SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Marko Savić

Zagreb, 2013

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Dragutin Lisjak, dipl. ing.

Student:

Marko Savić

Zagreb, 2013.

Izjavljujem da sam završni rad na temu „Optimiranje genetičkim algoritmom“ izradio samostalno koristeći stečena znanja tijekom studija na Fakultetu strojarstva i brodogradnje te koristeći navedenu literaturu.

Ovom prilikom htio bih se prije svega zahvaliti mentoru doc.dr.sc. Dragutinu Lisjaku na utrošenom vremenu i trudu prilikom izrade ovog završnog rada.

Zahvaljujem se također i svima koji su mi omogućili uspješno studiranje i pomogli na bilo koji način, a posebno svojoj obitelji.

Marko Savić

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **MARKO SAVIĆ** Mat. br.:0035174247

Naslov rada na hrvatskom jeziku: **OPTIMIRANJE GENETIČKIM ALGORITMOM**

Naslov rada na engleskom jeziku: **OPTIMIZING USING GENETIC ALGORITHMS**

Opis zadatka: 1. Dati pregled metoda evolucijskih algoritama,
2. Detaljno opisati genetički (GA) algoritam,
3. Na konkretnom primjeru prikazati način rada GA-algoritma,
4. Zaključak.

Zadatak zadan:

16. studenog 2012.

Zadatak zadao:

Doc.dr.sc. Dragutin Lisjak

Rok predaje rada:
1. rok: 15. veljače 2013.
2. rok: 11. srpnja 2013.
3. rok: 13. rujna 2013.

Predviđeni datumi obrane:
1. rok: 27., 28. veljače i 1. ožujka 2013.
2. rok: 15., 16. i 17. srpnja 2013.
3. rok: 18., 19., i 20. rujna 2013.

Predsjednik Povjerenstva:

Prof. dr. sc. Zoran Kunica

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
a_p	mm	Dubina rezanja
b	mm	Širina odvojene čestice
f	mm	Posmak
N		Ukupni broj generacija gentskog algoritma
Ra	μm	Srednje aritmetičko odstupanje profila
Ry	μm	Najveća visina profila
Rz	μm	Visina neravnina profila mjerena u deset točaka
vc	m/s	Brzina rezanja

SAŽETAK

Genetski algoritmi predstavljaju dio evolucijskih algoritama koji služe za optimiranje raznih funkcija. Koriste se za optimiranje kompliciranijih matematičkih funkcija u slučajevima kada tražimo globalni minimun ili maksimum. U ovom radu prikazan je način funkcioniranja genetskih algoritama, kao i prikaz problema koji se rješavaju primjenom genetskih algoritama. Objašnjen je način dekodiranja varijabli na jezik koji računalo prepoznaje te je prikazano rješenje funkcije cilja u MatLab programskom paketu. Navedene se i opisane moguće postavke genetskog algoritma, te utjecaj operatora mutacije i križanja na efikasnost genetskog algoritma. Prikazan je način traženja minimuma i maksimuna zadane funkcije cilja.

Prikazani su grafovi za praćenje učinkovitosti genetskih algoritama za slučaj optimiranja Rastringove funkcije i za slučaj optimiranja hrapavosti za strojnu obradu.

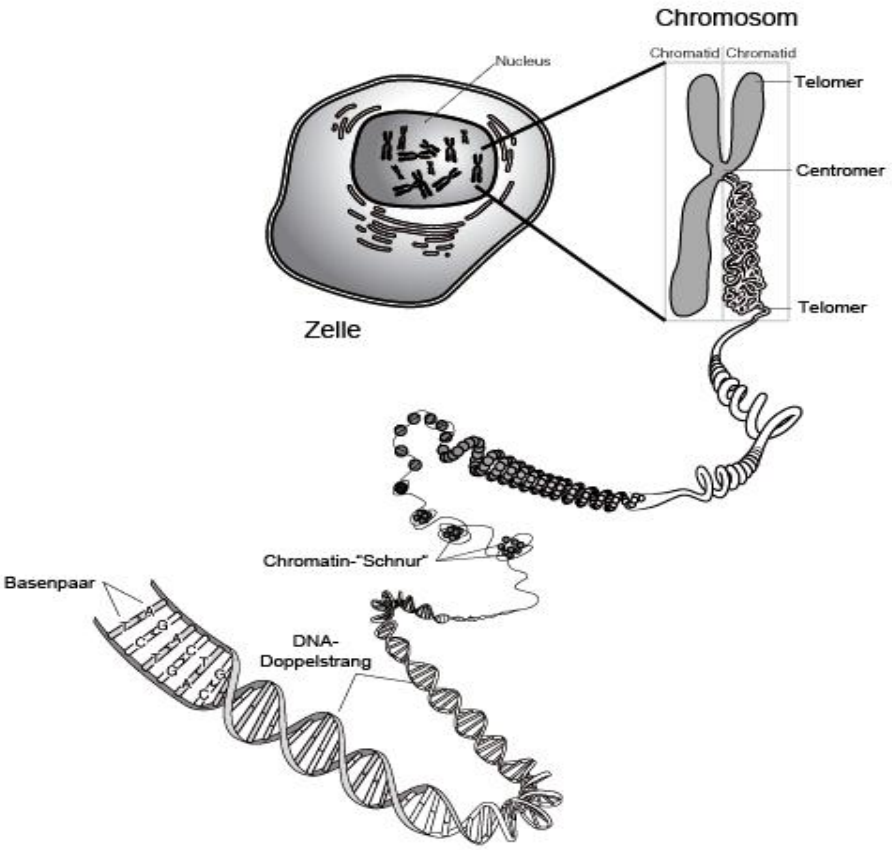
U drugom djelu rada optimira se funkcija hrapavosti pomoću genetskih algoritama, i to tako da se traže odgovarajući režimi rada u svrhu dobivanja minimalne hrapavosti za zadane uvjete. Rezultati optimizacije su dani grafički i u tablicama, i to za različite slučajeve. Za svaki slučaj je dana usporedba u kojoj se vidi učinkovitost genetskog algoritma s obzirom na zadani broj generacija.

1. Uvod

Inteligencija se može definirati kao sposobnost sustava da se prilagodi na okoliš koji se konstantno mijenja. Iz svakodnevnog iskustva očito je da je inteligentno ponašanje najlakše uočiti u ljudima. Evolucijski sustavi za opzimizaciju simuliraju evoluciju na računalima, sa ciljem stvaranja inteligentnijeg računala, odnosno onog koje će nam ponuditi najbolje rješenje našeg problema u najkraćem mogućem roku. Sličan način rješavanja može se primjeniti i kod problema optimizacije, gdje računalo pomoću niza iteracija nudi optimalno rješenje. Značajna razlika između evolucije u prirodi i one na računalima je u tome što za računalnu evoluciju nisu potrebni milijuni godina. Prirodnu evoluciju moguće je simulirati pomoću genetskih algoritama. Genetski algoritmi su spadaju u grupu evolucijskih algoritama. Ovakvi sustavi koriste se za optimizaciju raznih problema pomoću tehnika kombinacije. U grupu evolucijskih algoritama spadaju:

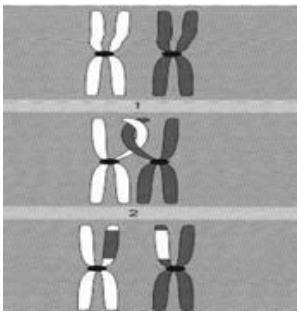
- Genetski algoritmi
- Evolucijske strategije
- Genetsko programiranje
- Evolucijsko programiranje
- Klasifikatorski sustavi sa sposobnošću učenja

Potreba za evolucijskim algoritama javila se nakon što su dobivene komplicirane funkcije koje je bilo moguće optimizirati samo pomoću višestrukog računanja funkcije. 1960. godine I. Rechenberg u svome djelu „Evolucijske strategije“ donosi ideju evolucijskog računanja. Ideja je prihvaćena od strane istraživača na području računalne znanosti, te se počinje intenzivno proučavati. Genetski algoritmi su rezultat istraživanja John-a Holland-a, u čemu su mu pomogli njegovi kolege i studenti. Kao algoritmi koji crpe ideju iz prirode, pogodni su za pretraživanje velikog prostora mogućnosti za najoptimalnijim rješenjem, tj. preživljavanjem najpogodnijih. Charles Darwin je ustanovio da je glavni pokretač evolucije prirodna selekcija. Priroda određuje uvjete za život - one jedinke koje imaju bolji genetski materijal prilagodit će se tim uvjetima i opstati, dok će slabije jedinke s vremenom izumrijeti. Možda najpoznatiji primjer vrste koja se nikako nije uspjela prilagoditi novim okolišnim uvjetima upravo su dinosauri koji su, kao što nam je poznato, izumrli. Kako jedna vrsta izumire, druga vrsta stvara nove potomke i na taj način broj živih bića uvijek ostaje barem približno konstantan.



Slika 1. Kromosom kao transportni oblik molekule DNA

Evolucijski algoritmi rade preko niza iteracija pomoću raznih operatetra, poput križanja i mutacije. Križanjem se nasljedna svojstva prenose sa roditeljske generacije na potomstvo. Pritom roditelji na svoju djecu prenose genetski materijal, tj. svoje dobre i loše osobine. Slučajna promjena nekih svojstava jedinke naziva se mutacija. U genetskom algoritmu mutacija je operacija kod koje se promjena gena vrši nad jednom jedinom jedinkom. Za razliku od mutacije, operacija križanja je operacija višeg reda jer se nove jedinke stvaraju kombinacijom gena više različitih jedinki iz roditeljske skupine.



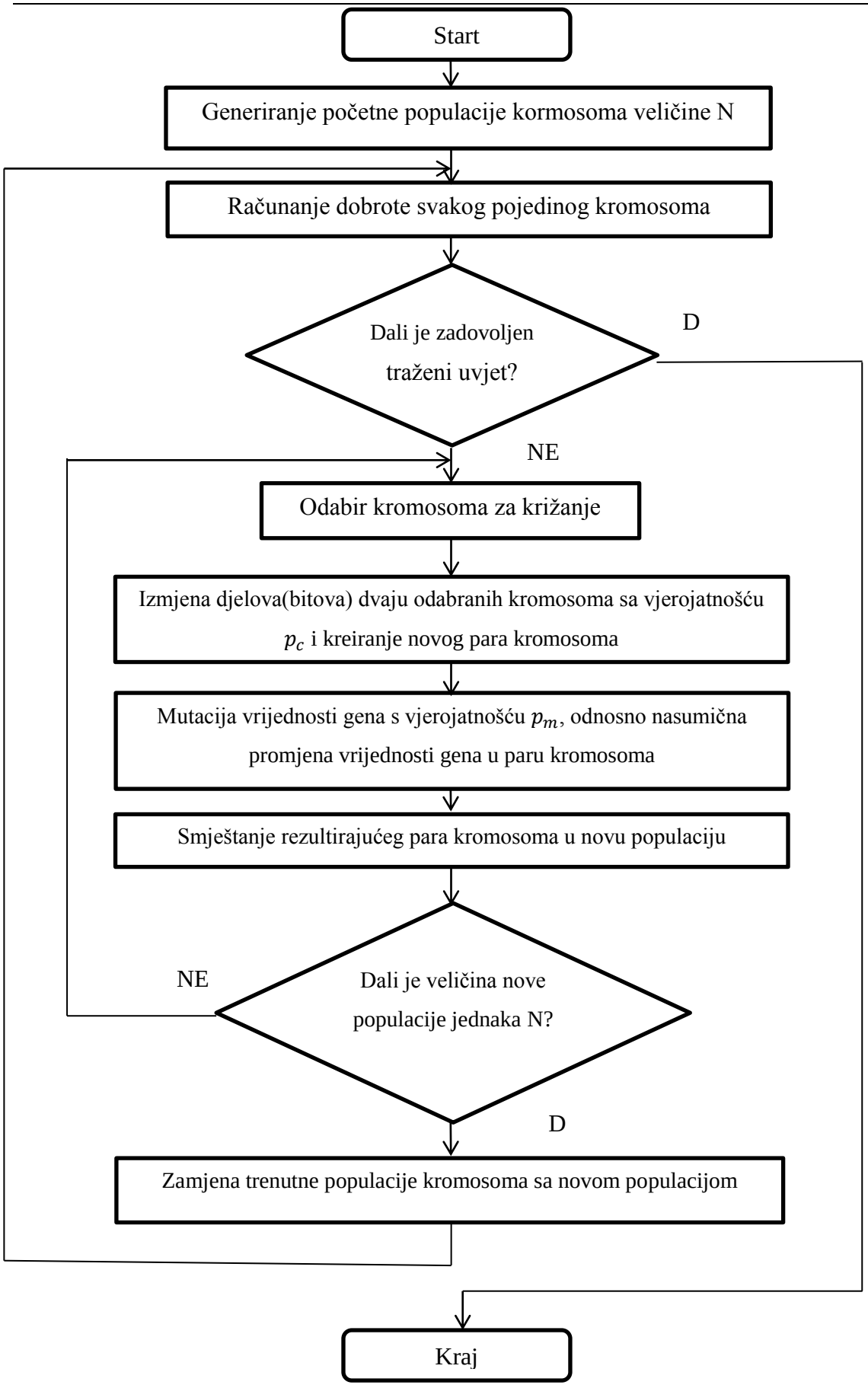
Slika 2. Operacija križanja

1.1 Genetski algoritmi

Genetski algoritmi su tip stohastičkog načina pretraživanja algoritma koji se bazira na biološkoj evoluciji. U slučaju kada imamo potpuno definiran problem, sasvim običan genetski algoritam nam može poslužiti za traženje optimalnog rješenja. Ovakav način rješavanja problema nam može poslužiti u raznim situacijama, kao naprimjer kod raspodjele snage kod više sustava za održavanje. Svi genetski algoritmi funkcioniraju na istom principu, koji se najjednostavnije može prikazati u sljedećim koracima:

1. Zadavanje problema: u ovom koraku treba se definirati početni problem, i to u obliku varijable kao kromosoma koji ima određenu duljinu, zatim treba definirati veličinu populacije N, te vjerojatnosti za mutaciju i križanje
2. Definiranje funkcije cilja: ovdje se definira funkcija cilja pomoću koje će se moći izmjeriti koliko dobro populacija odgovara zadanom problemu. Funkcija cilja je također i osnova za odabir kromosoma koji će sudjelovati u mutaciji i križanju prilikom sljedeće reprodukcije
3. U trećem koraku se nasumično generira početna populacija kromosoma
4. U četvrtom koraku se određuje dobrota svakog pojedinačnog kromosoma
5. U petom koraku se odabire prvi par kromosoma koji će se križati, i to iz početne populacije. Početni kromosomi se odabiru na temelju njihove dobrote.
6. Kreira se novi par kromosoma koji su nastali primjenom gentičkih operatora mutacije i križanja
7. Novi par kromosoma kreira novu populaciju, također početne veličine populacije N
8. Korak 5 se ponavlja sve dok se ne kreira dovoljno velika populacija
9. Početni kromosomi se zamjenjuju novima
10. Ponavlja se od koraka 4 sve dok nije zadovoljen kriterij koji smo postavili

Iz ovih se koraka vidi kako je ovdje riječ o iterativnom procesu koji se može prikazati jednostavnim dijagramom toka. Svaka pojedina iteracija predstavlja generaciju. Očekuje se da je broj generacija za jednostavan genetski algoritam između 50 i 500, zavisno o postavljenom problemu. Nakon izvršenja dovoljnog broja iteracija, očekuje se da dobijemo „najkvalitetnije“ kromosome, odnosno optimalno rješenje.



Slika 3. Dijagram toka genetskog algoritma

Genetski algoritam je recept koji kazuje što treba raditi s genetskim materijalom kako bi se s određenom vjerojatnošću nakon određenog vremena postiglo zadovoljavajuće rješenje zadanog optimizacijskog problema Genetski materijal je skup svojstava koji opisuju neku jedinku. Genetski algoritam ne određuje na koji način je genetski materijal pohranjen u radni spremnik, niti kako treba manipulirati genetskim materijalom, vec samo kaže da se genetski materijal treba razmjenjivati, slučajno mijenjati, a bolje jedinke trebaju s većom vjerojatnošću preživljavati selekciju. Kako će se bolje jedinke selektirati za reprodukciju, te kako će se i s kolikim vjerojatnostima obaviti križanje i mutacija, određuje se na temelju iskustva i eksperimentalno, tj. heuristički. Stoga postoji velika sloboda u izradi genetskih algoritama. Cijena te slobode su loši ili nikakvi rezultati koji se najčešće dobivaju s genetskim algoritmom koji nema podešene parametre ili nema genetske operatore prilagođene problemu. Podešavanjem (optimiranjem) parametara genetskog algoritma mogu se postići zadovoljavajući rezultati. Međutim, sam postupak podešavanja je dugotrajan proces. Za postizanje zadovoljavajućih rješenja genetskim algoritmima potrebno je računalo velike procesorske snage. Zato se za rješavanje problema optimizacije koriste i umrežena računala kako bi se postiglo brže vrijeme riješavanja problema. Određivanje funkcija cilja i dobrote je redovito kompliciran i zahtjevan zadatak koji se rješava eksperimentalno. Za ovakve probleme potrebno je veliko vrijeme, u kompliciranijim slučajevima nije dovoljno i preko 100 sati eksperimentiranja kako bi se pronašla tražena funkcija cilja. Drugi veliki problem je postavljanje postavaka genetskih algoritama. Kod kompliciranijih problema potrebno je utvrditi najpovoljnije postavke za operacije križanja i mutacije. Kod takvih problema uvodi se pojam paralelnih genetskih algoritama koji omogućavaju paraleno odvijanje više operacija na odvojenim računalima, a sve u svrhu smanjenja vremena potrebnog za izvršenje genetskog algoritma. Genetski algoritam koji ima ispravno podešene parametre za optimiranje je vrlo efikasan način rješavanja problema. U slučaju ispravno postavljenog problema i pravilno podešenog genetskog algoritma moguće je dobiti vrlo točan skup rješenja. Zbog ovoga je vrlo raširena primjena genetskih algoritama. Genetski algoritmi mogu vrlo efikasno i brzo riješiti problem optimiranja ukoliko su zadovoljeni svi uvjeti za brzo izvođenje genetskog algoritma.

