Marin Golub

Genetski algoritam

Prvi dio

Evolucija u prirodi
Jednostavni genetski algoritam
Genetski operatori
Prikaz brojem s pomičnom točkom
Primjeri

Zadnja značajnija promjena obavljena 27. rujna 2004. godine (verzija 2.3)
Zadnje ispravke u tekstu su načinjene 5.5.2010.
Verzija 1.0 izašla 6.listopada 1997. godine

Zahvaljujem se Domagoju Jakoboviću na pomoći i sugestijama tijekom pripreme ovog teksta.
Domagoj je ujedno i autor poglavlja 3.8 Prilagodljivi genetski algoritam (AGA) i jednog dijela uvoda.

Zahvaljujem se studentu Zoranu Rušinoviću što mi je ukazao na niz grešaka koje su se potkrale u tekstu.
Verzije teksta 2.3 i 2.2 se razlikuju upravu u ispravljenim spomenutim greškama. Zadnja dva odlomka u poglavlju 3.2.2 su rezultat Zoranovih primjedbi.

SADRŽAJ

1. UVOD

U posljednjih petnaestak godina zabilježen je značajan razvoj genetskih algoritama. Genetski algoritam se primjenjuje i daje dobre rezultate u području učenja kod neuronskih mreža, pri traženju najkraćeg puta, problemu trgovačkog putnika, strategiji igara, problemima sličnim transportnom problemu, problemu raspoređivanja procesa, problemu određivanja parametara sustava, optimiranju upita nad bazom podataka, itd.

Genetski algoritam je heuristička metoda optimiranja koja imitira prirodni evolucijski proces. Evolucija je robustan proces pretraživanja prostora rješenja. Živa bića se tijekom evolucije prilagođavaju uvjetima u prirodi, tj. životnoj okolini. Analogija evolucije kao prirodnog procesa i genetskog algoritma kao metode optimiranja, očituje se u procesu selekcije i genetskim operatorima. Mehanizam odabira nad nekom vrstom živih bića u evolucijskom procesu čine okolina i uvjeti u prirodi. U genetskim algoritmima ključ selekcije je funkcija cilja, koja na odgovarajući način predstavlja problem koji se rješava. Slično kao što su okolina i uvjeti u prirodi ključ selekcije nad nekom vrstom živih bića, tako je i funkcija cilja ključ selekcije nad populacijom rješenja u genetskom algoritmu. Naime, u prirodi jedinka koja je najbolje prilagođena uvjetima i okolini u kojoj živi ima najveću vjerojatnost preživljavanja i parenja, a time i prenošenja svojega genetskog materijala na svoje potomke. Za genetski algoritam jedno rješenje je jedna jedinka. Selekcijom se odabiru dobre jedinke koje se prenose u slijedeću populaciju, a manipulacijom genetskog materijala stvaraju se nove jedinke. Takav ciklus selekcije, reprodukcije i manipulacije genetskim materijalom jedinki ponavlja se sve dok nije zadovoljen uvjet zaustavljanja evolucijskog procesa.

Genetski algoritmi predloženi su od strane Johna H. Hollanda još u ranim sedamdesetima. Tijekom nešto više od dva desetljeća, a posebno u posljednjih nekoliko godina, pokazali su se vrlo moćnim i u isto vrijeme općenitim alatom za rješavanje čitavog niza problema iz inžinjerske prakse. To se može objasniti njihovom jednostavnošću; kako same ideje na kojoj su osnovani, tako i njihove primjene; te doprinosu niza znanstvenika i inžinjera na njihovom prilagođavanju velikom broju problema i povećanju efikasnosti. Paralelno s povećanjem primjene povećava se i opseg istraživanja rada i svojstava genetskih algoritama i pokušavaju se svesti njihovi elementi na neke teorijske osnove. Nažalost, rezultati postignuti na teorijskom području su dvojbeni, a genetski algortimi ostaju i do danas u osnovi heurističke metode.

Po načinu djelovanja ubrajaju se u metode usmjerenog slučajnog pretraživanja prostora rješenja (*guided random search techniques*) u potrazi za globalnim optimumom. U istu grupu možemo ubrojiti još neke metode koje se temelje na sličnim principima: to su evolucijske strategije (*evolutionary strategies*), simulirano kaljenje (*simulated annealing*) i genetsko programiranje (*genetic programming*).

Evolucijske strategije, razvijene u Njemačkoj u šezdesetim godinama ovoga stoljeća, imaju puno zajedničkih osobina sa genetskim algoritmima i često im se, kada su u pitanju razne varijante obaju pristupa, teško određuju granice. Obje metode održavaju populaciju rješenja nad kojom provode definirane operacije što se periodički ponavljaju. Zato se faze takvog procesa, po uzoru na prirodne evolucijske tokove, zovu i generacije.

Simulirano kaljenje je proces koji je za svoj uzor upotrijebio termodinamičko kretanje materije prema stanju minimalne energije u postepenom snižavanju temperature kao parametra sustava. Metoda operira na jednom rješenju od koga se u svakoj iteraciji traži "susjedno" rješenje. Staro se uvijek zamjenjuje novim ako je postignut napredak u zadovoljavanju kriterija, a moguće je i da lošije rješenje zamijeni bolje ako se zadovolji određena mjera stohastičnosti koja se regulira sa "temperaturom" sustava. Što je veća temperatura, veća je vjerojatnost da novo, makar i lošije rješenje, zamijeni staro. Proces kreće od neke određene temperature koja dozvoljava relativno veliku vjerojatnost prihvaćanja (veća od 50%), a zatim se taj parametar eksponencijalno smanjuje sve dok kretanje ne postane gotovo determinističko.

Snaga tih metoda, a pogotovo genetskih algoritama, leži u činjenici da su oni sposobni odrediti položaj globalnog optimuma u prostoru s više lokalnh ekstrema, u tzv. višemodalnom prostoru. Klasične determinističke metode će se uvijek kretati prema lokalnom minimumu ili maksimumu, pri čemu on može biti i globalni, ali to se ne može odrediti iz rezultata. Stohastičke metode, tako i genetski algoritmi, nisu ovisne o nekoj eventualnoj početnoj točki i mogu svojim postupkom pretraživanja s nekom vjerojatnošću locirati globalni optimum određene ciljne funkcije. Osnovna razlika u primjeni između klasičnih i stohastičkih metoda je ta što za rezultat neke, recimo, gradijentne metode možemo sa sigurnošću reći da je postignut lokalni ekstrem unutar željene preciznosti. Za rezultat rada genetskog algoritma, međutim, nismo u mogućnosti sa stopostotnom vjerojatnošću reći da li predstavlja globalni ili samo lokalni optimum, te da li je isti određen sa željenom preciznošću. Koliko god se performanse stohastičkih metoda poboljšavale, one nikada neće moći dati niti jedan rezultat sa apsolutnom sigurnošću. Sigurnost dobivenih rezultata značajno se povećava postupkom ponavljanja procesa rješavanja, što kod klasičnih metoda nema smisla. Od kada su genetski algoritmi nastali, velika se pažnja poklanja istraživanjima vezanim za povećanje djelotvornosti izvedbe.

