarao na temelju različitih ograničenja i različite metode procjene nego kod ovog rada. Na slici 5.1 se može vidjeti da je po izrazu 3.1 dobiven idealan raspored (vrijednost dobrote = 10000), a po izrazu 5.2 vrijednost dobrote od 1057 što i nije toliko dobro.



Slika 5.1: Procjena dobrote prisutnog rada prema izrazu 5.2

5.4 Zaključak

Iz ovdje priloženih rezultata se može zaključiti da raspored stvoren na temelju jednih uvjeta ne mora nužno biti optimalan ako se procjenjuje na temelju nekih drugih uvjeta. Izraz 5.2 zbraja kvadrate razlika između broja osoblja koje radi određenu smjenu na određeni dan i minimalnog broja potrebnog za taj isti dan i istu smjenu. Iz toga se može zaključiti da će ukupni broj biti veći ako radi veći broj osoblja nego što je potrebno. Idealno bi bilo kad bi u svakoj smjeni radilo točno onoliko osoblja koliko je minimalno potrebno.

6. Zaključak

U ovom radu je prikazano da uporaba evolucijskih algoritama služi svrsi i da se uz pomoć njih u kratkom intervalu može stvoriti zadovoljavajući raspored radnih smjena koji obuhvaća veće vremenske periode i veći broj osoblja. Mjesta za napredak još uvijek ima. Mogu se dodati nova ograničenja ili izmijeniti postojeća. Može se promijeniti metoda procjene tako da se uzimaju u obzir i osobne želje kao što je to učinjeno u radu za usporedbu. Mogu se koristiti novi algoritmi, načini križanja, mutacije itd. Problem kao takav je globalan i zasigurno će se u budućnosti još razmatrati i pokušati će se primijeniti neki drugi algoritmi pretrage.

Prilikom izrade ovog rada i statistike, ako se izuzmu slučajevi u poglavlju 5 gdje je postavljen veliki broj osoblja, ni u jednom pokušaju nije stvoren idealan raspored. Uvijek je prekršeno barem jedno ograničenje, što budi želju za daljnjim usavršavanjem.

7. Literatura

[1] Vinko Bedek (2008), *Stvaranje rasporeda sati genetskim algoritmima*, Fakultet elektrotehnike i računarstva, http://www.zemris.fer.hr/~yeti/studenti/radovi/Bedek_Vinko.zip

[2] Daniel Domović (2012), *Vizualizacija rada algoritama zasnovanih na evolucijskom računanju,* Fakultet elektrotehnike i računarstva, http://www.zemris.fer.hr/~golub/ga/studenti/2012_domovic/diplomskiRad_Domovic.htm#_Toc327943775

[3] *Rješavanje problema usmjeravanja vozila genetskim algoritmom*, Fakultet elektrotehnike i računarstva, http://www.maturskiradovi.net/forum/attachment.php?aid=1716

[4] Wikipedia, http://en.wikipedia.org/wiki/Genetic_algorithm

[5] Aleksandar Prokopec, *Prilagodljivi genetski algoritam,* Fakultet elektrotehnike i računarstva, http://www.zemris.fer.hr/~golub/ga/studenti/prokopec/diplomski.htm

[6] Domagoj Jakobović (2007), *Genetski algoritmi – predavanje,* Fakultet elektrotehnike i računarstva, http://www.zemris.fer.hr/~yeti/studenti/GA%20predavanje.pdf

[7] Juraj Petrović (2007), *Evolucijski algoritmi,* Fakultet elektrotehnike i računarstva, http://www.zemris.fer.hr/~yeti/studenti/radovi/Petrovic_Juraj_Seminar.pdf

[8] Burke, Causmaecker, Berghe (2004), *The Handbook of Scheduling:* Algorithms, Models, and Performance Analysis: *Novel Metaheuristic Approaches to Nurse Rostering Problems in Belgian Hospitals*

[9] Augustine, Faer, Kavountzis, Patel (2009), *A Brief Study of the Nurse Scheduling Problem (NSP),* http://www.math.cmu.edu/~af1p/Teaching/OR2/Projects/P23/ORProject_Final_Copy.pdf

[10] Maenhout, Vanhoucke (2005), *NSPLib – A Nurse Scheduling Problem Library: A tool to evaluate (meta-)heuristic procedures,*

http://www.projectmanagement.ugent.be/sites/default/files/files/nsp/PaperNSPLib.pdf

[11] http://gp.zemris.fer.hr/ecf/algorithms.html