1.2 Struktura genetskog algoritma

Genetski algoritmi obavljaju posao koji se najjednostavnije može opisati na način da genetski algoritam vrši selekciju boljih kromosoma od kojih onda stvara nove. Početni kromosomi odabiru se nasumično iz početne populacije koja je određene veličine za svaki problem. Reprodukcija novih kromosoma traje sve dok se postigne zadovoljavajuća vrijednost funkcije cilja. Reprodukcija se odvija pomoću genetskih operatora mutacije i križanja. Križanje prenosi svojstva roditelja na djecu, a mutacija slucajno mijenja svojstva jedinke. Genetski algoritam ne specificira kako se križanjem prenose svojstva roditelja na djecu, niti kako se slučajno mijenjaju svojstva jedinki, niti kako se selektiraju bolje jedinke za reprodukciju i način na koji se generira početna populacija. Upravo je ta sloboda u odabiru vrste križanja, mutacije, selekcije i inicijalizacije otežavajuća okolnost u procesu izgradnje genetskog algoritma za rješavanje specifičnog optimizacijskog problema. Naime, pokazalo se da ne postoji takav skup genetskih operatora za koji bi genetski algoritam, ako se primijeni za rješavanje proizvoljnog skupa optimizacijskih problema, davao superiorne rezultate u odnosu na genetski algoritam s nekim drugim operatorima.

```
Genetski_algoritam(){
     generiraj_pocetnu_populaciju();
     dok
     (nije_zadovoljen_uvjet_završetka_evolucijskog_procesa){
     selektiraj_bolje_jedinke_za_reprodukciju();
     reprodukcijom_generiraj_novu_populaciju();
}
}
```

Slika 4. Struktura genetskog algoritma[3]

Primjena genetskih algoritama je moguća u slučajevima kada problem možemo opisati kao pretraživanje ili optimizaciju proizvoljnih podataka, te poznajemo način mogućeg mjerenja uspješnosti svakog pojedinog rješenja. Iako genetski algoritam pretražuje i optimizira zadani skup, rješenje ne predstavlja uvijek optimum procesa, te je potrebno pažljivo konstruirati okruženje evolucije kako bi se zaista dobili rezultati bliski rješenju. Prvi korak u konstrukciji ispravnog evolucijskog okruženja je razumijevanje svih njegovih dijelova (populacija, dobrota, selekcija, operatori) te njihove interakcija.

1.2.1 Populacija

Populacija je skup jedinki iste vrste smještenih na nekom području. Kako dio populacije stari i umire, tako se razmnožavanjem stvaraju novi potomci i veličina populacije u svakoj generaciji ostaje konstantna. U genetskom algoritmu populacija predstavlja skup jedinki od kojih je svaka jedinka potencijalno rješenje zadanog problema. Početna populacija može biti odabrana slučajnim odabirom ili nekim drugim optimizacijskim postupkom. Odumiranje slabijih jedinki koje se nisu uspjele prilagoditi novim životnim uvjetima i opstanak jedinki koje su to uspjele u genetskim se algoritmima provodi uporabom takozvane funkcije dobrote. Razmnožavanje jedinki koje su se svojim dobrim svojstvima uspjele probiti provodi se operacijom križanja.

Iz iteracije u iteraciju jedinke u populaciji poprimaju sve poželjnija svojstva. Algoritam obično završava dosezanjem zadanog broja iteracija. Kada je uvjet završetka ispunjen, iz dobivene populacije odabire se najbolja jedinka i ona predstavlja rješenje optimizacijskog problema.

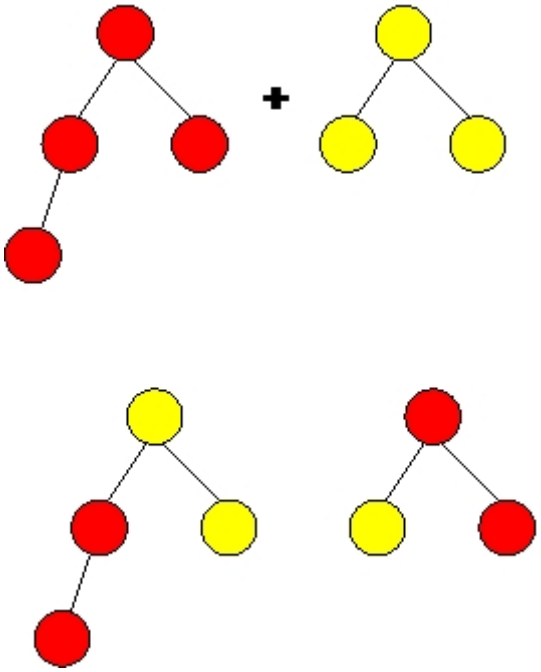
1.2.2 Operator križanja

Operator križanja radi na principu slučajnog odabira. Ova operacija započinje nasumičnim odabirom točke križanja gdje se dva početna kromosoma susreću. Nakon te točke se djelovi kromosoma izmjenjuju. Rezultat su dva nova kromosoma koji su nastali od dva početna kromosoma. U ovom primjeru prosječna kvaliteta kromosoma se povečala sa 36 na 42.

Tablica 1. Princip rada operatora križanja

Operator križanja			
X6	10-01	*X6+1*	10 00
X2	01-00	*X2+1*	01 01

U tabeli je prikazan princip rada križanja. Par kromosoma X2 i X6 je odabran za križanje, nakon toga je nasumično odabrana točka nakon koje se izmjenjuju geni kromosoma, u ovom slučaju nakon drugog gena. Zatim se zamjenjuju geni i nastaju kromosomi X_{6+1} i X_{2+1}. To su kromosomi nakon prve operacije križanja.



Slika 5. Operacija križanja

1.2.3 Operator mutacije

Mutacija, koja kao takva rijetko nastupa u prirodi, predstavlja promjenu jednog gena u kromosomu. Ovakavim operaterom može se postići značajno povećanje prosječne kvalitete kromosoma, ali može doći i do štetnih efekata. Mutacija se koristi kao jamstvo da algoritam ne bude zatočen na nekom lokalnom optimumu. Odabir i proces operacije križanja može stagnirati na nekom lokalnom optimumu. Pod takvim uvjetima, može se dogoditi da se prosječna kvaliteta kromosoma neće mijenjati niti povećavati kroz niz generacija. Ovakvo riješenje nam se poslije može činiti kao optimalno, ili kao lokalno optimalno, zbog toga što nema promjene prosječne kvalitete kromosoma i to zbog toga što algoritam pretraživanja ne može nastaviti. Mutaciju možemo tako protumačiti kao jamstvo protiv gubljenja genetičke različitosti.

Slika 6. Operacija mutacije

Operator mutacije radi tako da nasumično promjeni vrijednost gena u nekom kromosomu. Tako, naprimjer, u kromosomu X1 može mutirati njegov drugi gen. Mutacija se može

dogoditi u bilo kojem genu bilo kojeg kromosoma, s određenom vjerojatnošću. Ova vjerojatnost obično iznosi između 0,001 i 0,01 i ovisi o problemu kojeg optimiziramo. Genetski algoritmi s ovakvim postavkama daju optimalna riješenja, i to obično nakon nekoliko stotina generacija (iteracija).

2. Primjer rješavanja funkcije genetskim algoritmom

Genetski algoritmi koriste stohastičke metode pretraživanja, što znači da bi dobrota više populacija u nizu mogla ostati približno ista kroz niz generacija. Nakon nekog vremena izvjesno je da će se pojaviti generacija superiorne dobrote. Ovaj način rada genetskih algoritama stvara problem prlikom definiranja funkcije cilja. Uobičajena praksa prilikom primjene genetskih algoritama je da se nakon određenog broja generacija završi rad genetskog algoritma. Nakon toga se proučavaju najbolji kromosomi u populaciji. U slučaju da nije dobiveno zadovoljavajuće rješenje, genetski algoritam se ponovno pokreće. Kako bi bolje razumjeli rad genetskih algoritama, možemo uzeti jednostavnu funkciju $f(x) = 15x - x^2$. U ovoj funkciji neka varijabla x varira između 0 i 15, a cilj je dobiti maksimalnu vrijednost funkcije. Radi jednostavnosti, varijabla x može poprimiti samo cijelobrojne vrijednosti. Zbog toga što varijabla x može poprimiti samo vrijednosti između 0 i 15, možemo ih zapisati u binarnom obliku, i to na taj način da će svaki bit predstavljati jedan gen kromosoma. U sljedećoj tabeli dan je binarni oblik varijable x.

Tablica 2. Binarni oblik varijable x

Cijelobrojni oblik	Binarni kod	Cijelobrojni oblik	Binarni kod	Cijelobrojni oblik	Binarni kod
1	0001	6	0110	11	1011
2	0010	7	0111	12	1100
3	0011	8	1000	13	1101
4	0100	9	1001	14	1110
5	0101	10	1010	15	1111

Neka veličina populacije N bude jednaka 6, vjerojatnost križanja $p_c = 0{,}7$, a vjerojatnost mutacije $p_m = 0{,}001$ (odabrane vrijednosti su uobičanje kod genetskih algoritama). Funkcija cilja je definirana

$$\boldsymbol{f(x) = 15x - x^2} \qquad \textbf{(1)}$$

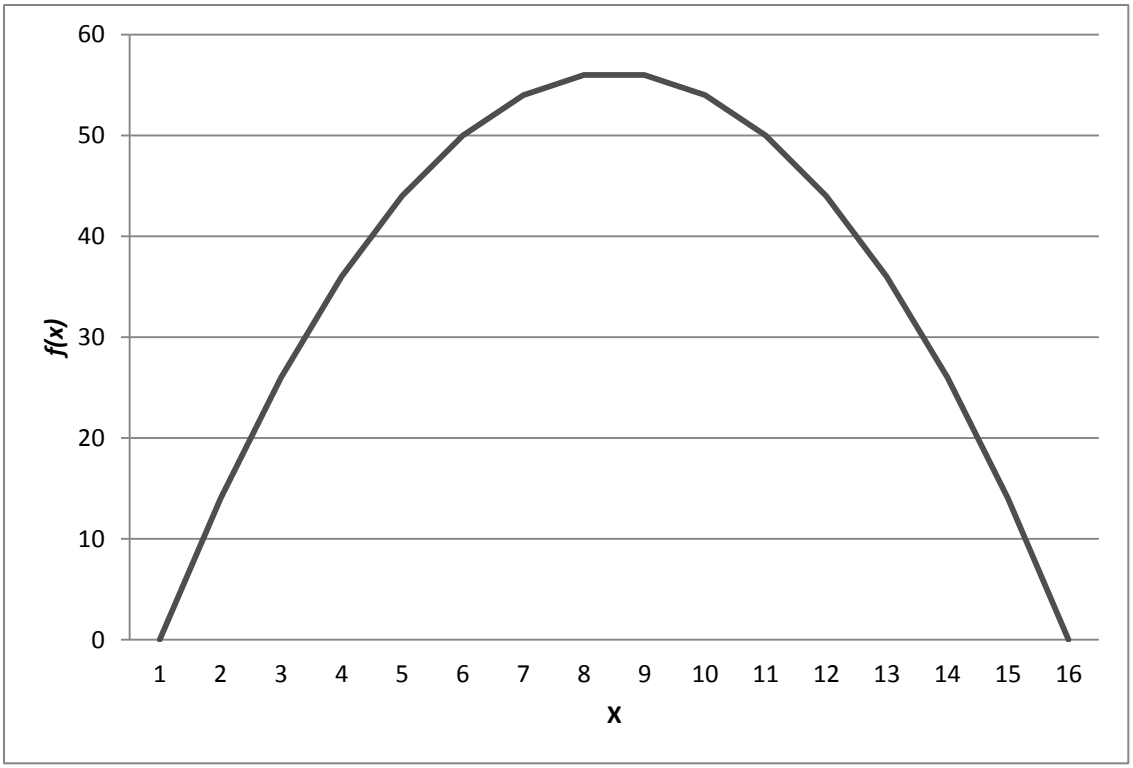
Genetski algoritam zatim kreira početnu populaciju kromosoma tako da popunjava šest 4-bitnih nizova, i to tako da nasumično popuni kromosom jednica i nulama. Početna

populacija može izgledati kao u sljedećoj tabeli. Problem u praksi može imati i preko tisuću generiranih kromosoma.

Tablica 3. Početna populacija genetskog algoritma

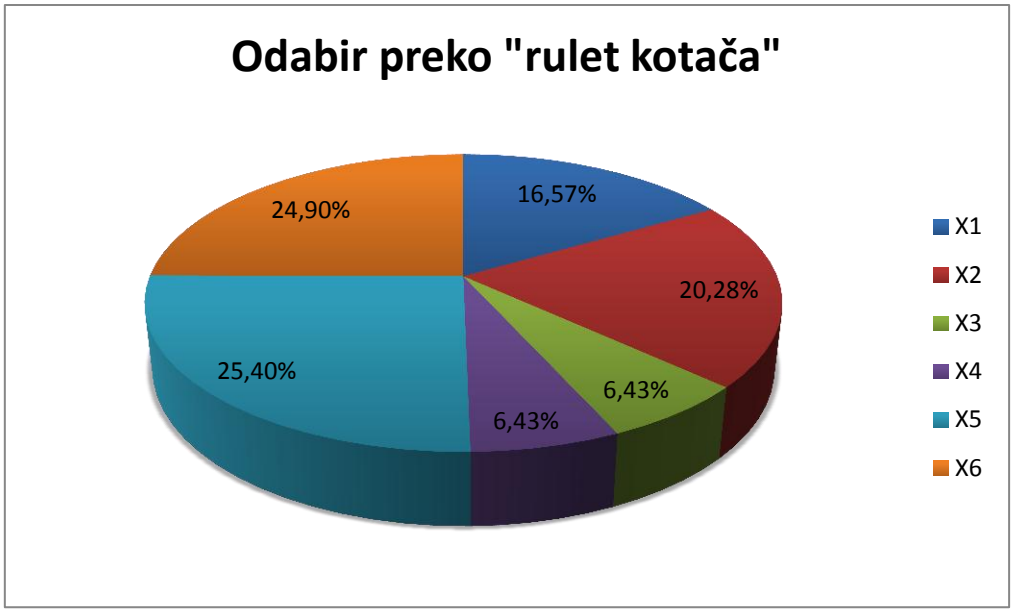
Oznaka kromosoma	Binarni niz kromosoma	Dekodirani kromosom	Dobrota kromosoma	Mjera dobrote kromosoma, %
X1	1100	12	36	16,5
X2	0100	4	44	20,2
X3	0001	1	14	6,4
X4	1110	14	14	6,4
X5	0111	7	56	25,7
X6	1001	9	54	24,8

Idući korak uključuje računanje dobrote svakog pojedinog generiranog kromosoma. Ovaj postupak obavlja genetski algoritam. Pretvaranje tih rezultata u postotke će kasnije poslužiti za generiranje grafaičkog prikaza udjela svakog pojedinog kromosoma. Najbolji kromosomi će zbog toga imati najviše šanse za obavljanje operacija križanja i mutacije. Sličan proces se odvija i u prirodi. Prilikom prirodne selekcije, može se reći kako će samo najbolje vrste preživjeti, a zatim i dalje prenositi svoje gene na potomke. Genetski algoritmi koriste sličan pristup, ali s tom razlikom da kod genetskih algoritama veličina populacije kromosoma ostaje nepromijenjena prilikom prijelaza sa jedne generacije na drugu. Prikazan je grafički prikaz funkcije *f(x)*.



Slika 7. Graf funkcije *f(x)*

Sa grafa se vidi da je maksimalna vrijednost funkcije između 7 i 9, što samo potvrđuje već prije računatu „kvalitetu“ kromosoma. Sa prethodne tablice se može također i očitati mjera dobrote svakog kromosoma, i to u postotcima. Ovaj broj nam pokazuje kolike su šanse da svaki pojedini kromosom bude odaberen za operatere mutacije i križanja. Tako se vidi da najveće šanse za odabir imaju kromosomi X5 i X6, dok kromosomi X3 i X4 imaju najmanju šansu za odabir. Ovaj odabir je nasumičan te se koriste razne tehnike i metode odabira kromosoma za mutaciju i križanje. Jedan od popularnijih načina odabira je preko „rulet kotača“. Ovaj način ilustriran je na sljedećem grafu.



Slika 8. Princip odabira preko „rulet kotača“

Kako se vidi sa grafa, svaki pojedini kromosom je zauzeo odeđen dio kotača. Područje koje zauzima svaki kromosom unutar kotača je jednako mjeri kvalitete svakog pojedinog kromosoma. Tako se vidi kako su najkvalitetniji kromosomi X5 i X6 (oni zauzimaju najveću površinu na grafu), dok kromosomi X3 i X4 imaju najmanju kvalitetu (zauzimaju najmanju površinu na grafu). Kromosomi za križanje i mutaciju se biraju tako da se generira nasumičan broj između 0 i 100, a zatim je odabran kromosom na čijem se području nalazi generirani broj. U ovom primjeru imamo populaciju od 6 kromosoma, što znači da će se ovaj proces ponavljati šest puta kako bi postigli šest kromosoma za iduću generaciju. Tako naprimjer, možemo dobiti kromosome X6 i X2 za „roditeljske“ kromosome, odnosno one kromosome koji će biti podvrgnuti operacijama križanja i mutacije, a sve u cilju dobivanja veće prosječne kvalitete kromosoma.

2.1 Primjer riješavanja funkcije s dvije varijable

Prethodni primjer bio je jednostvan za prikazati i izračunati. Ali u slučaju da imamo problem u kojem su prisutne dvije ili više varijabli, situacija se komplicira. U ovom primjeru tražimo maksimum funkcije s dvije varijable.

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3)e^{-x^2-y^2} \qquad \textbf{(2)}$$

U ovoj funkciji neka varijable x i y mogu poprimiti vrijednosti između -3 i 3.

Prvi problem se javlja u prikazivanju varijabla kao kromosoma. Treba prikazati parametre *x* i *y* u obliku ulančanog binarnog niza u kojem je svaki parametar predstavljen pomoću osam binarnih bitova. Zatim je potrebno odrediti veličinu početne populacije i nasumično generirati početnu populaciju. Nakon toga je potrebno odrediti kvalitetu svakog pojedinog kromosoma. Ovaj korak se sastoji od dvije operacije. Prvo se kromosom dekodira u realnu vrijednost (parametre *x* i *y*) u intervalu između -3 i 3. Zatim se dekodirane vrijednosti varijabla *x* i *y* zamjenjuju u vrhove funkcija.

2.1.1 Dekodiranje varijabli

Dekodiranje je operacija poću koje će se u ovom primjeru kromosom, koji je niz od 16 bitova, pretvoriti u realnu vrijednost. Najprije se niz od 16 bitova podjeli u dva 8-bitna niza.