2. PRIRODNI EVOLUCIJSKI PROCESI

2.1 Borba za opstanak

Charles Darwin je primijetio da živa bića u pravilu stvaraju više potomaka od cijele njihove populacije. Prema tome, broj jedinki koje čine populaciju trebao bi eksponencijalno rasti iz generacije u generaciju. Unatoč toj činjenici, broj jedinki jedne vrste teži ka konstantnom broju. Primijetivši različitosti među jedinkama iste vrste, zaključio je da priroda selekcijom jedinki regulira veličinu populacije. Dobra svojstva jedinke, kao što su otpornost na razne bolesti, sposobnost trčanja, itd., pomažu da jedinka preživi u neprestanoj borbi za opstanak.

Evolucija je neprekidan proces prilagođavanja živih bića na svoju okolinu, tj. na uvjete u kojima žive. U prirodi vlada nemilosrdna borba za opstanak u kojoj pobjeđuju najbolji, a loši umiru. Da bi neka vrsta tijekom evolucije opstala, mora se prilagođavati uvjetima i okolini u kojoj živi, jer se i uvjeti i okolina mijenjaju. Svaka slijedeća generacija neke vrste mora pamtiti *dobra* svojstva prethodne generacije, pronalaziti i mijenjati ta svojstva tako da ostanu dobra u neprekidno novim uvjetima.

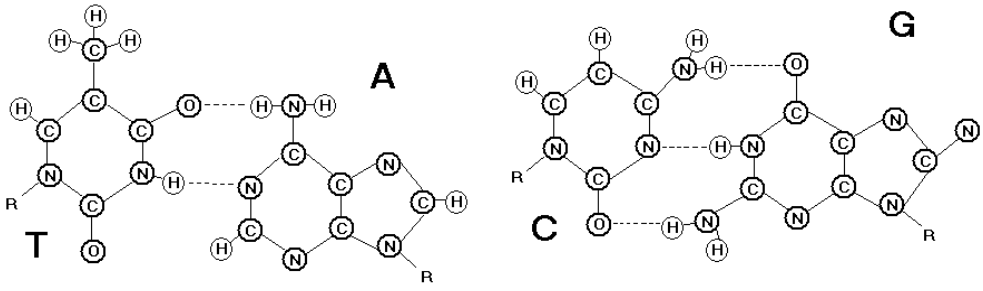
Svaka se jedinka može okarakterizirati nizom svojstava, kao što su npr.: sposobnost trčanja, boja kože, boja očiju, oštrina vida, prilagodljivost na niske temperature, broj zubi, oblik zubi, i sl. Slabe jedinke, odnosno jedinke koje imaju loša svojstva, imaju malu vjerojatnost preživljavanja u borbi za opstanak i one će najvjerojatnije odumrijeti, a zajedno s njima i loša svojstva. Dakle, dobra svojstva imaju veću vjerojatnost nasljeđivanja, odnosno prenošenja na slijedeću generaciju.

Danas se pretpostavlja da su sva svojstva jedinke zapisana u kromosomima. Kromosomi su lančaste tvorevine koje se nalaze u jezgri svake stanice, što znači da svaka stanica biljnog ili životinjskog podrijetla posjeduje sve informacije o svim svojstvima jedinke. Skup informacija koje karakteriziraju jedno svojstvo zapisano je u jedan djelić kromosoma koji se naziva gen. Kromosomi dolaze uvijek u parovima: jedan kromosom je od oca, a drugi od majke. Dakle, za svako svojstvo postoje dva gena ili dvije informacije. Takav par gena gdje jedan i drugi gen nosi informaciju za jedno svojstvo naziva se *alel*. U genetskom paru geni mogu biti ravnopravni ili neravnopravni, tako da je jedan dominantan, a drugi recesivan. U neravnopravnom paru dominantan gen određuje rezultantno svojstvo, dok se uz ravnopravni par gena dobiva svojstvo koje je negdje između svojstava oca i majke.

Sva svojstva jedinke obično nisu zapisana u samo jednom paru kromosoma, već u nekoliko desetaka parova kromosoma. Npr., čovjek ima 46 kromosoma odnosno 23 para kromosoma.

2.2 Molekula DNK kao nositelj informacije

Kemijsku strukturu koja je prisutna u kromosomima otkrili su Watson i Crick 1953. godine. Molekula deoksiribonukleinske kiseline (DNK) je u obliku dvije spirale građene od fosforne kiseline i šećera, a mostovi između spiralnih niti građeni su od dušičnih baza i to adenina (A), gvanina (G), timina (T) i citozina (C). Dušične baze su međusobno povezane vodikovim vezama i to adenin s timinom i gvanin sa citozinom kako je prikazano na slici 2.1.

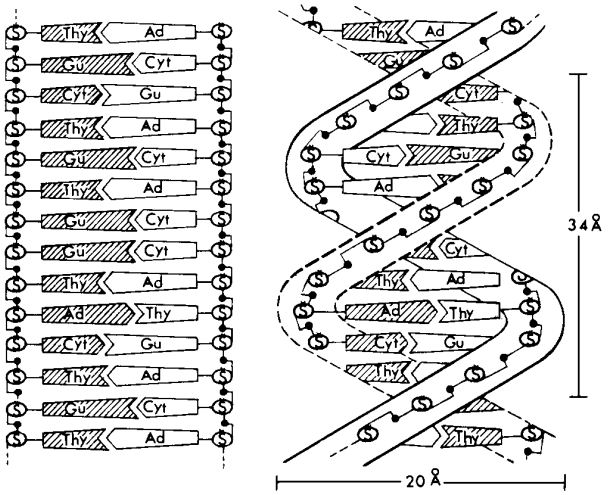


Slika 2.1: Parovi baza u DNK

Pokazalo se da su upravo te dušične baze jedinice informacije. Slično kao što je u računalu najmanja jedinica informacije jedan bit (0 ili 1), tako je u prirodi najmanja jedinica informacija jedna dušična baza (A, G, C ili T). Jedan kromosom se sastoji od dvije komplementarne niti DNK i to ako je, npr. djelić jedne spiralne niti AAGTCA, tada je ekvivalentan dio druge spirale TTCAGT. Ovakva struktura dviju spiralnih niti DNK omogućava prenošenje informacije dijeljenjem kromosoma pri diobi stanice. Naime, kada se stanica treba podijeliti, kromosomske niti se razmotaju. Budući da u jezgri stanice ima mnoštvo slobodnih baza, te baze se vežu na svoje parove na nitima DNK. Tako je informacija sačuvana, kopirana i prenešena na dva novonastala kromosoma.