Tablica 4. 16 bitni niz podjeljen u 2 8-bitna

X	*Y*
10001010	00111011

Ova dva niza se zatim pretvaraju iz binarnog oblika (baza je 2) u decimalni (baza je 10).

$$(10001010)_2 = 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = (138)_{10}$$

$$(00111011)_2 = 0 * 2^7 + 0 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0 = (59)_{10}$$

Nakon dekodiranja potrebno je raspon brojeva koje je moguće poprimiti binarnim oblikom $(0 \div 2^8 - 1)$ pretvoriti u raspon brojeva koje mogu poprimiti varijable x i y (od -3 do 3).

$$\frac{6}{256 - 1} = 0{,}0235294$$

Kako bi dobili prave vrijednosti varijabla x i y potrebno je pomnožiti njihove decimalne vrijednosti sa 0,0235294 i oduzeti 3.

$$x = (138)_{10} * 0{,}0235294 - 3 = 0{,}2470588$$

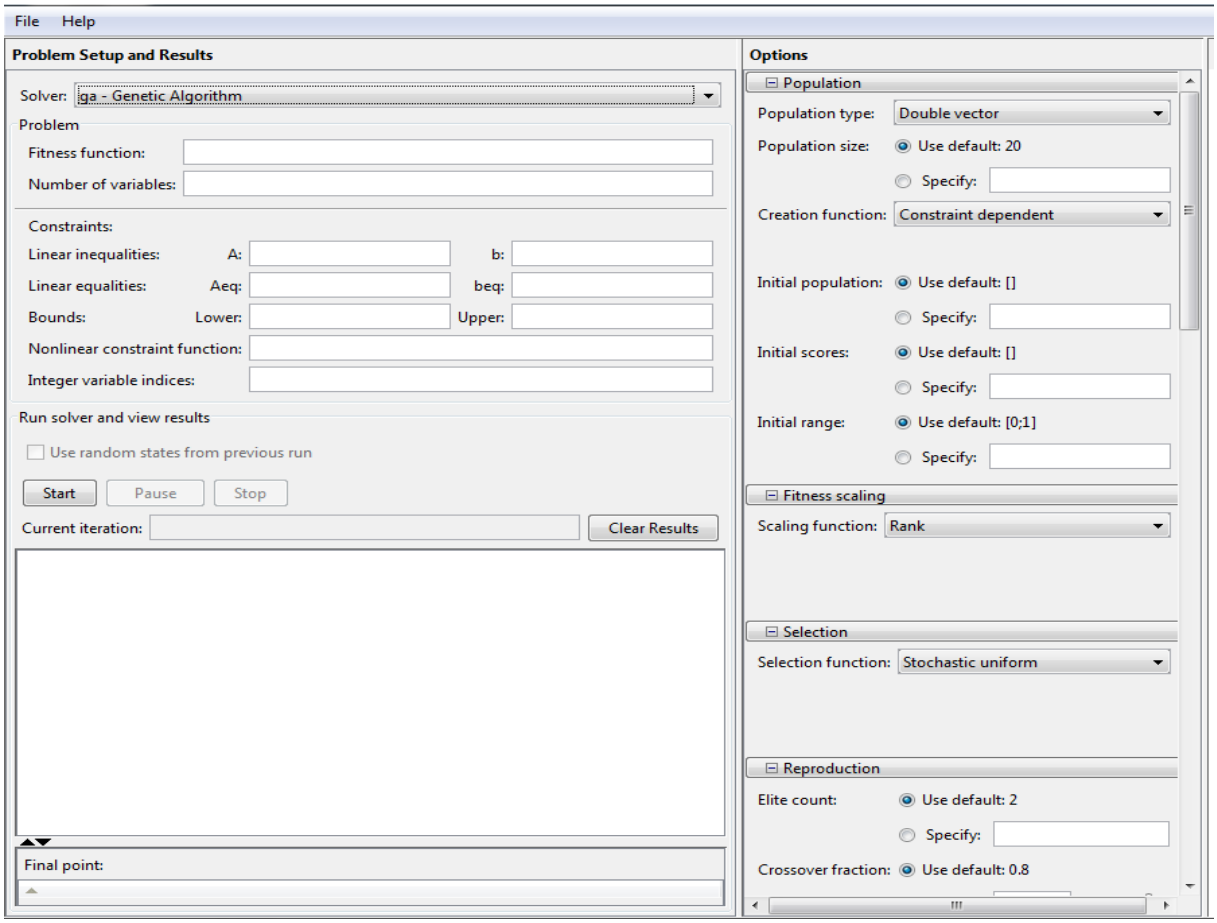
$$y = (59)_{10} * 0{,}0235294 - 3 = -1{,}6117647$$

Ukoliko je potrebno, mogu se primjeniti i druge metode dekodiranja.

Genetski algoritam sada uzima dekodirane vrijednosti varijabla *x* i *y* te računa vrijednosti kvalitete kromosoma. Ovaj proces može se i računati pomoću programa MatLab koji omogućava korištenje genetskih algoritama za rješavanje kompliciranih matematičkih funkcija.

3. Prikaz riješenja funkcije pomoću MatLab programskog paketa

MatLab programski alat omogućava riješavanje ovakvih, i kompliciranijih funkcija, uz pomoć alata za optimizaciju. MatLab nudi mnogo mogućnosti riješavanja matematičkih funkcija uz velike mogućnosti odabira ograničenja genetskih algoritama. Kako bi riješili zadani primjer u MatLab-u, potrebno je pozvati aplikaciju za optimizaciju.

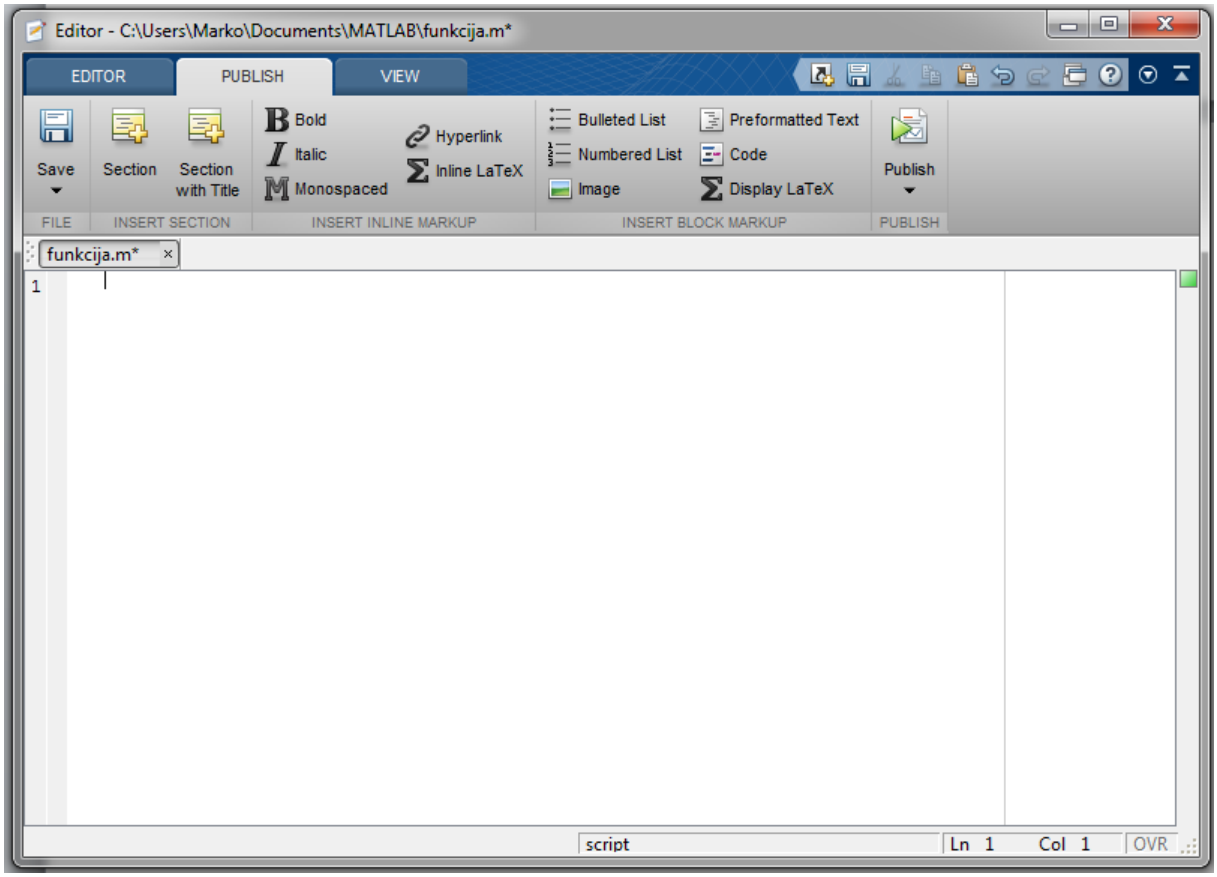


Slika 9. Početne postavke genetskog algortima

U prozoru za optimizaciju možemo odabrati način rješavanja matematičke funkcije. Ovdje odabiremo Ga-Genetic Algorithm. Sljedeće treba definirati funkciju cilja (fitness function) te broj nezavisnih varijabli u funkciji cilja. Ovim alatom je moguće riješavati razne probleme ukoliko je poznata funkcija cilja. Prilikom riješavanja matematičke funkcije genetskim algoritmom treba još i defenirati broj generacija (iteracija), veličinu populacije, raspon, postavke za definiranje vjerojatnosti operacija križanja i mutacije.

3.1 Pisanje funkcije cilja u MatLab-u

Funkciju cilja u MatLab-u treba napisati u obliku funkcija.m kako bi se mogla poslije učitati u genetski algoritam. Funkcija cilja se piše pomoću MatLab-ova Script alata. Ukoliko funkcija ima više od dvije varijable, potrebno je definirati vektorski oblik neke varijable, npr. varijable *x*. Vektor *x* zatim definiramo kao vektor koji je duljine odgovarajućeg broja varijabli, npr. 2. Za vektor *x* sada su definirane komponente vektora *x(1)* i *x(2*).



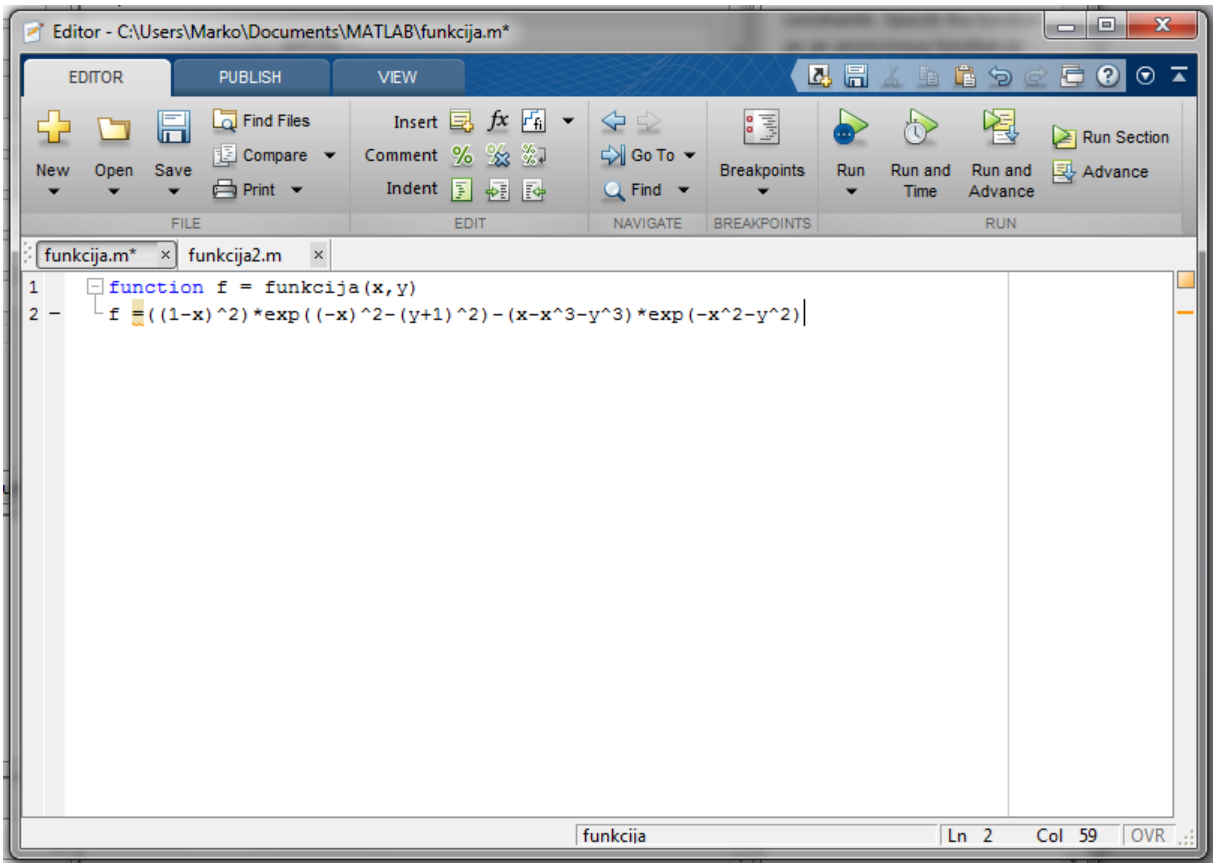
Slika 10. Prozor za pisanje funkcija u MatLab programskom alatu

U ovom prozoru potrebno je definirati funkciju cilja u formatu koji je razumljiv MatLab-u. Funkcija cilja glasi

$$f(x,y) = (1-x)^2 e^{-x^2-(y+1)^2} - (x - x^3 - y^3)e^{-x^2-y^2} \quad \textbf{(3)}$$

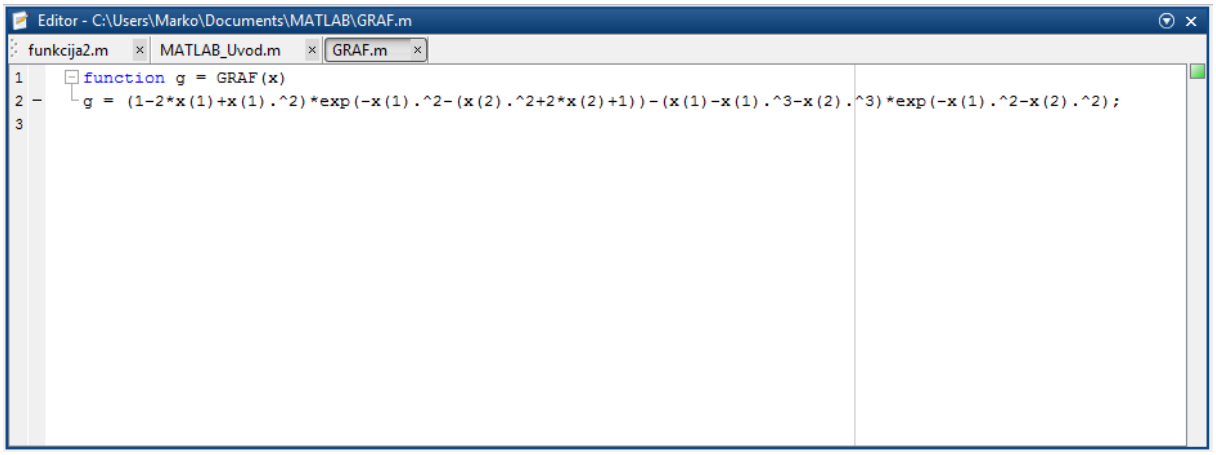
Pisana u formatu za MatLab funkcija izgleda:

```
f =((1-x)^2)*exp((-x)^2-(y+1)^2)-(x-x^3-y^3)*exp(-x^2-y^2)
```
(4)



Slika 11. Funkcija u MatLab programskom alatu

Funkcija pisana u vektorskom obliku



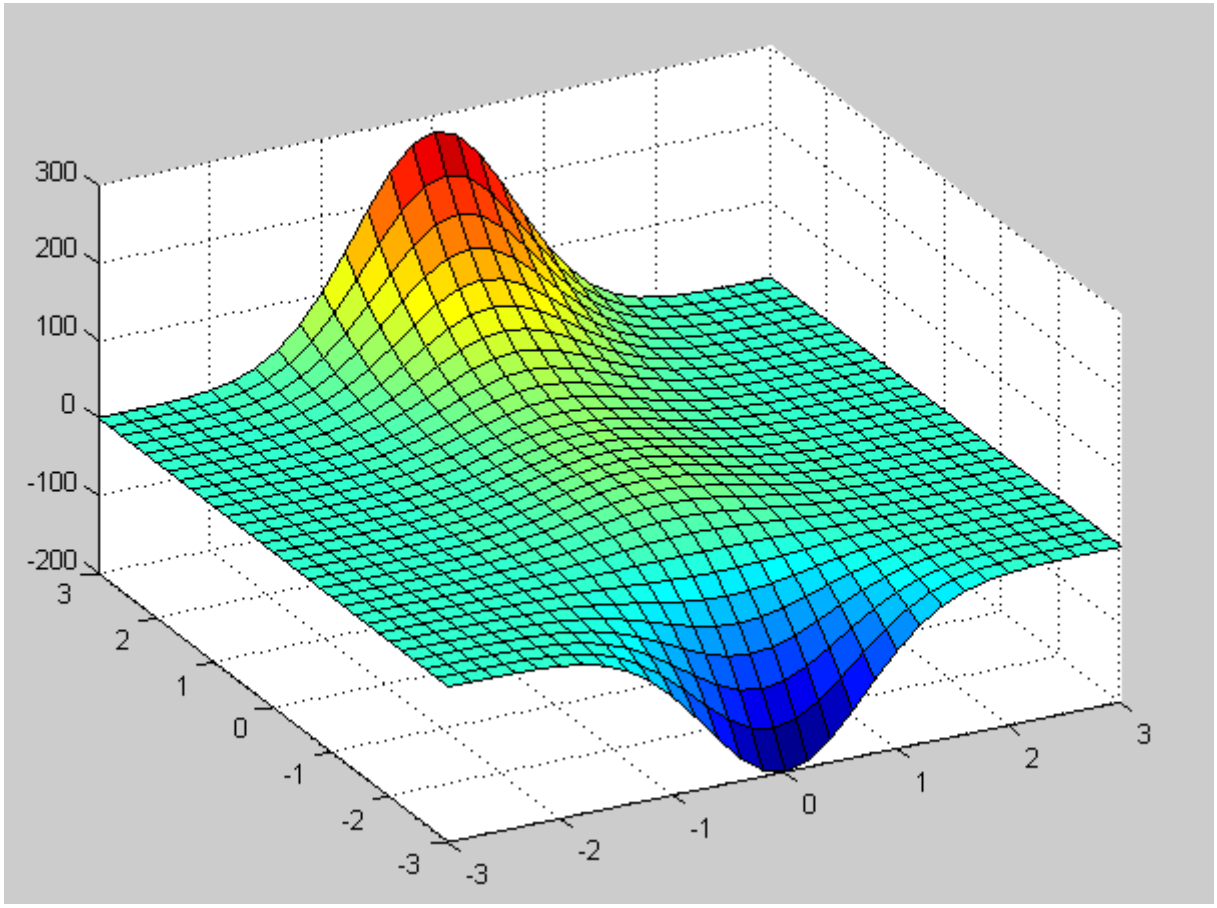
Slika 12. Funkcija u vektorskom obliku

```
g = (1-2*x(1)+x(1).^2)*exp(-x(1).^2-(x(2).^2+2*x(2)+1))-(x(1)-
    x(1).^3-x(2).^3)*exp(-x(1).^2-x(2).^2);
```
(5)

3.2 Definicija parametara optimizacije

MatLab nam pruža mogućnost definiranja svih potrebnih ograničenja koja su nam potrebna za riješavanje problema optimizacije. Tako prilikom definiranja postavki genetskih algoritam trebamo odrediti razne parametre. Parametri uključuju: linearne nejednakosti, linearne jednakosti, gornje i donje granice, veličinu populacije, početnu populaciju, načine odabira kromosoma za reprodukciju, vjerojatnosti za operacije križanja i mutacije, te na kraju kriterij prekida genetičkog algoritma.Prije definiranja tih postavki napraviti čemo graf funkcije za našu funkciju, i to u granicama koje su nam potrebne, u ovome slučaju od -3 do 3. Za to će nam poslužiti naredbe meshgrid i surf. Pomoću naredbe meshgrid pripremamo radnu površinu za grafički prikaz funkcije u x,y,z ravnini. Naredbom surf zatim pozivamo stvaranje grafa funkcije u 3-d okolišu.