2.3 Genetski kod

Dvadeset danas poznatih aminokiselina su osnovni građevni dijelovi staničnih struktura, proteina, hormona i enzima. Informacija o strukturi proteina je zapisana u kromosomu, odnosno u molekuli DNK. Genetska šifra je otkrivena u razdoblju 1961. do 1965. godine i predstavlja jedan od najvećih uspjeha molekularne biologije ovog stoljeća.



Slika 2.2: Struktura dvostruke spirale DNK

Genetska šifra sadržana u DNK mora biti na neki način zapisana linearnim redosljedom dušičnih baza duž polinukleotidnog lanca. Dovoljno je promatrati samo redoslijed u jednom lancu, jer je drugi lanac prvome komplementaran. Budući postoji 20 aminokiselina, za jediničnu riječ, tj. informaciju za jednu aminokiselinu, nije dovoljan jedan nukleotid, jer ima samo 4 mogućnosti. Dva nukleotida su također premalo, jer je moguće 4^2=16 kombinacija. Pretpostavka da triplet nukleotida (64 kombinacije) nosi informaciju o vrsti aminokiseline, pokazala se točnom. Rezultati istraživanja koja kombinacija u jednom tripletu predstavlja određenu aminokiselinu prikazani su u tablici. U procesu sinteze proteina sudjeluje i ribonukleinska kiselina (RNK) i to kao prenosilac informacije iz stanične jezgre do mjesta u stanici gdje se obavlja sinteza. RNK je slične građe kao i DNK, samo što umjesto dušične baze timina (T) ima uracil (U), ali uloge kod prijenosa informacije su im iste.

Prema tablici 2.1 niz AUG UCC UAU AUC GUU UAA predstavlja lanac sljedećih aminokiselina: Ser - Tyr - Ile - Val. Genetska šifra AUG predstavlja znak za početak, a UAA za završetak lanca aminokiselina. Na isti način određena je na primjer boja očiju, jer boju očiju određuje jedan proteinski lanac, a informacija o redosljedu proteina, odnosno aminokiselina zapisana je u određenom segmentu DNK.

Prvo slovo	Drugo slovo				Treće slovo
	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met ili START	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Tablica 2.1: Tablica genetske šifre

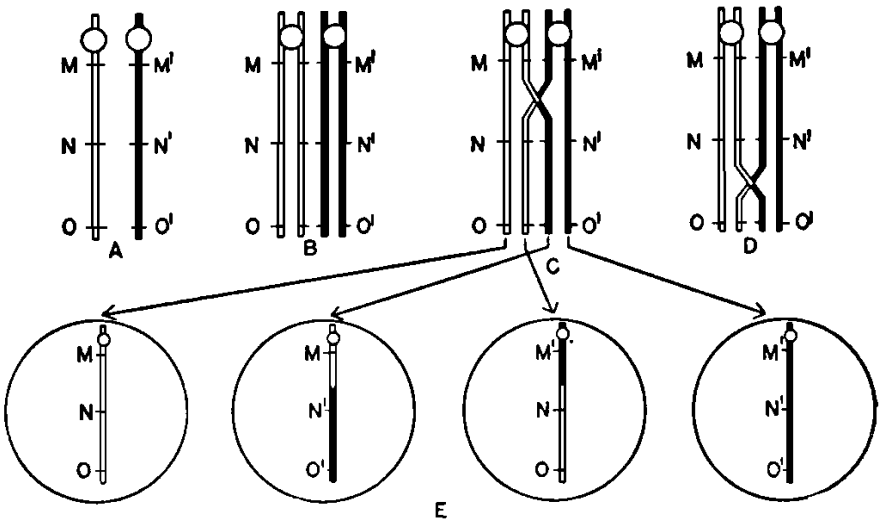
Ala-alanin
Arg-arginin
Asp-asparaginska kiselina
Asn-asparagin
Cys-cistein
Gln-glutamin
Glu-glutaminska kiselina
Gly-glicin
His-histidin
Ile-izoleucin
Leu-leucin
Lys-lizin
Met-metionin
Phe-fenilanin
Pro-prolin
Ser-serin
Thr-treonin
Trp-triptofan
Tyr-tirozin
Val-valin

Jedan gen nosi informaciju za jedno svojstvo i danas se smatra da sadrži oko 1000 nukleotida, točnije nukleotidnih parova, odnosno parova dušičnih baza. Kompletan genetski materijal neke jedinke, npr. žabe, sastoji se od oko $3{,}2*10^9$ nukleotida.

Slika 2.2 prikazuje spiralnu strukturu molekule DNK koja se sastoji od dviju komplementarnih nizova nukleotida. Prikazani spiralni oblik ima DNK kad nije u procesu diobe: mitoze ili mejoze. Samo tada su kromosomi vidljivi običnim mikroskopom. Kad se stanica nalazi u procesu diobe, DNK se "rasplete" u duge kromosomske niti. U obliku niti, DNK je spremna za diobu.

2.4 Križanje i mutacija

Križanjem se prenosi genetski materijal, a s njim i svojstva roditelja na djecu. Svaka jedinka posjeduje genetski materijal oba roditelja. Neka je broj kromosoma koje ima jedinka *2n*. Slijedeća generacija jedinki također treba imati isti broj kromosoma, što znači da jedan roditelj treba dati polovičan (haploidan) broj kromosoma *n*. To se postiže prilikom sinteze spolnih stanica staničnom diobom koja se naziva mejoza. Prilikom mejoze, događa se još jedan značajan proces. Za vrijeme diobe kromosoma, kromosomi se najprije razmotaju u duge niti. Par kromosoma se razdvoji, ali ne u potpunosti, tako da se još drži zajedno u samo jednoj točki koja se naziva centromera. Slijedi udvostručavanje kromosoma; vežu se slobodne baze s komplementarnim bazama na kromosomskim nitima. Kromosomi se još uvijek drže u zajedničkoj centromeri koja se još nije podijelila. Tada dolazi između unutrašnjih niti (rjeđe i vanjskih) do izmjene pojedinih segmenata kromosomskih niti (slika 2.3). Taj proces naziva se križanje ili izmjena faktora. Na učestalost izmjene faktora djeluje temperatura, spol, dob, ishrana, itd.



Slika 2.3: Izmjena faktora (*crossing-over*)

Neka svojstva koja definira ili stekne jedinka u svojem životu (kao što je ožiljak) ne prenose se na potomke.

Mutacija je slučajna promjena gena. Neki geni mutiraju lakše, a neki teže, pa se dijele na stabilne i nestabilne gene. Vjerojatnost mutacije jednog gena je konstantna. No, zanimljivo je da je vjerojatnost mutacije različita za različite gene i kreće se od 10^{-4} do 10^{-5}. Zatim, vjerojatnost da gen A postane gen B nije ista kao i vjerojatnost da se gen B mutacijom promijeni u gen A.

2.5 Genetski algoritam: slika evolucije vrsta

Pojedine vrste živih bića se uz pomoć evolucije prilagođavaju uvijek novim prilikama i uvjetima u prirodi. Dakle, u prirodi evolucija vrste nije potraga za rješenjem (jedinkom koja je najbolje prilagođena uvjetima u prirodi), već prilagođavanje postojeće populacije na nove uvjete.

Genetski algoritam (GA) kao metoda optimiranja nastao je oponašanjem evolucije. Simuliranje prirodnog evolucijskog procesa na računalu svodi se na grube aproksimacije rješenja. Naime, veličina genotipa, npr. žabe je približno $3.2*10^9$ nukleotida, što odgovara količini podataka na računalu od 0.8GB[1] za jednu jedinku. Znači, vjerna simulacija evolucije jedne vrste od milijun jedinki zahtjeva memorijski prostor od približno milijun gigabajta, što je unatoč eksponencijalnom rastu mogućnosti računala, danas, a i još će neko vrijeme biti, nedostižno i neizvodivo.

[1] S pomoću jednog nukleotida, odnosno jedne dušične baze može se spremiti isto toliko informacija koliko i s pomoću dva bita u binarnom prikazu, (4 različite informacije). Dakle, $3.2*10^9$ nukleotida imaju isti kapacitet kao i $6.4*10^9$ bitova, što je ekvivalentno kapacitetu od $6.4*10^9/8=0.8*10^9$ bajtova. To je približno 0.8GB.

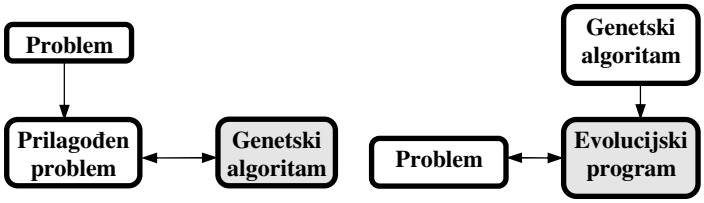
3. JEDNOSTAVNI GENETSKI ALGORITAM

3.1 Opća razmatranja

Evolucija je prirodni proces traženja najbolje i najprilagodljivije jedinke na okolinu i uvjete u prirodi. Dakle, evolucija, odnosno prirodni evolucijski proces, jest metoda optimiranja. Ideja za genetski algoritam može se naći u mehanizmu prirodnog odabira: u prirodi, od skupa istobitnih pojedinaca, preživljavaju oni koji su najbolje prilagođeni snalaženju u okolini siromašnoj sredstvima za život (lijepu usporedbu možemo naći u modernoj ekonomiji). Najsposobniji dobivaju priliku da dominiraju slabijima i također se reproduciraju. Ako poistovjetimo mjeru sposobnosti pojedinca da preživi sa nasljednim materijalom koji nosi u sebi, genima, tada možemo reći da geni dominantnih pojedinaca opstaju dok geni slabijih izumiru jer nemaju potomstva. Pored te pojave, u prirodi pri svakoj reprodukciji dolazi do rekombinacije gena koja uzrokuje različitost među pojedincima iste vrste, ali i sličnosti s roditeljima jedinke. Promatramo li samo gene, možemo reći da smo u svakoj novoj generaciji dobili novi skup gena, gdje su neki pojedinci lošiji od onih u prethodnoj generaciji a neki bolji. Izmjena gena koja nastaje pri reprodukciji se u genetskim algoritmima, iako to nije sasvim u skladu sa biologijom, naziva križanje. Osim križanja, uočava se još jedna pojava, ali u znatno manjem opsegu. Riječ je o slučajnom mijenjanju genetskog materijala koje nastaje pod djelovanjem vanjskih uzroka a naziva se mutacija. Križanje i mutacija se kod genetskih algoritama nazivaju genetskim operatorima, a proces izdvajanja najsposobnijih jedinki unutar svake generacije odabirom (selekcija).

Moguće je identificirati dva pristupa rješavanja određenog problema: prilagoditi problem genetskom algoritmu ili genetski algoritam prilagoditi specifičnostima problema. Ukoliko odaberemo prilagođavanje algoritma problemu, potrebno je modificirati njegov rad tako da rukuje sa veličinama svojstvenim određenom zadatku. Najčešće se tu radi o upotrebi ili definiranju drugačijih struktura podataka i operatora. Za vrlo veliki broj takvih slučajeva razvijeni su naročiti specijalizirani genetski algoritmi, koji se tada obično nazivaju evolucijskim programima. Uglavnom se tada postižu velika povećanja djelotvornosti i takvi proizvodi se mogu naći i na softverskom tržištu, te imaju veliku korisnost u primjeni. Ovakav način je, opet, problemski ovisan i zahtjeva dosta rada na prilagođavanju; ponekad je taj problem ravan onome koji se želi riješiti, a i za svaku novu primjenu potrebna je nova prilagodba. Zato se kompromis postiže pravljenjem evolucijskog programa koji će se moći primijeniti na čitavu klasu problema.

S druge strane, nastoji li se riješiti zadani problem uz pomoć genetskog algoritma, tada je potrebno problem prilagoditi genetskom algoritmu. Moguće rješenje se najčešće prikazuje kao niz bitova (binarni prikaz). Međutim, postoji čitav niz problema za koje je teško ili nemoguće primijeniti binarni prikaz (npr., problem rasporeda). Zatim, uobičajni genetski operatori mogu generirati značajan postotak nemogućih rješenja (npr., za problem najkraćeg puta) koji ne donose nikakva poboljšanja, već samo usporavaju algoritam. Osim uobičajnog binarnog prikaza koriste se razni drugi prikazi (matrice, nizovi realnih brojeva, itd.) i definiraju se usko specijalizirani genetski operatori primjenjivi samo za određeni problem kako bi se izbjegla nemoguća rješenja. Promjenom prikaza i genetskih operatora dobiva se prilagođeni genetski algoritam problemu koji se naziva evolucijski program (slika 3.2).