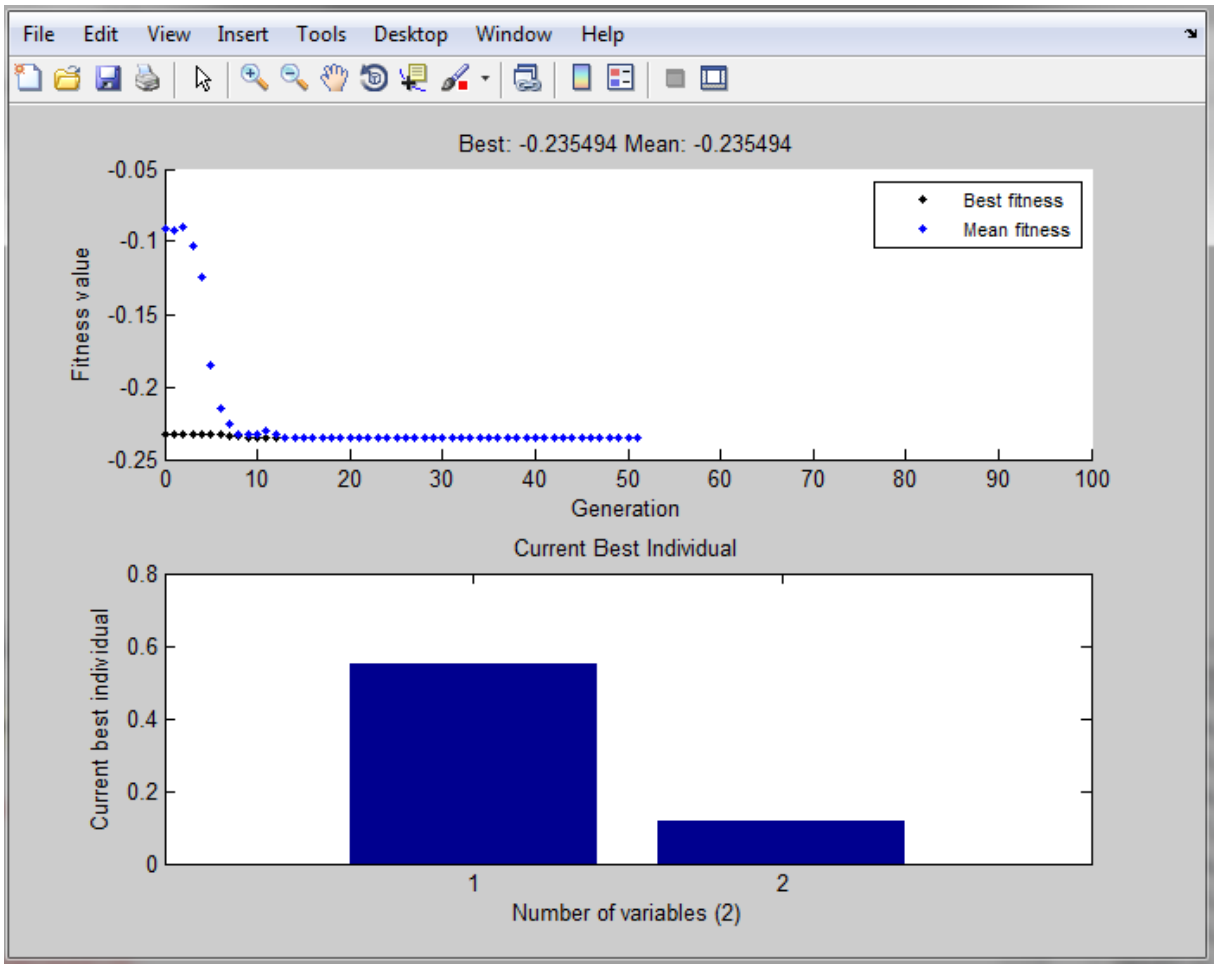
```
function g = GRAF2(x,y)
[x,y]=meshgrid(-3:.2:3,-3:.2:3);
g = ((1-x).^2)*exp(-x.^2-(y+1).^2)-(x-x.^3-y.^3)*exp(-x.^2-y.^2);
surf(x,y,g);
```



Slika 13. Graf funkcije $f(x, y)$

3.3 Traženje minimuma i maksimuma funkcije pomoće genetičkog algoritma

MatLab prema svojoj definiciji uvijek pomoću genetičkog algoritma traži minimum funcije. U ovom slučaju potrebno je naći maksimum funkcije. Maksimum funkcije možemo naći na način da minimiramo funkciju suprotnog predznaka, *-f(x)*. Minimum ove funkcije je zapravo maksimum tražene funkcije.



Slika 14. Minimizirana funkcija i vrijednosti varijabli

Ovdje je prikazano kako genetski algoritam konvergiran prema riješenju, odnosno minimumu funkcije. Vidimo da već nakon 10 generacija(iteracija) dobivamo dobro riješenje. Na donjem grafu vidimo najbolje pojedine kromosome. To su oni kromosomi za koje ova funkcija postiže najmanju vrijednost.

Tablica 5. Optimirane vrijednosti varijabla x i y

Varijabla	Optimirani iznos
X	0.55
Y	0.15

3.3.1 Generirani kod u MatLab programskom paketu

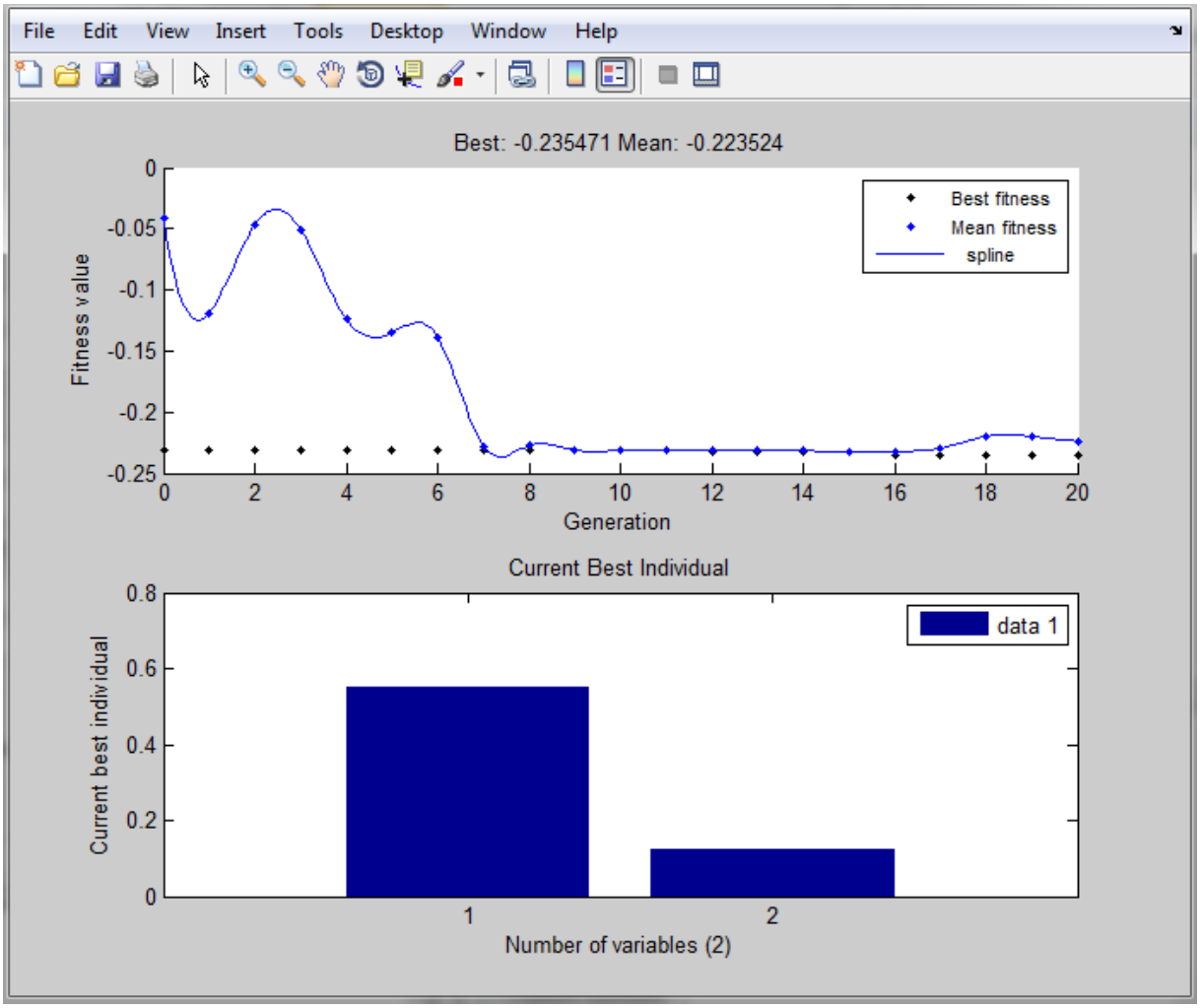
```
function [x,fval,exitflag,output,population,score] =
GENET_ALG1(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'SelectionFcn', @selectionroulette);
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@GRAF,nvars,[],[],[],[],lb,ub,[],[],options);
```

Prikazan je automatski generirani kod iz MatLab-a. U tom kodu se može vidjeti da su postavljene početne postavke za vrijednosti veličine populacije (20), broj generacija te za vjerojatnosti operacija križanja i mutacije. Gornju i donju granicu je potrebno definirati, što se vidi u kodu pod lb (donja granica) i ub (gornja granica).

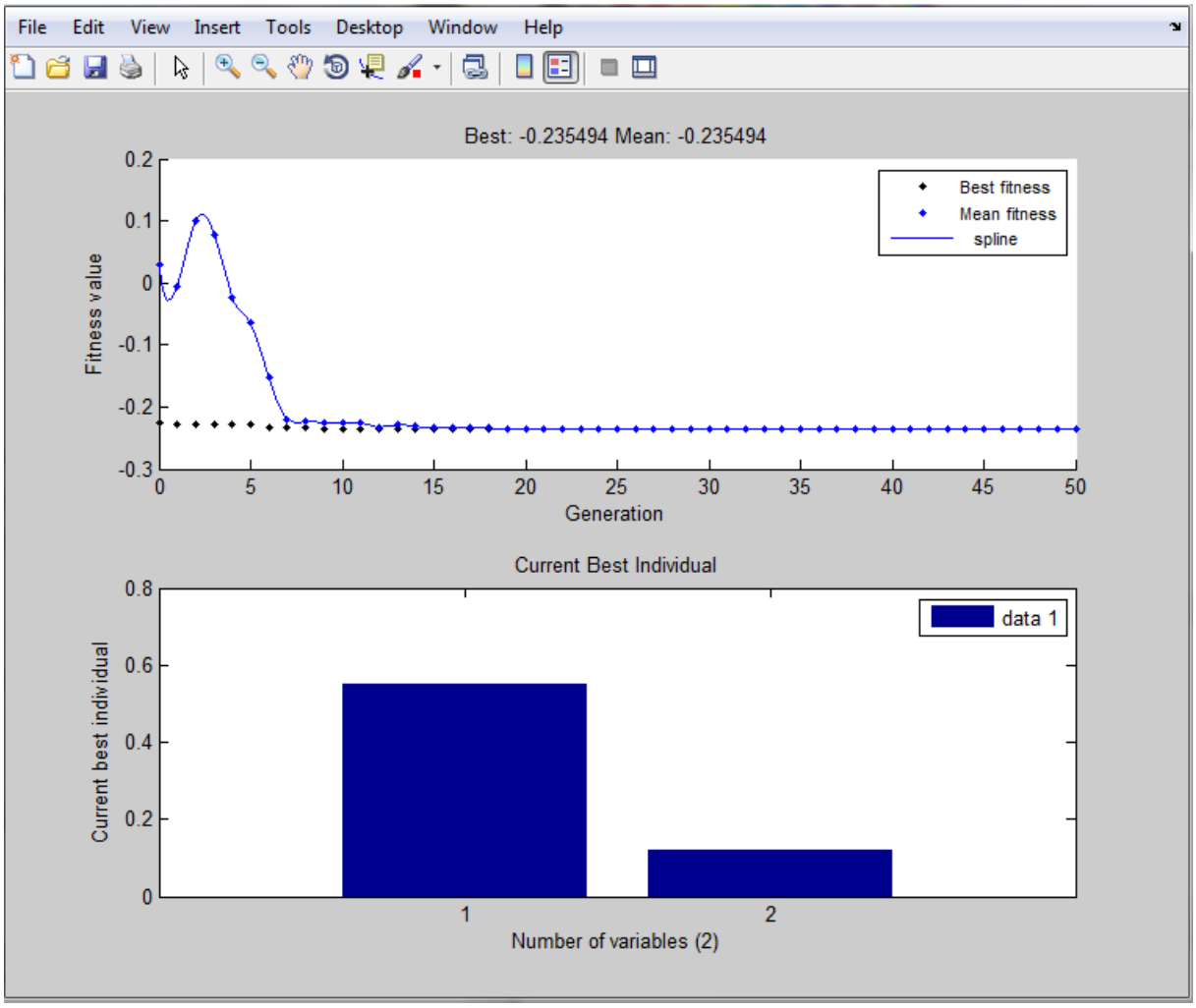
3.4 Grafovi za praćenje učinkovitosti genetskih algoritama

Budući da genetski algoritmi koriste stohastičke tehnike pretraživanja, rezultati se razlikuju od iteracije do iteracije. Rezultat je krivulja koja prikazuje prosječnu krivulju učinkovitosti genetskog algoritma kroz niz generacija. Na tom grafu je također prikazan i potreban broj generacija da bi genetski algoritam postigao traženo rješenje. Može se također vidjeti i brzina konvergencije prema traženom rješenju. Tako nam x-os prikazuje broj generacija koji je bio potreban za postizanje tražene vrijednosti funkcije cilja. Y-os predstavlja vrijednost funkcije cilja, i to za svaku pojedinu generaciju. Pomoću ovih grafova također se može odrediti i najbolji omjer veličine populacije i broja generacija koje su potrebne za postizanje tražene funkcije cilja.



Slika 15. Graf za praćenje učinkovitosti genetskog algoritma

Na prvom grafu su prikazana dobivena rješenja u slučaju kada je odabrana veličina populacije od 10 kromosoma, a broj generacija (iteracija) je 20. Na drugom grafu je dan grafički prikaz veličina varijabla x(1) i x(2). Na sljedećoj slici će biti prikazana ista funkcija, ali u slučaju kada je veličina populacije jednaka 20, a broj generacija 50.



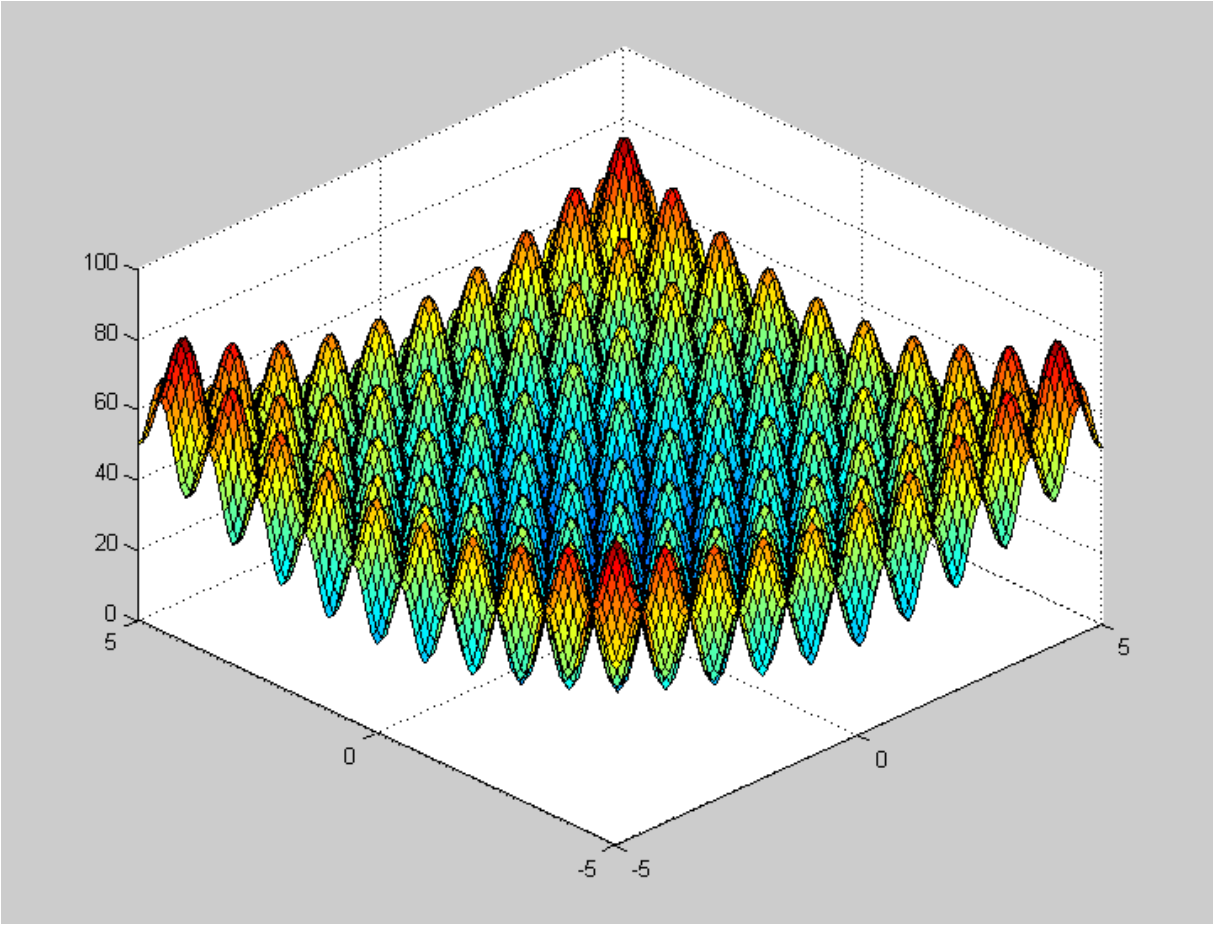
Slika 16. Graf za praćenje učinkovitosti genetskog algoritma s 50 generacija

Na prvom grafu je prikazana krivulja srednje vrijednosti funkcije cilja. Srednje vrijednosti i najbolje vrijednosti funkcija cilja su u oba slučaja približno jednake. Razlika je u odstupanjima sa prve slike u prvih osam generacija. Takva odstupanja se javljaju zbog faktora mutacije, kao i zbog relativno manje veličine populacije. Vidimo sa prve slike da genetski algoritam konvergira prema optimalnom rješenju prilično brzo, no zbog spomenutih odstupanja je dobro ostaviti veću populaciju, kao i veći broj generacija(iteracija).