Slika 3.1: Dva pristupa rješavanju problema s pomoću genetskog algoritma

Holland je u svom radu predložio (jednostavni) genetski algoritam kao računarski proces koji imitira evolucijski proces u prirodi i primjenjuje ga na apstraktne jedinke. Svaki evolucijski program održava populaciju jedinki u nekoj određenoj generaciji. Svaka jedinka predstavlja potencijalno rješenje problema koji se obrađuje; to može biti matematička funkcija, plan rada neke tvornice, put trgovačkog putnika i sl. Svaka je jedinka predstavljena jednakom podatkovnom strukturom (broj, niz, matrica, stablo itd). Te jedinke se nazivaju **kromosomi**. Svakom rješenju se pridjeljuje određena mjera kvalitete koja se u literaturi obično naziva **dobrota**, dok se funkcija koja tu kvalitetu određuje naziva **funkcija cilja** ili **funkcija dobrote**. Tada se iz stare formira nova populacija izdvajajući, po nekom postupku odabira, bolje jedinke iz skupa postojećih. Neki članovi ove nove populacije podvrgnuti su utjecajima genetskih operatora koji iz njih formiraju nove jedinke. Operatori se dijele na unarne, koji stvaraju novu jedinku mijenjajući manji dio genetskog materijala (*mutacijska grupa*) i operatore višeg reda, koji kreiraju nove individue kombinirajući osobine nekoliko jedinki (*grupa križanja*). Nakon nekog broja izvršenih generacija čitav postupak se zaustavlja kada se zadovolji uvjet zaustavljanja, a najbolji član trenutne populacije predstavlja rješenje koje bi trebalo biti sasvim blizu optimuma. Struktura genetskog algoritma prikazana je na slici 3.3.

```
Genetski_algoritam
{
   t = 0
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
        t = t + 1;
        selektiraj P'(t) iz P(t-1);
        križaj jedinke iz P'(t) i djecu spremi u P(t);
        mutiraj jedinke iz P(t);
   }
   ispiši rješenje;
}
```

Slika 3.2 : Struktura genetskog algoritma

Prilikom inicijalizacije generira se početna populacija jedinki. Obično se početna populacija generira slučajnim odabirom rješenja iz domene mada je moguće početnu populaciju generirati uniformno (sve jedinke su iste pa u početku evolucijskog procesa GA nije učinkovit i taj postupak se ne preporuča) ili usaditi početno (inicijalno) rješenje u početnu populaciju dobiveno nekom drugom optimizacijskom metodom. Slijedi proces koji se ponavlja sve dok ne istekne vrijeme ili je zadovoljen neki uvjet (npr. funkcija dobrote 95% jedinki odstupa za manje od ε). Taj proces se sastoji od djelovanja genetskih operatora selekcije, križanja i mutacije nad populacijom jedinki. Tijekom selekcije loše jedinke odumiru, a bolje opstaju te se u slijedećem koraku, križanju, razmnožavaju. Križanjem se prenose svojstva roditelja na djecu. Mutacijom se mijenjaju svojstva jedinke slučajnom promjenom gena. Takvim postupkom se postiže iz generacije u generaciju sve veća i veća prosječna dobrota populacije.

Genetski algoritam simulira prirodni evolucijski proces. Za evolucijski proces kao i za genetski algoritam se može ustanoviti sljedeće:

- postoji populacija jedinki;
- neke jedinke su *bolje* (bolje su prilagođene okolini);
- *bolje* jedinke imaju veću vjerojatnost preživljavanja i reprodukcije;
- svojstva jedinki zapisana su u kromosomima s pomoću genetskog koda;
- djeca nasljeđuju svojstva roditelja;
- nad jedinkom može djelovati mutacija.

Za genetski algoritam jedinke su potencijalna rješenja, a okolinu predstavlja funkcija cilja. Želi li se implementirati genetski algoritam kao metoda optimiranja, potrebno je definirati proces dekodiranja i preslikavanja podataka zapisanih u kromosomu te odrediti funkciju dobrote.

Neka je problem slijedeći: uz pomoć genetskog algoritma treba pronaći optimum (minimum ili maksimum) funkcije $f(x)$ za $x \in [dg, gg]$. Funkcija je definirana za svaki x na zadanom intervalu. Dovoljno je definirati postupak za $max\{f(x)\}$, jer ako se traži $min\{f(x)\}$, ekvivalentan je problem riješiti $max\{-f(x)\}$. Valja naglasiti da se nad funkcijom $f(x)$ ne postavljaju nikakva dodatna ograničenja. Primjerice nije bitno da li funkcija ima prvu, drugu ili n-tu derivaciju, niti se ne zahtjeva da je neprekidna i sl. Dakle, $f(x)$ je potpuno proizvoljna.

Jednostavni genetski algoritam koristi **binarni prikaz**, **jednostavnu seleciju**, **križanje s jednom točkom prekida** i **jednostavnu mutaciju**. Binarni prikaz i spomenuti genetski operatori opisani su u poglavljima koja slijede.

3.2 Prikaz ili predstavljanje rješenja

Svi podaci koji obilježavaju jednu jedinku zapisani su u jednom kromosomu. Na primjer, neka je problem jednodimenzijski, odnosno traži se globalni optimum funkcije cilja jedne varijable x tako da vrijedi: $x \in [dg,gg]$ gdje su $dg,gg \in \Re$.. Tada jedna jedinka, odnosno jedan kromosom predstavlja jedno rješenje $x \in [dg,gg]$. Podaci (u ovom slučaju je to jedan realan broj) mogu biti pohranjeni na razne načine u jedan kromosom. Na već uobičajan način i radi lakše prezentacije algoritma, neka su podaci spremljeni u niz bitova koji čine jedan kromosom. Ovakav prikaz (reprezentacija) naziva se binarni prikaz i opisan je u ovom poglavlju. Nadalje, jedan kromosom može biti i realan broj (u programskom jeziku *C* to je broj s pomičnom točkom jednostruke ili dvostruke preciznosti). Ako je problem višedimenzijski, umjesto jednog broja koji je zapisan u kromosomu, treba biti polje brojeva veličine dimenzije problema.

Problemi mogu biti i druge prirode: npr. problem najkraćeg puta te problem rasporeda. U tom slučaju potencijalno rješenje zapisano u kromosomu je *put* ili *raspored*. U prvom slučaju kromosom može biti polje cijelih brojeva u kojem svaki broj označava redni broj mjesta, a niz brojeva označavaju put od početne točke do cilja. U drugom slučaju kromosom je dvodimenzijsko polje.

Općenito, kromosom može biti bilo kakva struktura podataka koja opisuje svojstva jedne jedinke. Za genetski algoritam je značajno da kromosom predstavlja moguće rješenje zadanog problema. Za svaku strukturu podataka valja definirati genetske operatore. Međutim, genetski operatori trebaju biti tako definirani da oni ne stvaraju nove jedinke koje predstavljaju nemoguća rješenja, jer se time znatno umanjuje učinkovitost genetskog algoritma.

3.2.1 Prikaz rješenja s pomoću prirodnog binarnog koda (binarni prikaz)

Prikaz rješenja može bitno utjecati na učinkovitost genetskog algoritma, stoga je izbor prikaza izuzetno značajan. Većina teorije vezane uz genetske algoritme je vezana upravo uz binarni prikaz. U praksi se pokazalo da binarni prikaz daje najbolje rezultate u većini primjera gdje se on može iskoristiti.

Kromosom kao binarni vektor predstavlja kodiranu vrijednost $x \in [dg, gg]$. Dužina n binarnog broja utječe na preciznost i označava broj bitova, odnosno broj jedinica ili nula u jednom kromosomu. U takav vektor je moguće zapisati 2^n različitih kombinacija nula i jedinica, tj. moguće je zapisati bilo koji broj u intervalu $[0, 2^n-1]$. Binarni vektor $v(0)=[000....0]$ predstavlja vrijednost $x=dg$, a vektor $v(2^n-1)=[111....1]$ predstavlja vrijednost $x=gg$. Općenito, ako je binarni broj $b \in [0, 2^n-1]$ zapisan kao binarni vektor $v(b)=[B_{n-1}B_{n-2}.....B_1B_0]$, gdje je $B_i=0$ ili 1, tada vrijedi:

binarni broj: ekvivalentan realan broj:

$$b = \sum_{i=0}^{n-1} B_i 2^i \qquad x = dg + \frac{b}{\left(2^n - 1\right)}\left(gg - dg\right) \qquad (3.1), (3.2)$$

Dekodiranje je proces pretvaranja binarnog broja u potencijalno rješenje. U ovom slučaju potencijalno rješenje je bilo koji realan broj x u intervalu $[dg, gg]$. Binarni vektor $v(b)$ predstavlja vrijednost x koja se izračunava prema formuli (3.2). Kodiranje, odnosno određivanje binarnog broja b za zadani realan broj x obavlja se prema formuli:

$$b = \frac{x - dg}{gg - dg}\left(2^n - 1\right). \qquad (3.3)$$

Neka je rješenje x preciznosti p i neka je $p \in N$. To znači da x odstupa od točnog rješenja za maksimalno 10^{-p}. Odnosno, rješenje x je točno na p decimala. Rješenje x je preciznosti p ako je zadovoljena nejednakost (3.4). Za zadanu preciznost p, kromosom mora biti duljine n:

$$\left(gg - dg\right) * 10^p < 2^n - 1, \qquad n \geq \frac{\log\left[(gg - dg) * 10^p + 1\right]}{\log 2}. \qquad (3.4), (3.5)$$

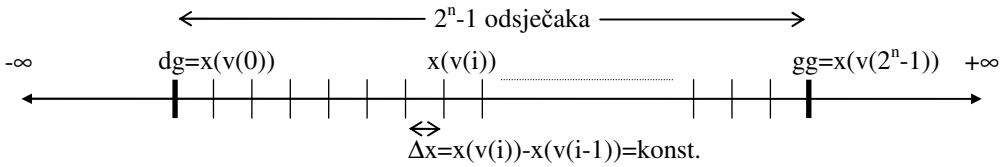
Primjer: Neka se kromosom sastoji od 4 okteta ili 32 bita. Zatim, neka je funkcija cilja realne varijable $f(x)$ definirana na intervalu $[dg, gg]$ duljine $d=gg-dg=10$. Uz zadane parametre zadovoljena je preciznost na 8 decimala, jer je razlika vrijednosti dvaju "susjednih" kromosoma $\Delta x=2{,}328*10^{-9}$ prema formuli (3.6).

Interval $[dg, gg]$ podijeljen je na 2^n-1 jednakih odsječaka kako je prikazano na slici 3.4. Kromosom može predstavljati jedan od 2^n vrijednosti: dg, $dg+\Delta x$, $dg+2\Delta x$, ..., $dg+(2^n-1)\Delta x$, $dg+2^n\Delta x=gg$.

preciznost na p decimala	Δx	*gg -dg*			
		1	10	100	1000
1	10^{-1}	4	7	10	14
2	10^{-2}	7	10	14	17
3	10^{-3}	10	14	17	20
4	10^{-4}	14	17	20	24
6	10^{-6}	20	24	27	30
9	10^{-9}	30	34	37	40
12	10^{-12}	40	44	47	50
15	10^{-15}	50	54	57	60
30	10^{-30}	100	103	107	110

Tablica 3.1: Dužina kromosoma u bitovima ako je zadana preciznost p i dužina intervala: gg-dg

$$\Delta x = x(v(i)) - x(v(i-1)) = \frac{1}{2^n - 1}\left(gg - dg\right), \quad \forall i \in [1, 2^n\text{-}1,]. \qquad (3.6)$$



Slika 3.3 : Kromosom $v(i)$ kao jedna od 2^n točaka jednoliko raspoređenih u intervalu $[dg, gg]$

3.2.2 Prikaz Grayevim kodom

Binarno kodiranje je zbog svoje jednostavnosti pogodno za implementaciju. Međutim, binarno kodiranje ima i jedan veliki nedostatak: Hammingova udaljenost među susjednim brojevima može biti velika, u najgorem slučaju jednaka duljini binarnog zapisa. Hammingova udaljenost između dva binarna broja je broj bitova u kojima se ta dva broja razlikuju. Na primjer, Hammingova udaljenost između brojeva $255_{10}=011111111_2$ i $256_{10}=100000000_2$ iznosi 9 (svih devet bitova je različito). Drugim riječima, ako je genetski algoritam u nekom koraku pronašao jedno *dobro* rješenje za $b=011111111_2$, a optimum se postiže za $b=100000000_2$ treba promijeniti svih devet bitova. Kako bi se ispravio taj nedostatak, za kodiranje brojeva koristi se i Grayev kod.

```
procedura Binarni_u_Grayev_kod{
    gm=bm;
    za k=m-1 do 1
        gk=bk+1 XOR bk
}
```

```
procedura Grayev_u_binarni_kod{
   v=gm;
   bm=gm;
   za k=m-1 do 1 {
      ako je (gk=1) v=1-v;
      bk=v;
   }
}
```

Slika 3.4: Procedure za konverziju brojeva iz binarnog koda u Gray-ev kod i obrnuto

Susjedni brojevi kodirani u Grayevom kodu razlikuju se samo u jednom bitu. Dakle, Hammingova udaljenost između susjednih kodnih riječi je 1. Algoritam transformacije binarnog broja $b= b_m b_{m-1}... b_2 b_1$ u Grayev kod $g=g_m g_{m-1}...g_2 g_1$ i obrnuto, iz Grayevog koda u binarni glasi:

$$g_m = b_m. \qquad g_k = b_k \oplus b_{k+1},\ k=1,2,...,m-1. \qquad b_k = \sum_{k=j}^{m} g_k \pmod 2,\ j=1,2,...,m. \qquad (3.7), (3.8), (3.9)$$

Grayev kod generiran po navedenim forumulama naziva se binarno reflektiran Grayev kod i često se implicitno podrazumjeva kada se govori o Grayevom kodu. Kod se naziva *reflektiran* zato što ga je moguće generirati na sljedeći način: Počnimo od Grayevog koda 0, 1. Zatim ga zapišemo u oba smjera: 0,1,1,0 (dakle prvo 0->1 pa 1->0). Nakon toga prvoj polovici dodamo 0, a drugoj polovici 1: (00, 01, 11, 10). Tako je dobiven 2- bitni Grayev kod. Za 3- bitni kod treba ponovno napisati 2-bitni kod u oba smjera – 00, 01, 11, 10, 10, 11, 01, 00 i ponovno dodati prvoj polovici nule a drugoj jedinice: (000, 001, 011, 010, 110, 111, 101, 100). Na ovaj način u svakoj iteraciji se udvostručuje broj bitova. Neki drugi 3-bitni Grayev kod glasi primjerice (000, 010, 011, 001, 101, 111, 110, 100).