4. Primjer rješavanje genetskih algoritama pomoću MatLab programskog paketa

Probleme optimiranja pomoću genetskih algoritama u MatLab programskom paketu možemo rješiti pomoću MatLab-ovog GA optimizacijskog alata. U sljedećem primjeru bit će prikazana Rastringova funkcija koja ima velik broj minimuma i maksimuma, ali samo jedan globalni. Cilj je pronaći vrijednosti varijabli za globalni minimum. Rastringova funkcija glasi:

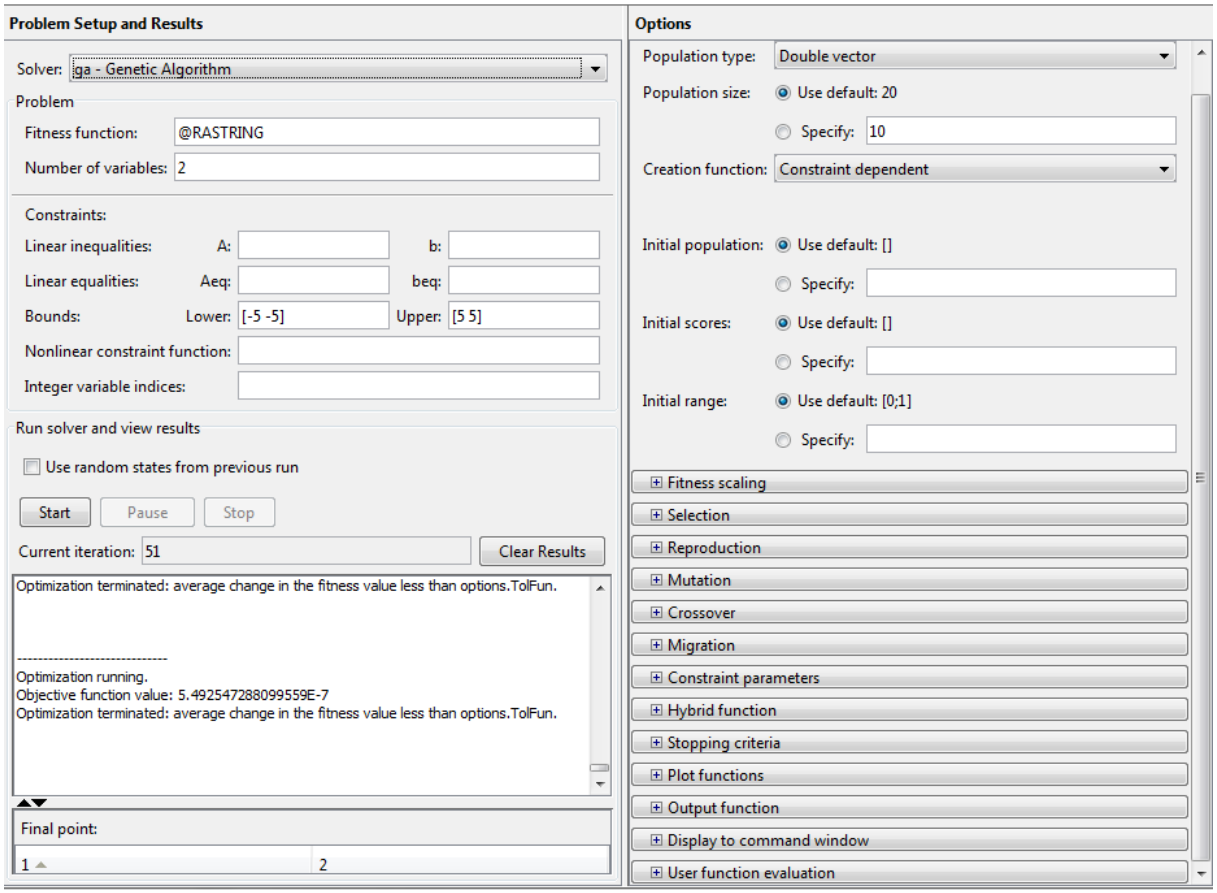
$$Ras(x) = 20 + x_1^2 + x_2^2 - 10[\cos(2 * \pi * x_1) + \cos(2 * \pi * x_2)] \quad \textbf{(6)}$$



Slika 17. Rastringova funkcija

Za Rastringovu funkciju potrebno je pronaći globalni minimum, koji se pojavljuje samo u točci (0,0). Funkcija ima velik broj lokalnih minimuma i zbog toga će se koristiti za testiranje

genetskog algoritma, kako bi se provjerilo dali genetski algoritam može pronaći globalni minimum unatoč prisutstvu velikog broja lokalnih minimuma.

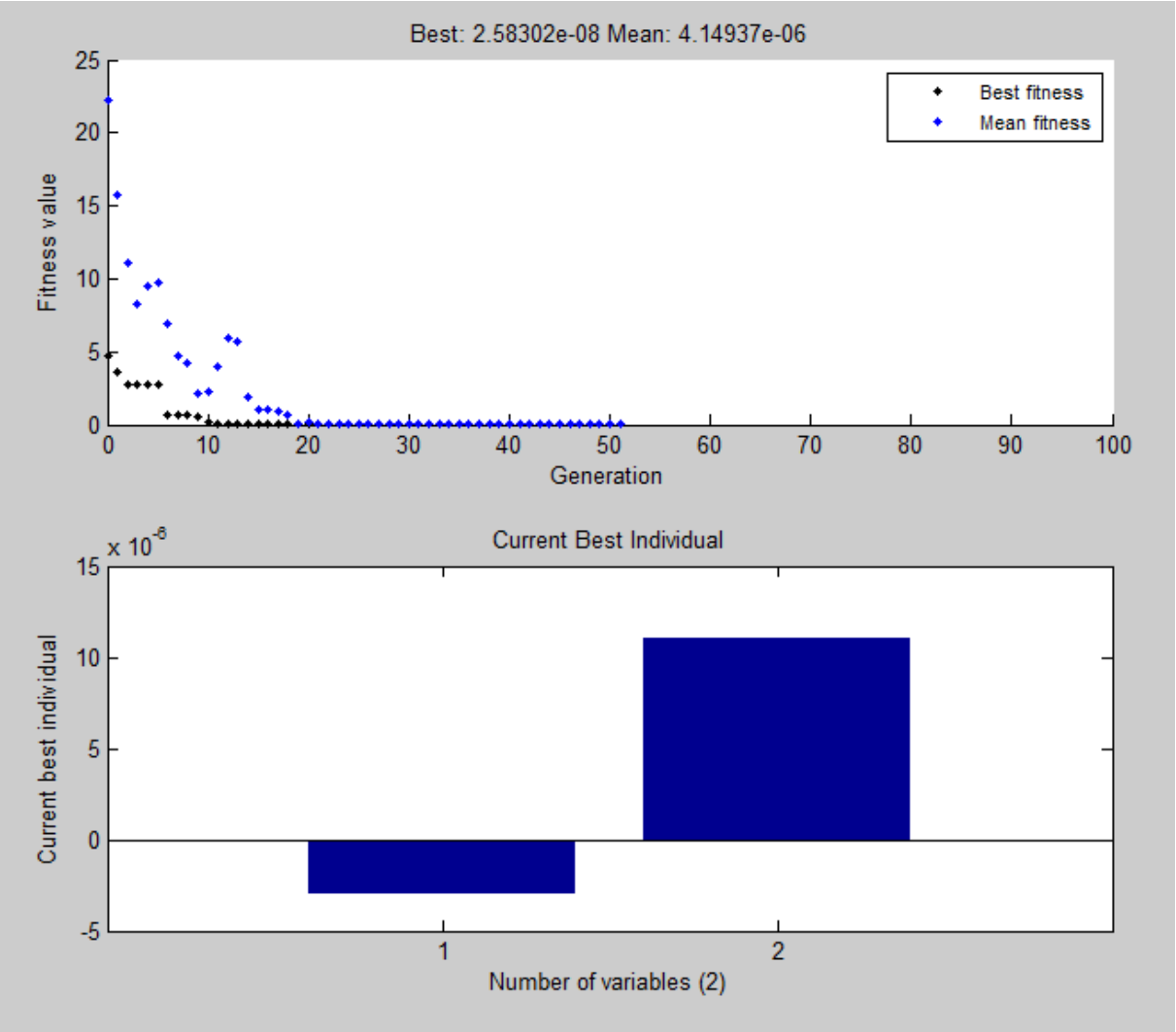


Slika 18. Postavke za rješavanje Rastringove funkcije

```
function [x,fval,exitflag,output,population,score] = GEN(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@RASTRING,nvars,[],[],[],[],lb,ub,[],[],options);
```

Prikazan je generirani kod za rješavanje Rastringove funkcije u MatLabu. Pomoću postavke nvars definira se broj varijabli. U Rastringovoj funkciji je broj varijabli 2, donje i gornje granice se mogu definirati, na primjer, između -5 i 5, kako je prikazano na grafu funkcije.



Slika 19. Rezultati optimizacije Rastringove funkcije

Tablica 6. Optimizirane vrijednosti Rastringove funkcije

Varijabla	Vrijednost
X1	$-3.01 * e^{-6}$
X2	$1.1 * e^{-5}$

Znamo da je globalni minimum Rastringove funkcije u točci (0,0) pa se može reći kako genetski algoritam dobro obavlja posao minimiziranja ove funkcije. Vrijednosti varijabli variraju vrlo blizu nule, što znači kako je genetski algoritam dobro obavio optimiranje funkcije.

4.1 Optimiranje funkcije hrapavosti pri obradi odvajanjem čestica[1]

Genetski algoritmi primjenu također pronalaze i prilikom optimiranja režima rada u cilju dobivanja najmanje vrijednosti parametra hrapavosti. Kako bi uspješno dobili optimalne vrijednosti režima obrade koji utječu na hrapavost površine, potrebno je matematički odrediti funkciju hrapavosti u ovisnosti od brzine rezanja, dubine rezanja i posmaka. Ovu funkciju moguće je dobiti samo eksperimentalnim istraživanjima. Ovo istraživanje provedeno je na tokarskom stroju Trens SLB-500, proizvedenom 2004. godine.[1]

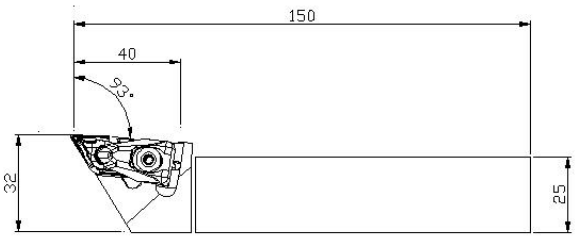
Slika 20. Tokarski obradni centar Trens SBL-500[1]

Tehničke karakteristike tokarskog obradnog centra Trens SBL-500 su:

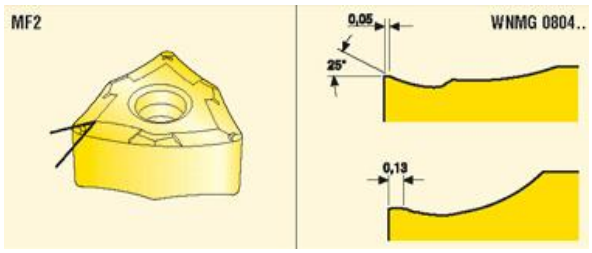
- Dimenzije: 1738 x 1900 mm (širina x visina)
- Težina: 6620 kg
- Radni prostor stroja:
  - najveći promjer tokarenja iznad kliznih staza: 405 mm
  - najveća duljina tokarenja: 845 mm
- Brzina posmičnih gibanja:
  - radna brzina za X os: 1÷10 m/min,
  - radna brzina za Z os: 1÷10 m/min,
  - pozicioniranje za X os: 18 m/min,
  - pozicioniranje za Z os: 24 m/min
- Glavno vreteno:
  - frekvencija vrtnje: 4200 o/min,
  - snaga: 18,5 KW,
  - "C" os za indeksiranje glavnog vretena
- Magazin alata:
  - - broj mjesta u revolverskoj glavi (SAUTER): 12 (6+6)
  - prihvat pogonjenog alata
  - snaga pogonjenih alata: 3,0 kW
  - frekvencija vrtnje: 3500 o/min
- Transporter odvojene čestice
- Upravljanje: SIEMENS 810 D SHOPTURN[1]

4.1.1 Alat[1]

Za ovo eksperimentalno istraživanje korišten je alat tvrtke Seco Tools GmbH. Izabran je držač oznake DDJNL2525M15 (λ=-6°) i rezna pločica oznake DNMG150608-MF2CM.[1]



Slika 21. Držač pločice[1][7]

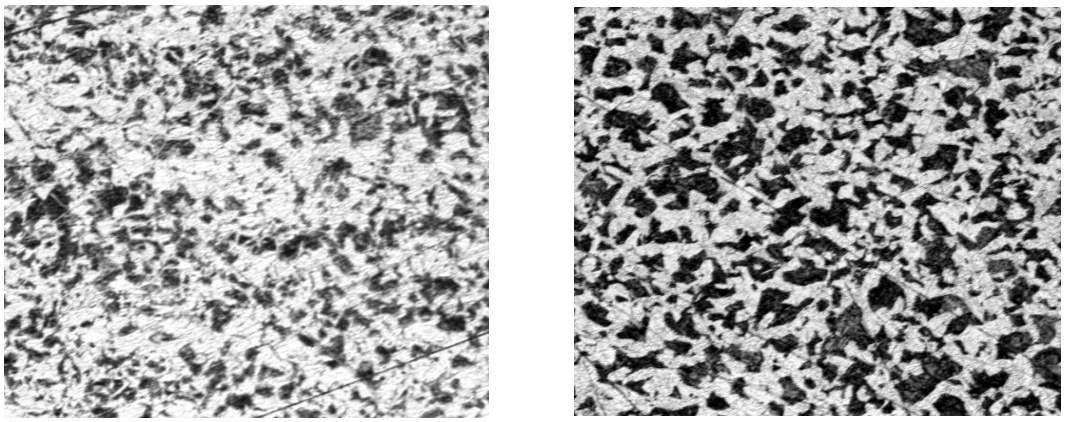


Slika 22. Rezna pločica tvrtke Seco Tools[1][7]

4.1.2 Materijal obratka[1]

Istraživanje je provedeno na materijalu Č4320. Taj materijal je tipičan predstavnik čelika za cementiranje i ima vrlo široku primjenu u strojogradnji.

Materijal je nabavljen u šipki ϕ100 mm, u meko žarenom stanju. Nakon valjanja mehaničke osobine materijala su po presjeku i dužini šipke neujednačene, a struktura neujednačena i usmjerena. Kako bi mehaničke osobine i struktura bile konstanta pokusa, na materijalu je proveden postupak normalizacije. Šipka je zagrijana na temperaturu 870 °C i hlađena na zraku.[1]



a) **b)**

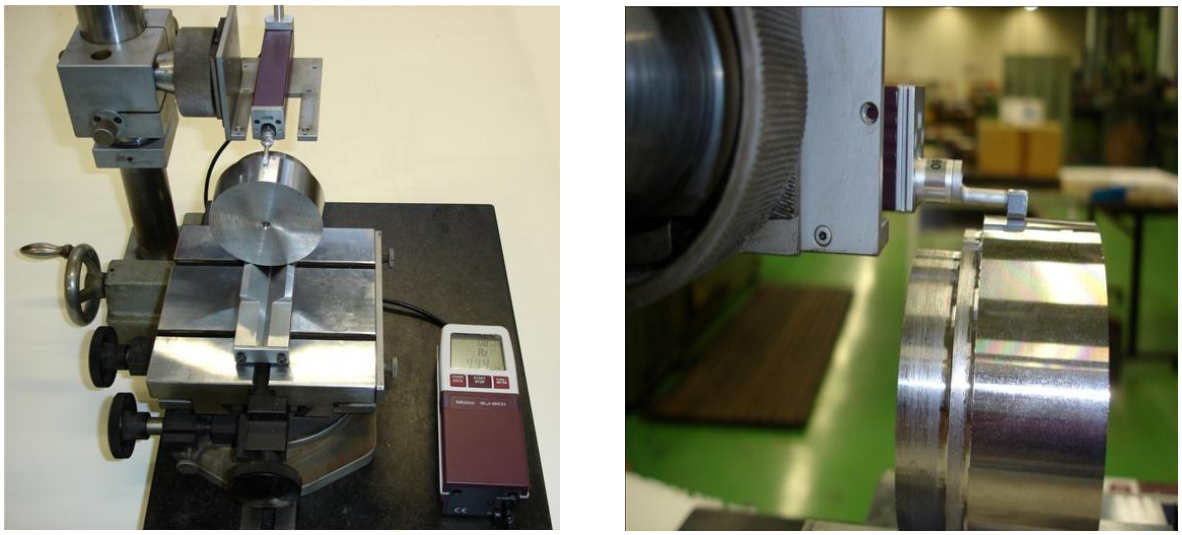
Slika 23. Mikrostruktura materijala Č4320 pri povećanju 100x: a) isporučeno stanje, b) normaliziran[1]

4.1.3 Mjerni alati[1]

Za mjerenje parametara hrapavosti obrađene površine korišten je mjerni uređaj Mitutoyo SP-201. Rezultati mjerenja vidljivi su na zaslonu uređaja i ne tiskaju se na papir.[1]

Karakteristike ovog mjernog uređaja su:

- Mjerni profili: R, P, DIN 4776
- Filter: 2CR-75%, PC-75%, PC 50%
- Prikaz veličina: Ra, Ry, Rz, Rq, Sm, S, Pc, R3z, mr, Rt, Rp, Rk, Rpk, Rvk, Mr1, Mr2, A1, A2, Vo
- Snaga: 1,5 W
- Standardna duljina mjerenja: 0-12.5mm
- Standardni put ticala: 0.25, 0.8, 2.5mm
- Dimenzije mjernog uređaja: 156.5 x 62 x 52mm
- Dimenzije ticala: 115 x 23 x 26mm
- Težina mjernog uređaja: 0.3kg
- Težina ticala: 0.2kg



a) **b)**

Slika 24. a) Uređaj za mjerenje parametara hrapavosti površine Mitutoyo SP-201, b) Zahvat ticala mjernog uređaja i obratka[1]

4.1.4 Ulazni podaci eksperimenta[1]

Tablica 7. Razine variranja faktora za matricu plana pokusa „2^3+6+6“[1]

Faktori	$x_{i,min}$	$x_{i,sr}$	$x_{i,max}$
v_c [m/s]	200	350	500
a_p [mm]	0,8	1,4	2
f [mm]	0,1	0,2	0,3
Kod faktora	-1	0	1

Tablica 8. Matrica plana pokusa „2^3+6+6“[1]

	Ulazna veličina			**Kod ulazne veličine**		
Redni broj mjerenja	$\mathbf{v_c}$ [m/s] (m/min)	$\mathbf{a_p}$ [mm]	**f** [mm]	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$
1.	3,33 (200)	0,8	0,1	-1	-1	-1
2.	8,33 (500)	0,8	0,1	+1	-1	-1
3.	3,33 (200)	2	0,1	-1	+1	-1
4.	8,33 (500)	2	0,1	+1	+1	-1
5.	3,33 (200)	0,8	0,3	-1	-1	+1
6.	8,33 (500)	0,8	0,3	+1	-1	+1
7.	3,33 (200)	2	0,3	-1	+1	+1
8.	8,33 (500)	2	0,3	+1	+1	+1
9.	5,83 (350)	1,4	0,2	0	0	0
10.	5,83 (350)	1,4	0,2	0	0	0
11.	5,83 (350)	1,4	0,2	0	0	0
12.	5,83 (350)	1,4	0,2	0	0	0
13.	5,83 (350)	1,4	0,2	0	0	0
14.	5,83 (350)	1,4	0,2	0	0	0
15.	3,33 (200)	1,4	0,2	-1	0	0
16.	8,33 (500)	1,4	0,2	+1	0	0
17.	5,83 (350)	0,8	0,2	0	-1	0
18.	5,83 (350)	2	0,2	0	+1	0
19.	5,83 (350)	1,4	0,1	0	0	-1
20.	5,83 (350)	1,4	0,3	0	0	+1

4.1.5 Rezultati pokusa[1]

Rezultati mjerenja parametara hrapavosti obrađene površine za postavljeni centralno kompozitni plan pokusa prvog reda (2^3+4) i drugog reda (2^3+6+6) prikazani su u tablici. Za svaku obrađenu površinu napravljena su tri mjerenja, a zatim je izračunata srednja vrijednost koja predstavlja rezultat mjerenja.[1]