Binarno reflektiran Grayev kod je ciklički kod što znači da se prvi i posljednji broj u nizu također razlikuju za samo jedan bit. Ciklički kodovi su u teoriji povoljni za primjene u GA jer će većina jednobitnih mutacija izazvati samo male promjene (*hill climbing effect*), ali će određen broj mutacija, kao što je jednobitna mutacija koja mijenja zadnji broj u prvi, izazvati jako velike promjene i na taj način omogućiti pretraživanje potpuno novog prostora rješenja (kod običnog binarnog prikaza jednobitna mutacija može promjeniti broj za najviše 2^(n-1) - promjenom MSB bita). Eksperimentalni rezultati, međutim, pokazuju da u praksi bitne razlike između ta dva prikaza nema i da je bolji efekt moguće postići primjenom složenijih operatora mutacija.

3.3 Inicijalizacija i uvjet završetka evolucijskog procesa

Populaciju potencijalnih rješenja predstavlja *VEL_POP(t)* binarnih vektora, gdje je *t* vrijeme ili redni broj generacije. Obično se uzima da je *VEL_POP(t)=konst=VEL_POP*, odnosno veličina populacije se ne mijenja tijekom evolucije. Vrijednost *VEL_POP* je parametar algoritma.

Proces inicijalizacije, odnosno generiranja početne populacije *P(0)*, je jednostavan: generira se *VEL_POP* slučajnih brojeva u intervalu *[0, 2^n-1]*, tj. binarnih vektora dužine *n* bitova. Kao što je već spomenuto, početna populacija može biti uniformna (sve jedinke su iste). Dodatno, u početnu populaciju se može dodati neko rješenje (međurješenje) dobiveno nekom drugom optimizacijskom metodom.

Evolucijski proces se neprestano ponavlja sve dok se ne zadovolji neki *uvjet*. *Uvjet završetka evolucijskog procesa* je najčešće unaprijed zadan broj iteracija. Budući je vrijeme izvođenja pojedine iteracije približno konstantno, time je zadano i trajanje optimiranja.

3.4 Funkcija dobrote

Funkcija dobrote ili funkcija ocjene kvalitete jedinke se u literaturi još naziva *fitness* funkcija, funkcija sposobnosti, funkcija cilja ili *eval* funkcija i u najjednostavnijoj interpretaciji ekvivalent je funkciji *f* koju treba optimizirati:

$$dobrota(v)=f(x), \qquad (3.10)$$

gdje binarni vektor *v* predstavlja realan broj $x \in [dg, gg]$. Što je *dobrota* jedinke veća, jedinka ima veću vjerojatnost preživljavanja i križanja. Funkcija dobrote je ključ za proces selekcije. Pokazalo se u praksi da se ova jednostavna transformacija može primijeniti samo uz neka ograničenja nad *f(x)*. Međutim, najopćenitije bi bilo da *f(x)* nema nikakva ograničenja kako je i bilo rečeno u uvodu ovog poglavlja. Stoga postoji više načina definiranja funkcije dobrote[2] kako je opisano poglavlju 3.5. Za zadani optimizacijski problem najveću poteškoću predstavlja definiranje funkcije dobrote, koja treba vjerno odražavati problem koji se rješava.

[2] Najgora funkcija dobrote je golf igralište.

Tijekom procesa evolucije "dobar" genetski algoritam generira, iz generacije u generaciju, populacije čija je ukupna dobrota i prosječna dobrota sve bolja i bolja. Ukupna dobrota populacije D i prosječna dobrota populacije $\overline{D}$ su definirani formulama:

$$D = \sum_{i=1}^{VEL\_POP} dobrota(v_i), \qquad \overline{D} = \frac{D}{VEL\_POP}. \qquad (3.11), (3.12)$$

3.5 Postupci selekcije

3.5.1 Uloga selekcije

Svrha selekcije je čuvanje i prenošenje dobrih svojstava na slijedeću generaciju jedinki. Selekcijom se odabiru *dobre* jedinke koje će sudjelovati u slijedećem koraku, u reprodukciji. Na taj način se *dobri geni* ili *dobri genetski materijal* sačuvaju i prenose na slijedeću populaciju, a *loši* odumiru. Postupak selekcije bi se mogao ostvariti sortiranjem i odabirom *VEL_POP-M* najboljih jedinki (*VEL_POP* je veličina populacije, a *M* je broj jedinki određenih za eliminaciju). Međutim, takav postupak dovodi do prerane konvergencije genetskog algoritma, tj. proces optimiranja se praktično završava u svega nekoliko prvih iteracija. Problem je u tome što se ovim postupkom *izgubi* dobar genetski materijal koji mogu sadržavati *loše* jedinke. Zato je potrebno osigurati i lošim jedinkama da imaju neku (manju) vjerojatnost preživljavanja. S druge strane, bolje jedinke trebaju imati veću vjerojatnost opstanka, tj. trebaju imati veću vjerojatnost sudjelovanja u procesu reprodukcije.

Genetske algoritme, s obzirom na vrstu selekcije, dijelimo na generacijske i eliminacijske. Generacijski genetski algoritam u jednoj iteraciji raspolaže s dvije populacije (što je ujedno i nedostatak generacijskog GA), jer se odabiru dobre jedinke iz stare populacije koje čine novu populaciju i nakon selekcije sudjeluju u procesu reprodukcije. Karakteristične vrste selekcija koje koristi generacijski GA su: jednostavna selekcija i turnirska selekcija. S druge strane eliminacijska selekcija je karakteristika eliminacijskog genetskog algoritma (*GA with steady-state reproduction*).