Tablica 9. Rezultati mjerenja[1]

Redni broj mjerenja	Ulazne veličine			Izlazne veličine	
	$\mathbf{v_c}$ [m/s] (m/min)	$\mathbf{a_p}$ [mm]	**f** [mm]	$\mathbf{R_a}$ [μm]	$\mathbf{R_z}$ [μm]
1.	3,33 (200)	0,8	0,1	0,53	3,39
2.	8,33 (500)	0,8	0,1	0,48	2,53
3.	3,33 (200)	2	0,1	0,64	4
4.	8,33 (500)	2	0,1	0,58	3,47
5.	3,33 (200)	0,8	0,3	3,75	14,6
6.	8,33 (500)	0,8	0,3	3,61	14,2
7.	3,33 (200)	2	0,3	3,74	15,2
8.	8,33 (500)	2	0,3	3,62	14,8
9.	5,83 (350)	1,4	0,2	1,56	7,88
10.	5,83 (350)	1,4	0,2	1,49	7,36
11.	5,83 (350)	1,4	0,2	1,42	6,91
12.	5,83 (350)	1,4	0,2	1,5	7,29
13.	5,83 (350)	1,4	0,2	1,49	7,56
14.	5,83 (350)	1,4	0,2	1,48	7,16
15.	3,33 (200)	1,4	0,2	1,71	8,52
16.	8,33 (500)	1,4	0,2	1,27	6,2
17.	5,83 (350)	0,8	0,2	1,43	6,83
18.	5,83 (350)	2	0,2	1,55	7,89
19.	5,83 (350)	1,4	0,1	0,55	3,23
20.	5,83 (350)	1,4	0,3	3,68	14,7

4.1.6 Rezultati ispitivanja polinomom drugog stupnja[1]

Polazna jednadžba u obliku polinoma drugog stupnja za mjereni parametar R_z glasi:

$$R_z = C + p_1 \cdot v + p_{11} \cdot v^2 + p_2 \cdot a_p + p_{22} \cdot {a_p}^2 + p_3 \cdot f + p_{33} \cdot f^2 + p_{13} \cdot v \cdot\cdot f + p_{12} \cdot v \cdot a_p + p_{23} \cdot a_p \cdot f \quad \textbf{(7)}$$

Podaci dobiveni mjerenjem dani u tablici statistički su obrađeni te su na temelju pojedinačnog testa (t-test) izbačeni parametri koji su bili iznad praga signifikantnosti (>α=0,05). Preostali procijenjeni parametri prikazani su u tablici.[1]

Tablica 10. Procjena parametara modela – polinom drugog stupnja (2^3+6+6)[1]

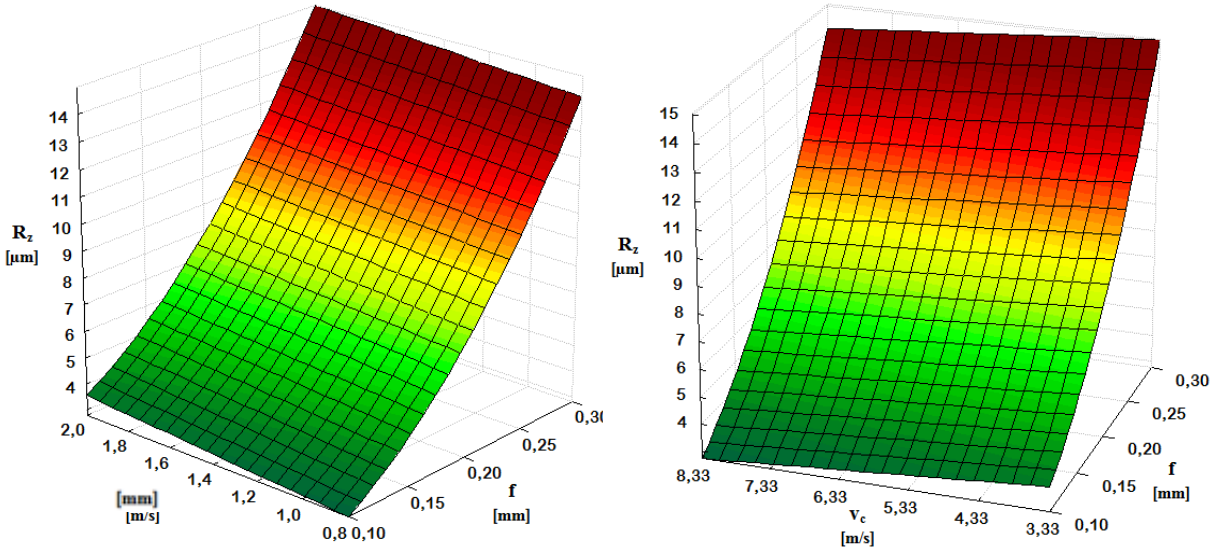
Parametar	Procjena	Standardna greška	t-vrijednost (*d.f.*=16)	Razina signifikantnosti (p-vrijednost)
C	1,9190	0,4187	4,58287	0,00031
p_1	-0,1804	0,04716	-3,82569	0,00149
p_2	0,6350	0,1965	3,2319	0,00522
p_{33}	142,897	2,90218	49,2377	0,0000

- Test veličina (t-vrijednost) se očitava iz tablice za pojedini parametar:
  - za konstantu C.. => t_C=4,58287
  - za parametar p_3 => t_{p_1}=-3,82569
  - za parametar p_3 => t_{p_2}=3,2319
  - za parametar p_3 => $t_{p_{33}}$=49,2377
- Budući da se test provodi uz razinu signifikantnosti α =5% tada je za broj mjerenja n=20 i broj stupnjeva slobode *d.f.*=16 iz tablice za t raspodjelu:
  - $t_{\propto}(n-k) = t_{0,05}(16) = 2{,}120$
- Usporedbom (t_{p_1}, t_{p_2}, $t_{p_{33}}$ i t_C) i $t_{\propto}$ proizlazi zaključak da su t_C i $t_{p_{ij}} > t_{\propto}$ te se uz razinu signifikantnosti 5% nulta hipoteza odbacuje, tj. ne prihvaća se pretpostavka da su parametri t_{p_1}, t_{p_2}, $t_{p_{33}}$ i t_C suvišne varijable u modelu.
- Zaključak se mogao donijeti i promatranjem p-vrijednosti jer se nulta hipoteza odbacuje uz razinu signifikantnosti α ako je p-vrijednost < α (u našem slučaju p-vrijednost=0,00 < α=0,05).

Temeljem toga, procijenjena regresijska jednadžba za mjereni parametar R_z glasi:

$$R_z = C + p_1 \cdot v + p_2 \cdot a_p + p_{33} \cdot f^2 =$$
$$= 1{,}919 + (-0{,}1804) \cdot v + 0{,}635 \cdot a_p + 142{,}897 \cdot f^2 \quad \textbf{(8)}$$

uz procjenu standardne greške 0,4187za konstantu *C*, 0,04715 za parametar p_1, 0,1965 za parametar p_2 i 2,9021 za parametar p_{33}.[1]



Slika 25. Dijagram ovisnosti R_z=f(a_p, f) za srednju brzinu v_c=5,83 m/s i dijagram ovisnosti R_z=f(v_c, f) za srednji posmak a_p=1,4 mm – model drugog stupnja (2^3+6+6)[1]

Kako iz dobivene jednadžbe, tako i promatranjem dijagrama dolazi se do zaključka kako za dane uvijete prema modelu u obliku polinoma drugog stupnja (plan pokusa 2^3+6+6) pokazatelj hrapavosti R_z ovisi najviše o vrijednosti posmaka (f), tj. povećanjem posmaka raste i vrijednost R_z dok povećanjem dubine rezanja vrijednost R_z, raste neznatno. Isto tako smanjenjem brzine rezanja postoji određeni porast vrijednosti R_z.[1]

4.1.7 Optimizacija funkcije hrapavosti pri strojoj obradi

Dobivenu funkciju hrapavosti sada treba optimizirati, i to na taj način da dobijemo najmanji parametar hrapavosti za zadane režime rada. Ovdje se zapravo provodi optimizacija režima rada, i to brzine rezenja, posmaka i dubine rezanja. Optimizacija se provodi u više slučajeva, i to u slučaju kada je konstanta brzina rezanja, posmak ili dubina rezanja. Dobiveni rezultati zatim će poslužiti kao podloga za usporedbu učinkovitoski gentskih algoritama. U posljednjem primjeru je prikazano optimirano rješenje u slučaju da su svi režimi obrade varijabilni.

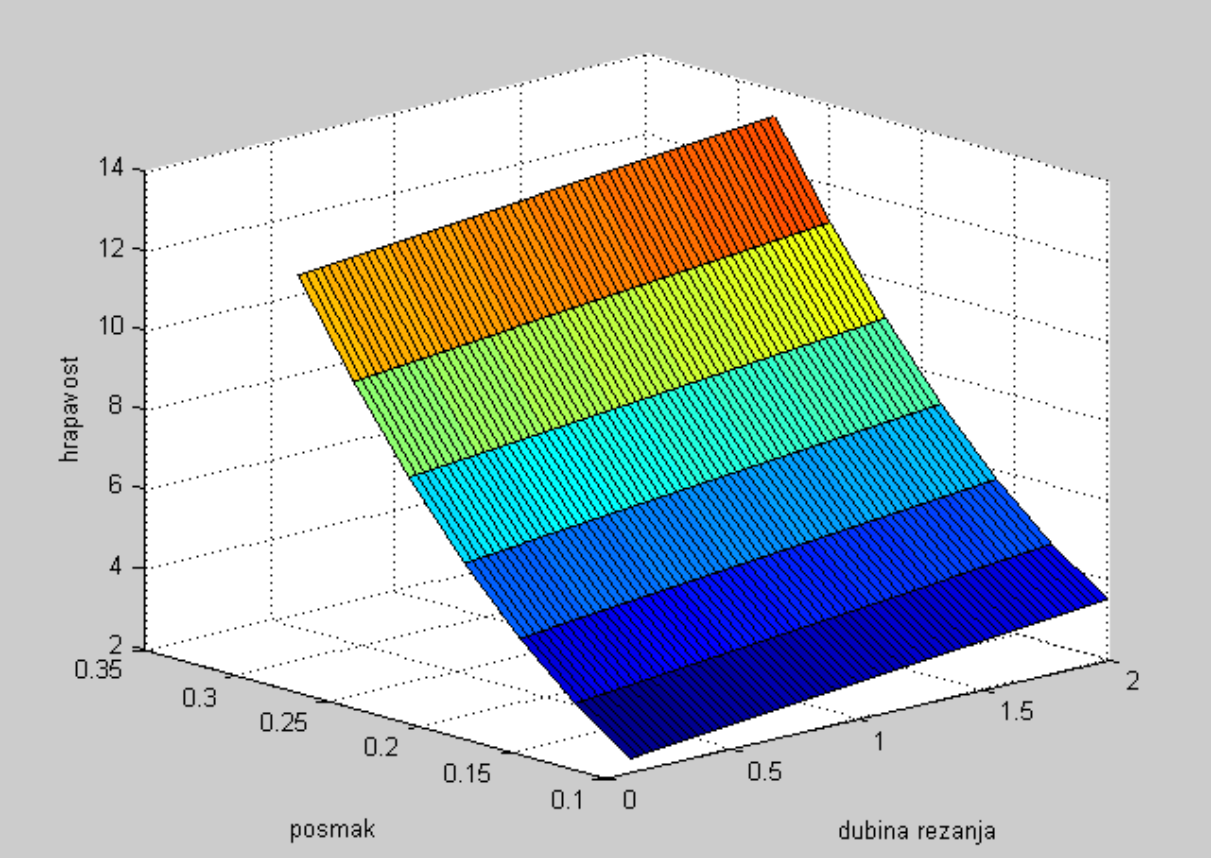
4.2 Rezultati optimizacije

Rezulate optimizacije može se prikazati grafički pomoću MatLab programskog paketa, i to za sva tri slučaja.

4.2.1 Optimizirana funkcija s konstantnom brzinom rezanja

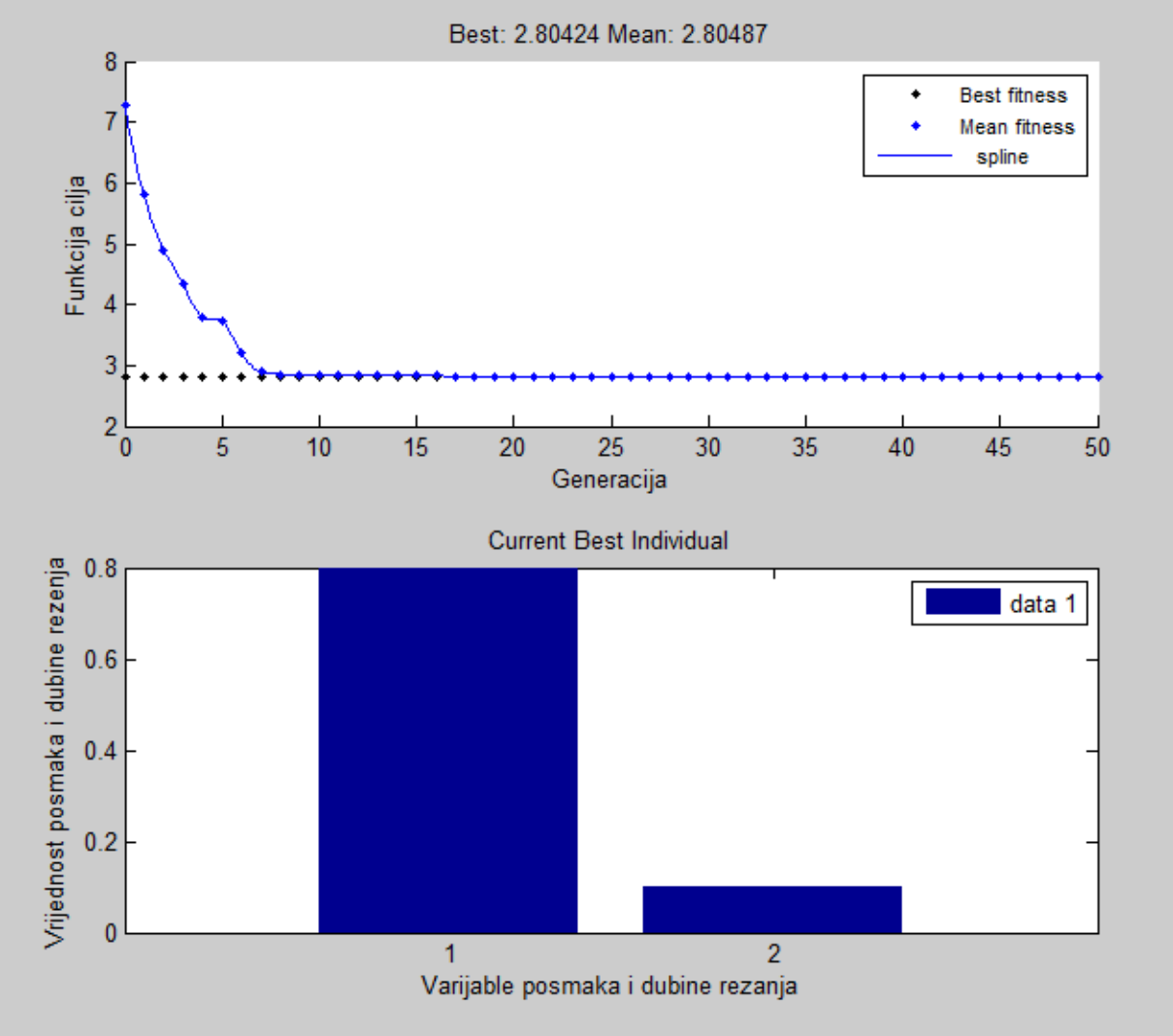
Funkcija je prvo optimizirana za brzinu rezanja u iznosu od 5,83 m/s. To je srednja brzina rezanja dobivena eksperimentalno.

$$R_z = C + p_1 \cdot v + p_2 \cdot a_p + p_{33} \cdot f^2 = \\ = 1{,}919 + (-0{,}1804) \cdot 5{,}83 + 0{,}635 \cdot a_p + 142{,}897 \cdot f^2 \quad \textbf{(9)}$$



Slika 26. Dijagram ovisnosti parametra hrapavosti o dubini rezanja i posmaku

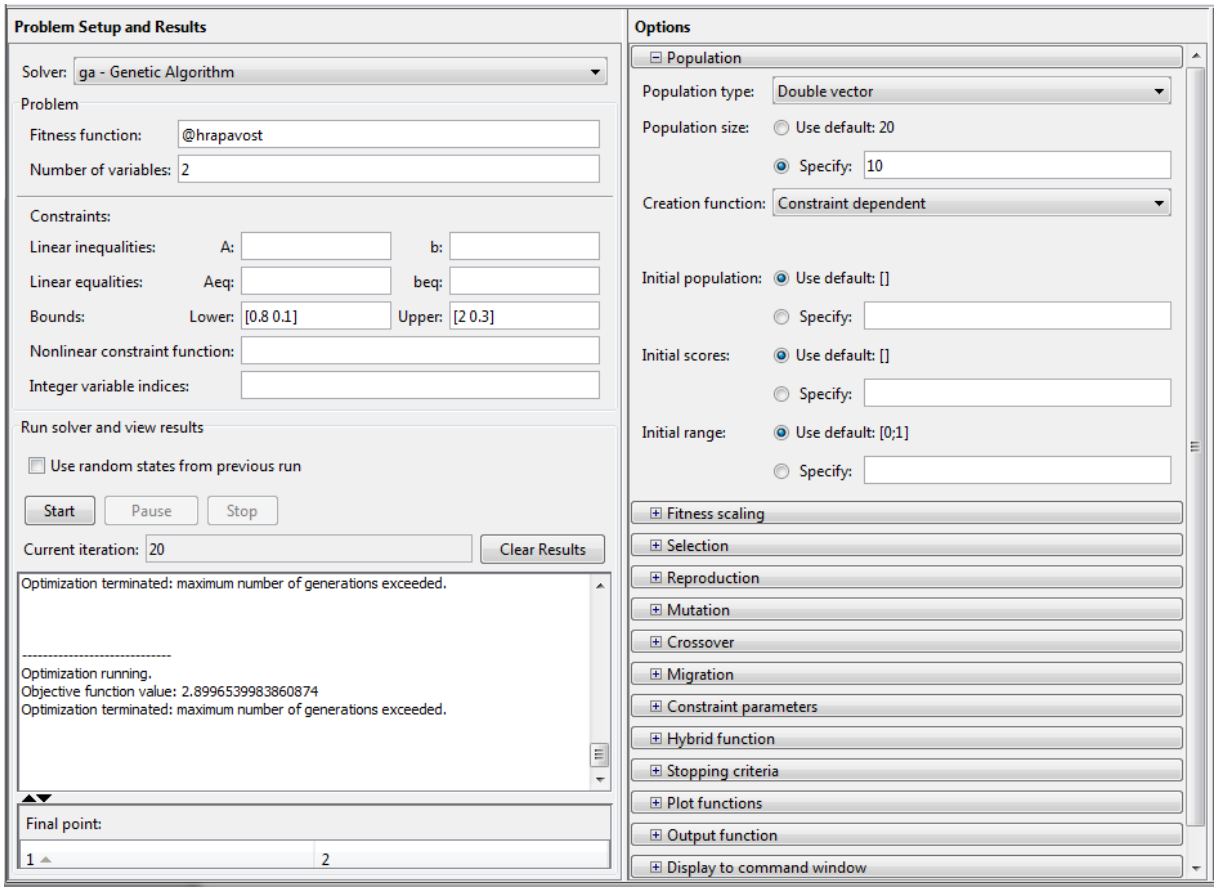
Sa grafa je vidljiva funkcija preko koje se vidi za koje posmake i dubine rezenja se postiže optimalna hrapavost. Očekuje se da preko genetskog algoritma dobijemo rješenja koja se podudaraju sa podacima sa grafa.



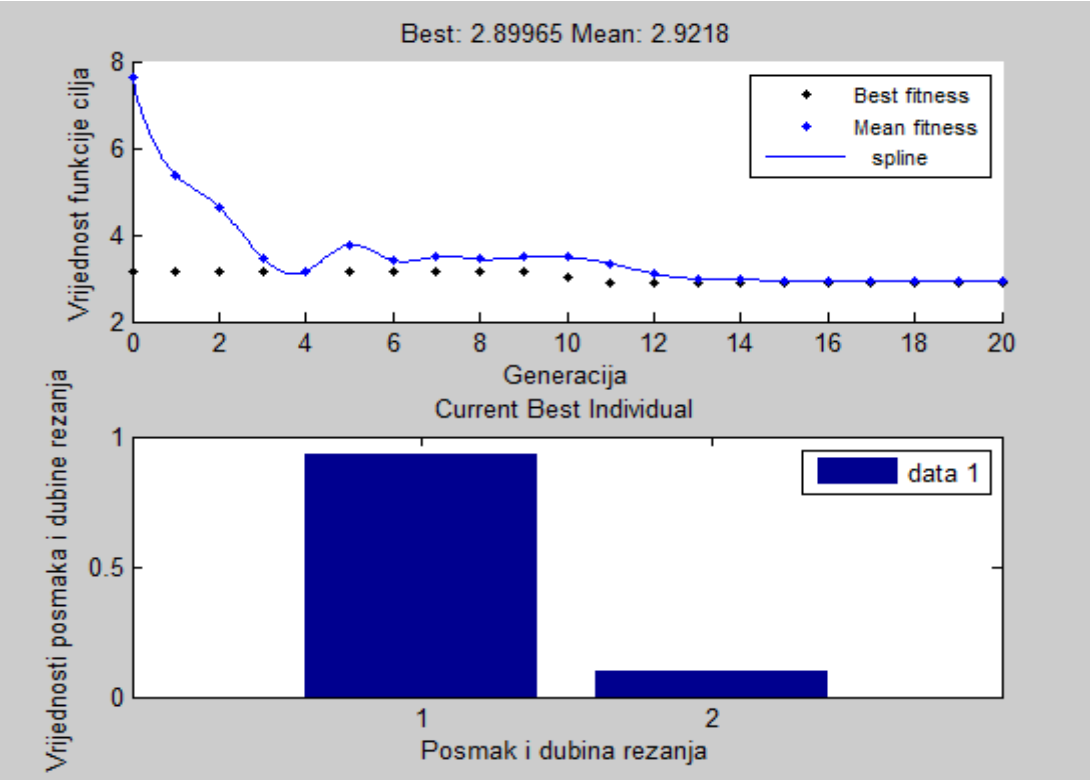
Slika 27. Rezultati optimizacije uz konstantnu brzinu rezanja i 50 iteracija

Sa grafa za praćenje izvedbe genetskog algoritma vidi se da su za srednju brzinu rezanja od 5,83 m/s najoptimalniji parametri i oni najmanji mogući. Ovo je sasvim logično zbog toga što se na ovaj način postiže najfinija obrada. Sa grafa sa praćenje izvedbe genetskog algoritma se vidi brzina konvergencije optimalnom rješenju. Uz postavke za početnu veličinu populacije od 20 kromosoma i 50 iteracija, vidimo da se optimalno rješenje postiže već nakon 10 iteracija. Potrebno je testirati potrebu za ovolikim brojem iteracija. U sljedećem primjeru genetski algoritam rješavati će isti problem, ali sa drugim postavkama. Za sljedeći primjer

odabrane su postavke od 20 generacija i to tako da se svaka generacija sastoji od 10 kromosoma.



Slika 28. Postavke genetskog algoritma za optimizaciju



Slika 29. Rezultati optimizacije uz konstantnu brzinu rezanja i 20 generacija

Sa prikazanih grafova za 20 iteracija i veličinu generacije od 10 kromosoma, vidimo da ne postižemo optimalno rješenje. To je zbog toga što je odabrana premalena veličina početne populacije.

Tablica 11. Usporedba rezulta obzirom na broj generacija uz konstantnu brzinu rezanja

	Brzina rezanja[m/s]	Dubina rezanja [mm]	Posmak [mm]	Hrapavost [μm]
20 iteracija	5,83	0,8	0,13	2,92
50 iteracija	5,83	0,8	0,10	2,80
100 iteracija	5,83	0,8	0,10	2,80
150 iteracija	5,83	0,8	0,10	2,80

Iz rezultata se vidi kako za odabranu brzinu rezanja genetski algoritam optimira funkciju tako da daje najniže vrijednosti posmaka i dubine rezanja. Minimalna postignuta hrapavost iznosi 2,80 μm, i to u slučaju kada postavimo barem 50 generacija genetskog algoritma.

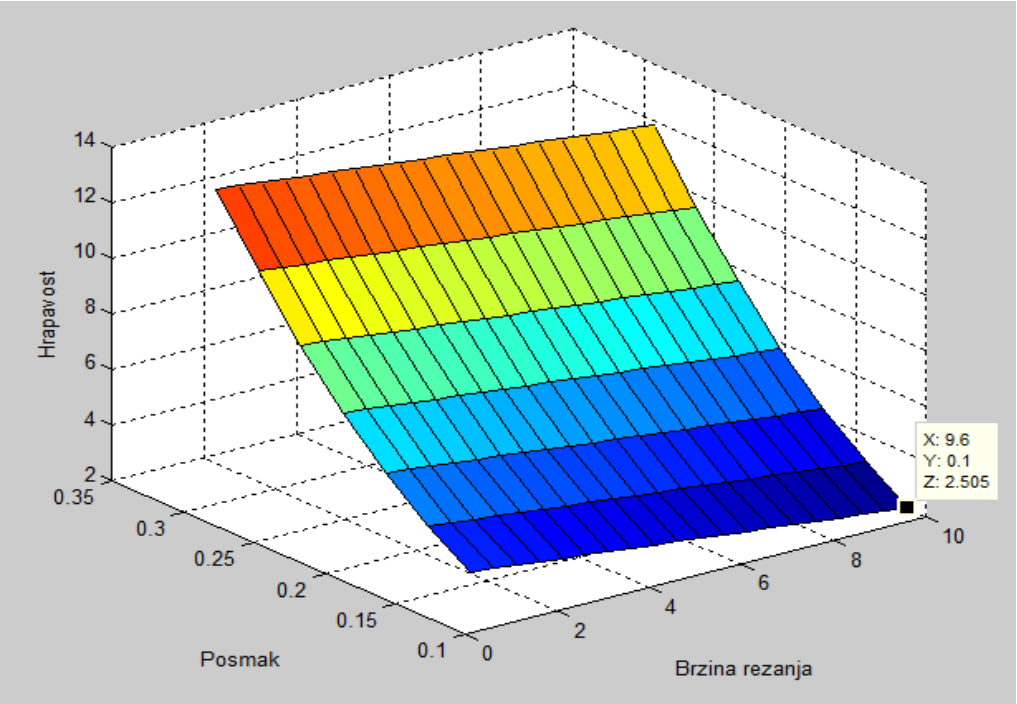
```
function [x,fval,exitflag,output,population,score] = GEN(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@HRAPAVOST,nvars,[],[],[],[],lb,ub,[],[],options);
```

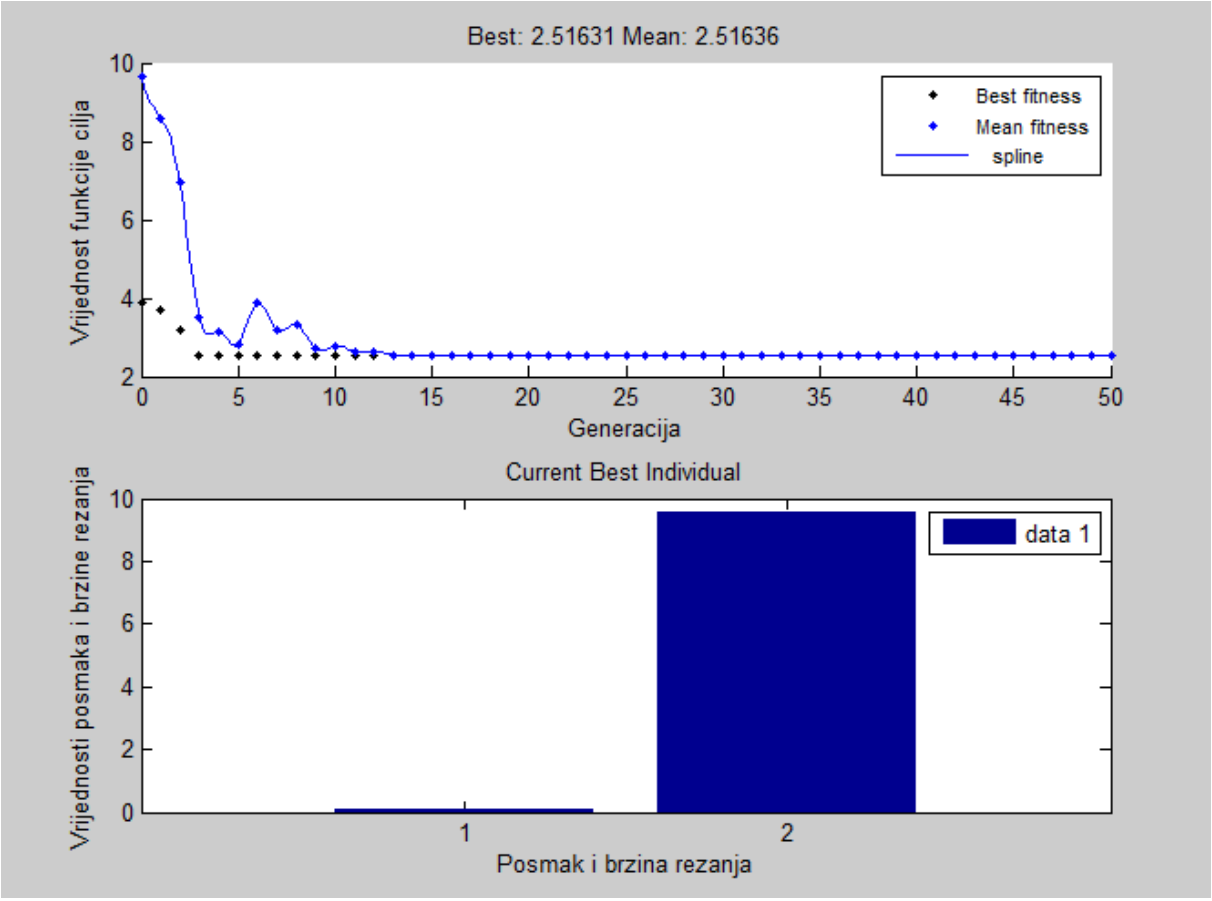
4.2.2 Optimizacija funkcije s konstantnom dubinom rezanja

U sljedećem primjeru biti će prikaza ista funkcija, ali u ovom slučaju tražimo optimalne režime rada uz konstantnu dubinu rezanja. Odabrana je srednja dubina rezanja dobivena eksperimentalno u iznosu od 1,4 mm. Vrijednosti brzine rezanja kreću se između 3 i 10 m/s, a posmaka između 0,1 i 0,3 mm.

$$R_z = C + p_1 \cdot v + p_2 \cdot a_p + p_{33} \cdot f^2 = \\ = 1{,}919 + (-0{,}1804) \cdot v + 0{,}635 \cdot 1{,}4 + 142{,}897 \cdot f^2 \quad \textbf{(10)}$$



Slika 30. Dijagram ovisnosti parametra hrapavosti o brzini rezanja i posmaku

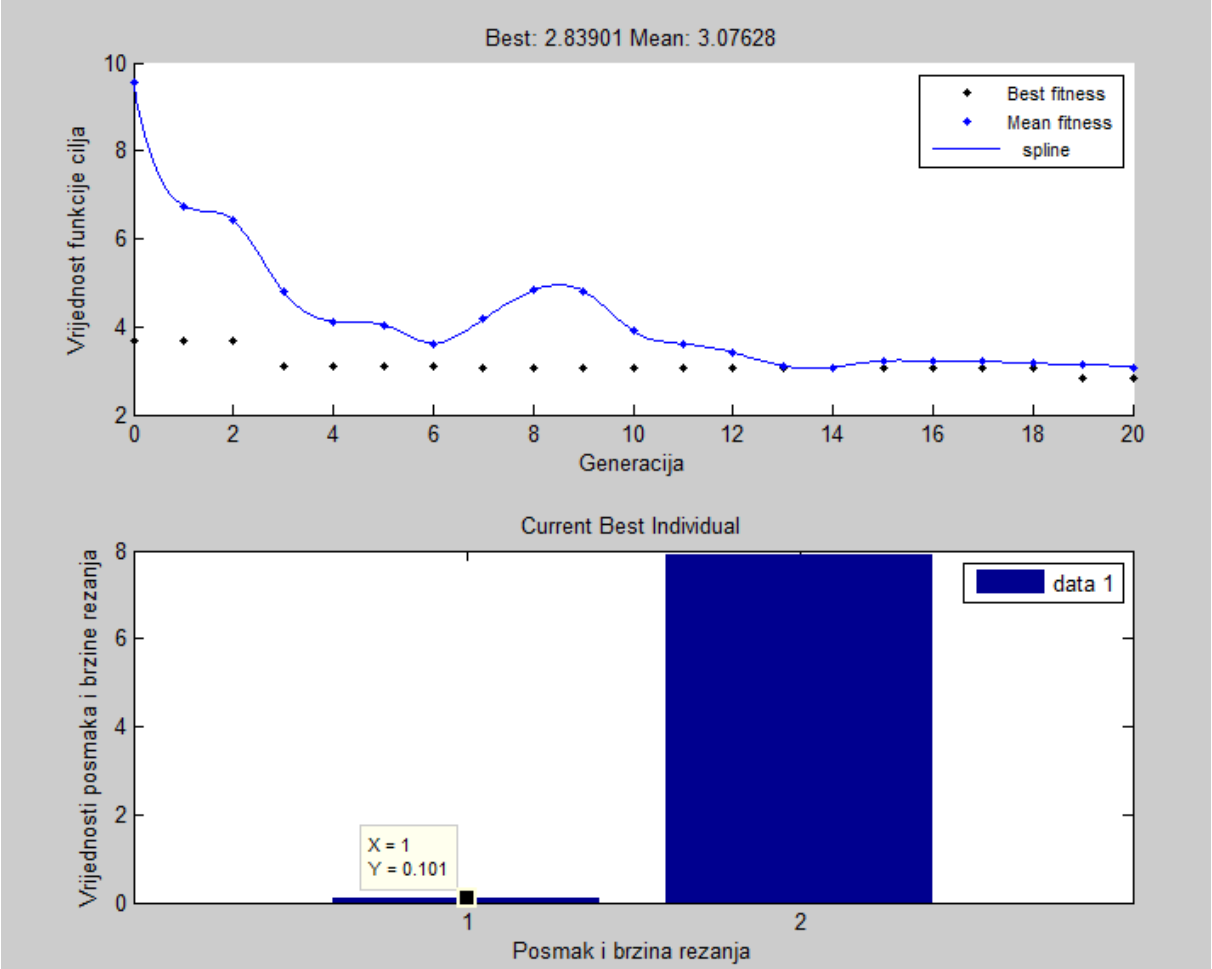


Slika 31. Rezultati optimizacije uz konstantnu dubinu rezanja i 50 generacija

Sa grafa se vidi kako se minimalna hrapavost postiže u slučaju kada je brzina rezanja maksimalna, a posmak minimalan. Ponovno je uočljivo kako genetski algoritam konvergira optimalnom rješenju već nakon 10 iteracija. Ovdje se zaključuje kako su ovdje postignuti optimalni parametri za operatore križanja i mutacije.

```
function [x,fval,exitflag,output,population,score] = GEN(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@HRAPAVOST,nvars,[],[],[],[],lb,ub,[],[],options);
```



Slika 32. Rezultati optimizacije uz konstantnu dubinu rezanja i 20 generacija

Na drugom grafu prikazani su dobiveni rezultati za postavke od 10 kromosoma u populaciji i 20 generacija. Nakon proračunatih 20 iteracija, vidi se kako je dobivena srednja hrapavost niža nego u slučaju kada smo postavili 50 iteracija sa 20 članova svake generacije.

Tablica 12. Usporedba rezulta obzirom na broj generacija uz konstantnu dubinu rezanja

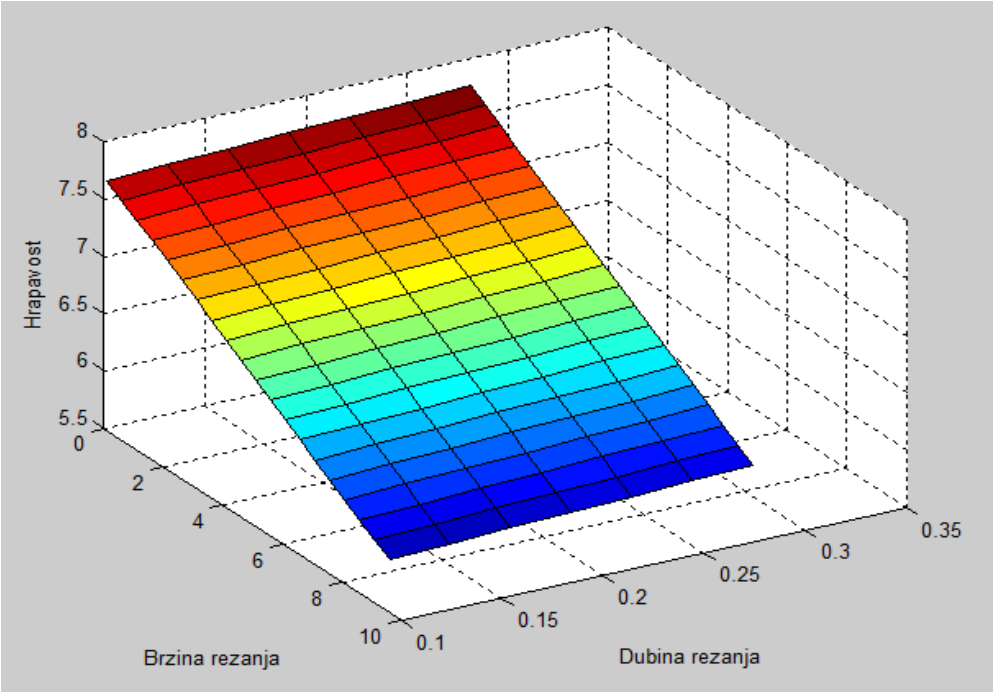
	Brzina rezanja [m/s]	Dubina rezanja [mm]	Posmak [mm]	Hrapavost [μm]
20 iteracija	7,95	1,4	0,101	3,07
50 iteracija	9,8	1,4	0,10	2,51
100 iteracija	9,8	1,4	0,10	2,51
150 iteracija	9,8	1,4	0,10	2,51

Iz tablice se vidi kako genetski algoritam postiže minimalnu hrapavost u slučaju kada je maksimalna brzina rezanja i minimalan posmak. Minimalna postignuta hrapavost je 2,51 μm.

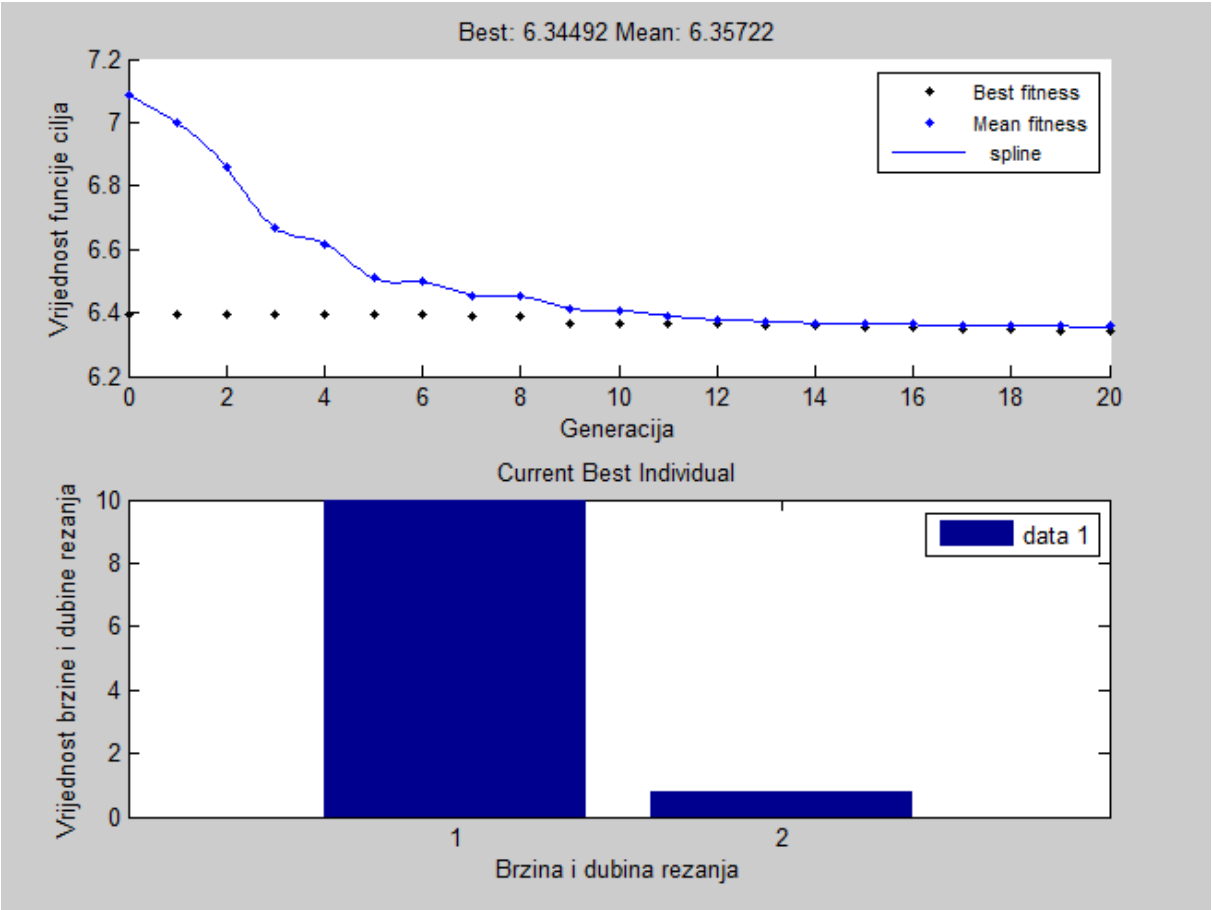
4.2.3 Optimizacija funkcije s konstantnim posmakom

Ovdje se traže optimalni režimi rada za slučaj kada imamo konstantni posmak. Ponovno se traži optimalna brzina rezanja, a umjesto posmaka sada se traži dubina rezanja. Za posmak je odabrana srednja vrijedonst tijekom eksperimentiranja u iznosu 0,2 mm. Brzina rezanja ponovno može poprimiti vrijednosti između 2 i 10 m/s, dok dubina rezanja može biti između 0,8 i 2 mm.

$$R_z = C + p_1 \cdot v + p_2 \cdot a_p + p_{33} \cdot f^2 = \\ = 1{,}919 + (-0{,}1804) \cdot v + 0{,}635 \cdot a_p + 142{,}897 \cdot 0{,}2^2 \quad \textbf{(11)}$$



Slika 33. Dijagram ovisnosti parametra hrapavosti o brzini i dubini rezanja

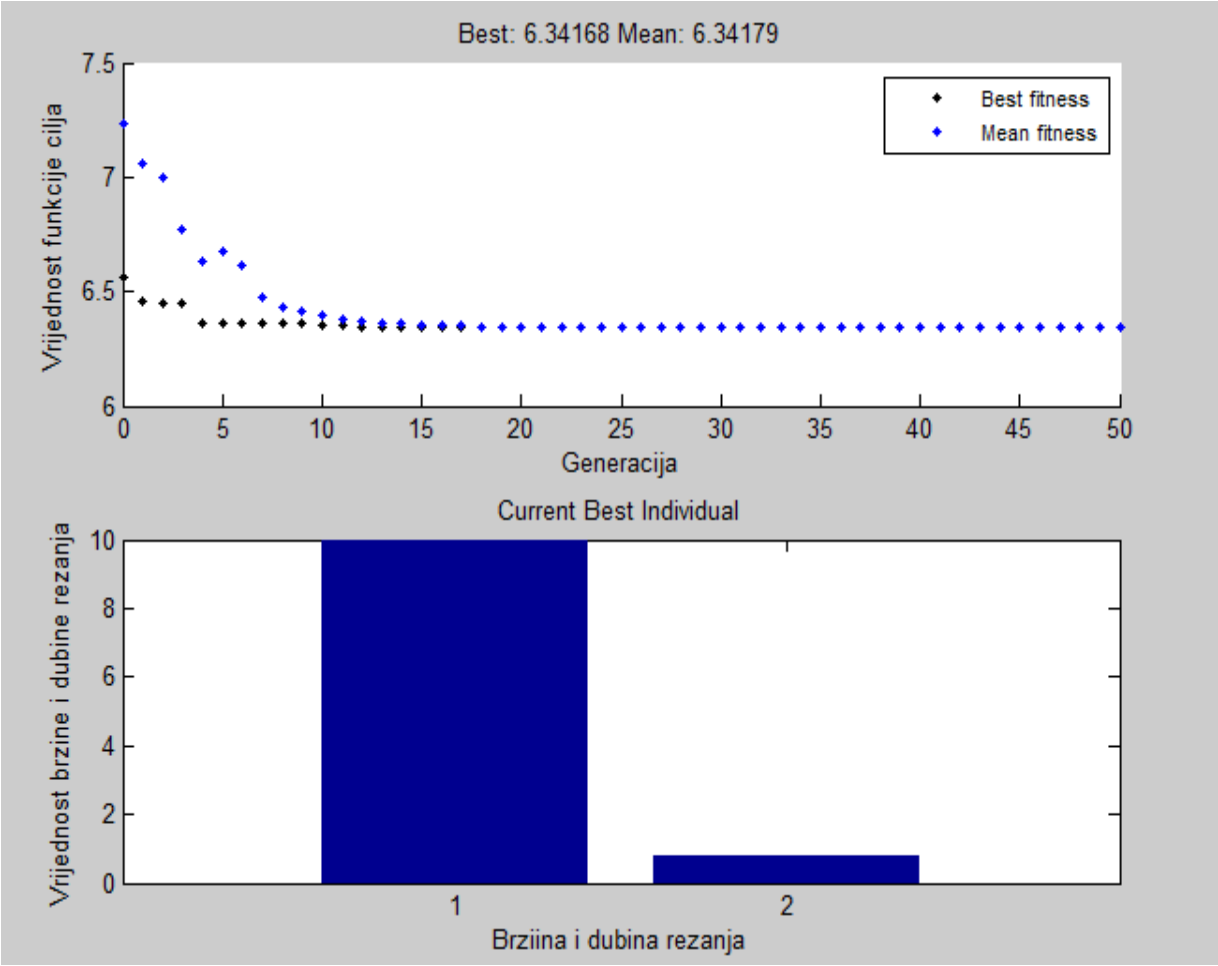


Slika 34. Rezultati optimizacije uz konstantan posmak i 20 generacija

Sa prikazanih grafova vidi se kako minimalnu hrapavost očekujemo za maksimalnu brzinu i minimalnu dubinu rezanja. U slučaju kada odaberemo 20 iteracija sa po 10 kromosoma u svakoj generaciji dobiva se srednja vrijednost parametra hrapavosti od 6,35.

```
function [x,fval,exitflag,output,population,score] = GEN(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@RASTRING,nvars,[],[],[],[],lb,ub,[],[],options);
```



Slika 35. Rezultati optimizacije uz konstantan posmak i 50 generacija

U slučaju kada odaberemo 50 generacija sa po 20 kromosoma dobivamo nešto bolje optimizirane režime rada. Sa svih grafova vidi se kako je za hrapavost najvažniji posmak. Hrapavost je najniža kada uzimamo najpovoljnije režime rada, odnosno minimalni posmak i dubinu rezanja, a maksimalnu brzinu rezanja.

Tablica 13. Usporedba rezulta obzirom na broj generacija uz konstantan posmak

	Brzina rezanja [m/s]	Dubina rezanja [mm]	Posmak [mm]	Hrapavost [μm]
20 iteracija	10	0,8	0,20	6,35
50 iteracija	10	0,8	0,20	6,34
100 iteracija	10	0,8	0,20	6,34
150 iteracija	10	0,8	0,20	6,34

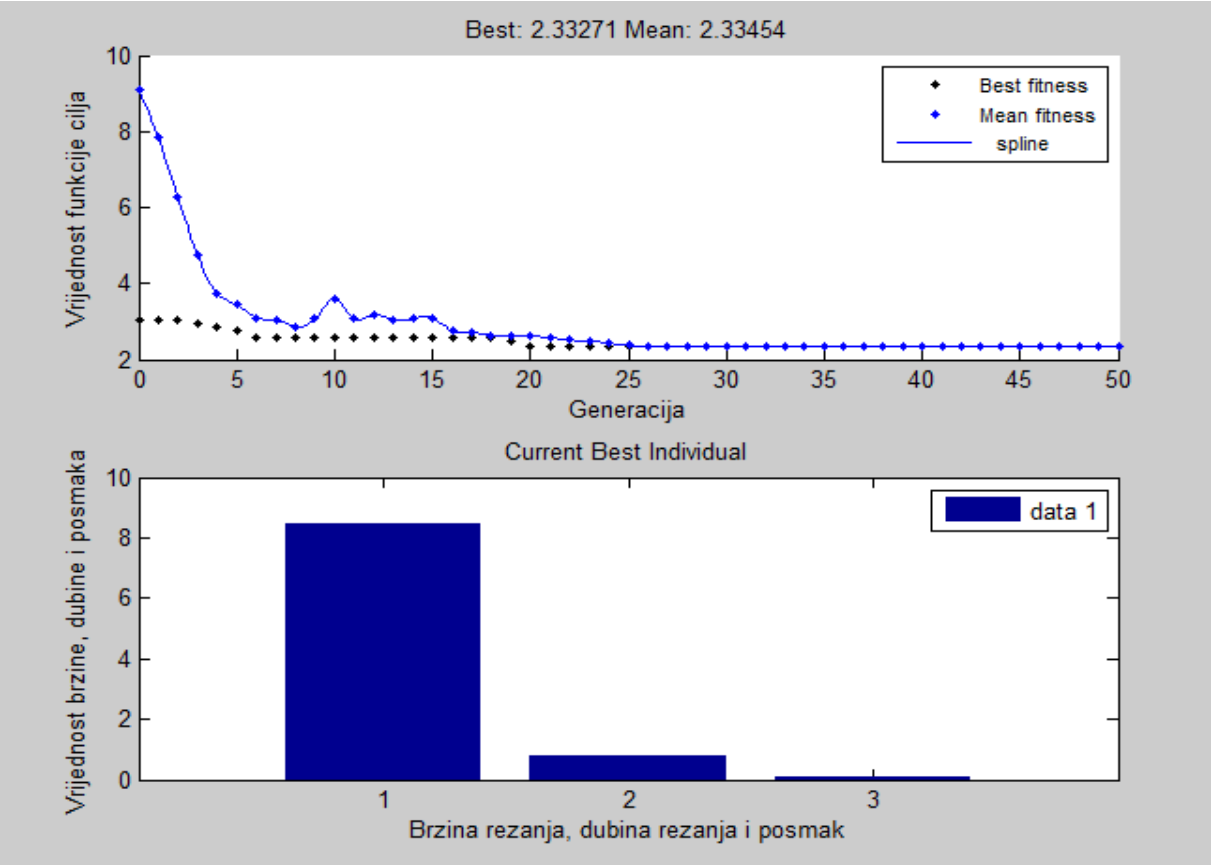
U slučaju konstantnog posmaka, i to u srednjoj vrijednosti u iznosu 0,20 mm, postiže se hrapavost u iznosu 6,34 μm. Iz ovoga se vidi kako najveći utjecaj na hrapavost ima upravo posmak.

```
function [x,fval,exitflag,output,population,score] = GEN(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@RASTRING,nvars,[],[],[],[],lb,ub,[],[],options);
```

4.2.4 Optimizacija funkcije s tri varijable

U slučaju kada imamo tri varijable, a želimo dobiti minimalnu hrapavost, potrebno optimizirati funkciju na odgovarajući način. Ovdje tražimo optimalnu brzinu rezanja, dubinu rezanja i posmak. Svaka varijabla može poprimiti vrijednosti u granicama dobivenim eksperimentalno. Tako brzini rezanja može iznositi između 4 i 9 m/s, posmak između 0,1 i 0,3 mm, a dubina rezanja između 0,8 i 2 mm.



Slika 36. Optimizirane vrijednosti režima rada za zahtjevanu minimalnu hrapavost

Sa grafa se potvrđuje ispitivani problem. Hrapavost će biti minimalna u slučaju da obradu vršimo najvećom brzinom, dok posmak i dubina rezanja trebaju biti minimalni.

Tablica 14. Rezultati optimizacije za zahtjevanu minimalnu hrapavost

	Brzina rezanja [m/s]	Dubina rezanja [mm]	Posmak [mm]	Hrapavost [μm]
50 iteracija	9	0,8	0,10	2,33
100 iteracija	9	0,8	0,10	2,33
150 iteracija	9	0,8	0,10	2,33

U slučaju kada se optimira funkcija sa sve tri varijable postiže se ukupno najmanja hrapavost, i to u iznosu 2,33 μm. Genetski algoritam optimirao je funkciju tako da je dobivena maksimalna brzina rezanja i minimalni posmak i dubina rezanja. Dobivena hrapavost je najmanja moguća za parametre u zadanim granicama, što se vidi po tome što je genetski algoritam dao iste rezultate i za 50, i za 100, i za 150 generacija.

```
function [x,fval,exitflag,output,population,score] = GEN(nvars,lb,ub)
%% This is an auto generated MATLAB file from Optimization Tool.

%% Start with the default options
options = gaoptimset;
%% Modify options setting
options = gaoptimset(options,'Display', 'off');
options = gaoptimset(options,'PlotFcns', {  @gaplotbestf @gaplotbestindiv
});
[x,fval,exitflag,output,population,score] = ...
ga(@HRAPAVOST,nvars,[],[],[],[],lb,ub,[],[],options);
```

5. Zaključak

Genetski algoritmi predstavljaju jednostavan pristup složenim problemima pretraživanja velikog prostora. Korištenjem modela učenja i evolucije iz prirode omogućuju modeliranje željenih svojstava sustava, te automatsko pronalaženje rješenja. U opisanim optimiranim slučajevima pokazano je kako je apstraktne modele genetskog algoritma moguće primjeniti i na stvarnim problemima s različitih područja. Ostvareni rezultati, iako pokazuju zadovoljavajuća rješenja, ne osiguravaju uvijek nalaženje optimalnog rješenja. Prilikom izrade genetskog algoritma potrebno je definirati slijedeće:

- Funkciju dobrote
- Selekciju
- Jedinke
- Operatore
- Uvjet završetka rada
- Parametre
- Veličinu populacije
- Vjerojatnost primjene operatora

Genetski algoritmi imaju razne prednosti i nedostatke koji ih čine manje ili više primjenjivim na razne probleme. Tako se u prednosti može uvrstiti primjenjivost genetskih algoritama na velik broj raznih problema bez matematičkih ograničenja (neprekinutost, derivabilnost i slično). Genetski algoritmi su primjenjivi na velik broj problema, mogu se primjeniti za optimizaciju svake matematičke funkcije. Zatim, genetski algoritmi nude velik stupanj slobode, odnosno, moguće je povećati efikasnost genetskog algoritma raznim jednostavnim zahvatima, poput promjene postavki genetskog algoritma, mogućnosti mutacije i križanja. Jednostavnim ponavljanjem genetskog algoritma može se povećati kvaliteta rješenja. Genetski algoritmi nam uvijek daju dobro rješenje, iako kod kompliciranijih funkcija ponekad ne daju globalni minimum ili maksimum kao konačno rješenje. Pod prednosti se još može uračunati i programska podrška. Postoji velik broj programa koji nam nude optimiranje funkcija pomoću genetskih algoritama.

Nedostatci su također brojni, kao najveći se možda ističe težine određivanja točne matematičke funkcije za zadani problem. Često je također potrebna prilagodba problema genetskom algoritmu, kao što problem predstavlja i postavljanje zadanih ograničenja

genetskog algoritma. Također, genetski algoritmi nam nikad ne nude 100% točna rješenja,a zbog stohastične prirode ne možemo znati točnost rješenja.

Genetski algoritmi predstavljaju područje koje ima mnogo prostora za napredak, ukoliko se uspiju otkloniti trenutni nedostatci. Unatoč svim nabrojanim nedostatcima genetski algoritmi se koriste za razne probleme optimiranja upravo zbog jednostavnosti i sigurnosti u dobre rezultate optimiranja. Tako su za zadanu funkciju hrapavosti dobiveni optimalni parametri, čime se pokazalo da su genetski algoritmi efikasni u rješavanju optimizacijskih problema u slučaju kada je poznata funkcija cilja.

6. Literatura

[1] Zninka, K: Diplomski rad, 2012.

[2] http://www.mathworks.com/help/gads/genetic-algorithm.html - pristupljeno 25.01.2013

[3] Golub, M: Genetski algoritam, Prvi dio, 2010.

[4] Golub, M: Genetski algoritam, Drugi dio, 2010.

[5] Golub, M: Evolucijski algoritmi, 2008.

[6] Negnevitsky, Michael: Artificial intelligence, 2005.

[7] http://www.secotools.com