3.5.2 Jednostavna selekcija

Genetski algoritam koji koristi jednostavnu selekciju naziva se generacijski genetski algoritam. Jednostavna selekcija (*roulette wheel parent selection*) generira novu populaciju *P'(t)* iz *P(t-1)* koja ima jednak broj jedinki kao i populacija *P(t-1)*, odnosno *VEL_POP(P'(t))=VEL_POP(P(t-1))*. Cilj jednostavne selekcije je odabir roditelja čija je vjerojatnost selekcije proporcionalna njihovoj dobroti. Postupak je slijedeći:

- ako f(x) poprima negativne vrijednosti, tada se dodaje pozitivna konstanta *C*:

$$dobrota(v)=f(x)+C, \text{ tako da je } dobrota(v)\geq 0,\ \forall x\in[dg,\ gg]; \qquad (3.13)$$

- izračunaju se sve vrijednosti $dobrota(v_i)$;
- izračuna se ukupna dobrota populacije prema formuli (3.11);
- izračunaju se kumulativne dobrote q_k za svaki kromosom prema formuli (3.14) tako da vjerojatnost selekcije p_k za svaki kromosom v_k iznosi p_k;

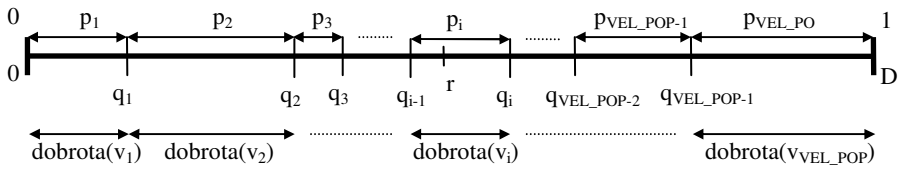
$$q_k = \sum_{i=1}^{k} dobrota(v_i), \text{ gdje je } k=1,2,...VEL\_POP, \qquad p_k = \frac{dobrota(v_k)}{D} \qquad (3.14), (3.15)$$

Time se dobiva da je vjerojatnost selekcije proporcionalna dobroti kromosoma:

$$p_k \approx dobrota(v_k). \qquad (3.16)$$

Lako je pokazati da vrijedi $\sum p_i = 1$, jer je $\sum dobrota(v_i) = D$.

- generira se slučajni realan broj r u intervalu $(0,D)$ i potraži se i-ti kromosom za koji vrijedi da je $r\in(q_{i-1}, q_i)$ i prenosi se u slijedeću populaciju.



Slika 3.5 : Kumulativna dobrota q_i i vjerojatnost selekcije p_i

Ovim postupkom selekcije jedan kromosom se može pojaviti više puta u slijedećoj populaciji. Što je jedinka bolja, to je veća vjerojatnost da bude odabrana u svakom od *VEL_POP* slučajnih izvlačenja. Ako je $dobrota(v_i)=2*\overline{D}$ tada će se v_i najvjerojatnije pojaviti dva puta u slijedećoj generaciji. Takvim postupkom se "teoretski" može dogoditi da u slijedećoj populaciji, prije križanja i mutiranja, bude samo jedna jedinka u *VEL_POP* primjeraka.

Nedostaci jednostavne selekcije

1. Funkcija $f(x)$ ne smije poprimati negativne vrijednosti ni za koji x_i, jer bi tada $dobrota(v(x_i))$ bila negativna ($i=1,2,...,VEL\_POP$). Budući da je vjerojatnost odabira jedinke proporcionalna s dobrotom, negativna vrijednost dobrote nema smisla.
2. Problem pod 1. se može riješiti transformacijom funkcije $f(x)$ u $g(x)=f(x)+M$, gdje je $M>>$. Na primjer: $f(x)=sin(x)$, a $M=1000$. Funkcija dobrote za sve kromosome će biti približno jednaka i iznosit će ≈1000. Ili općenito: neka je $f(x)$ poprima velike vrijednosti za svaki x. To znači da će prilikom selekcije svi kromosomi imati približno jednaku vjerojatnost pojavljivanja u slijedećoj populaciji: $p_i \approx 1/VEL\_POP$. Dakle, svi su podjednako i dobri i loši, što je sve skupa - loše!
3. Pojavljivanje duplikata kromosoma u slijedećoj generaciji se pokazalo kao veliki nedostatak. Eksperimenti su pokazali da čak do 50% od ukupnog broja kromosoma mogu biti duplikati, što samo usporava algoritam.

3.5.2.1 *Translacija ili pomak*

Problemi pod 1. i 2. se mogu riješiti na prvi pogled jednostavno: dodavanjem funkciji cilja $f(x)$ konstante koja je jednaka minimumu zadane funkcije u zadanom intervalu:

$$dobrota(v(x))=f(x)-min\{f(x)\}. \qquad (3.17)$$

Problem pronaći minimum funkcije $f(x)$ je ekvivalentan polaznom problemu (pronaći optimum zadane funkcije cilja). Stoga funkcija dobrote definirana prema formuli (3.17) nema smisla. Međutim, ideja je u redu i valja je samo na drugi način iskoristiti. Nije potrebno znati $min\{f(x)\}$ za svaki $x \in [dg, gg]$, dovoljno je znati najmanju vrijednost funkcije cilja za sve kromosome:

$$dobrota(v(x))=f(x)-min\{f(x_i)\}, \text{ za } i=1,2,...,VEL\_POP. \qquad (3.18)$$

Vrijednost $min\{f(x_i)\}$ nije ista za svaku generaciju i ona se mora u svakom koraku iznova računati. To ne znači usporavanje algoritma novom potragom za najlošijom jedinkom, jer se ionako provodi pretraga za najboljom jedinkom koja se čuva i prenosi u slijedeću generaciju radi poboljšavanja učinkovitosti algoritma. Dakle, identifikacija najbolje i najlošije jedinke se obavlja u jednom koraku pretrage.

Opisani postupak naziva se translacija ili pomak funkcije cilja (*windowing*). Na taj način funkcija dobrote niti može poprimiti negativne vrijednosti, niti se translacijom dobivaju približno iste vrijednosti.

3.5.2.2 *Linearna normalizacija ili sortiranje*

Ako funkcija f poprima negativne ili prevelike vrijednosti, alternativno rješenje istog problema je da se sortiraju vrijednosti $f(x_i)$. Kromosom s najmanjom vrijednošću $f(x_i)$ ima vrijednost dobrote *MIN_D*, slijedeći po veličini *MIN_D+korak*, itd. Veličina koraka je parametar algoritma. Postupak linearne normalizacije svodi se na sortiranje jedinki po njihovoj dobroti.

	dobrota(v(x))					
f(x)	200	105	104	100	99	90
Translacija	110	15	14	10	9	0
Linearna normalizacija s *korakom 5* i *MIN_D=10*	35	30	25	20	15	10
Linearna normalizacija s *korakom 1* i *MIN_D=1*	6	5	4	3	2	1

Tablica 3.2 : Primjer translacije i linearne normalizacije

3.5.3 Turnirska selekcija

Genetski algoritam s ugrađenom generacijskom turnirskom selekcijom u svakom koraku generira novu populaciju iz stare tako da *VEL_POP* puta odabire s jednakom vjerojatnošću k jedinki iz stare populacije, uspoređuje ih i najbolju jedinku kopira u bazen za reprodukciju nad kojim će u slijedećem koraku djelovati genetski operatori. Eliminacijska turnirska selekcija također nasumice odabire k jedinki, ali eliminira najlošiju i nadomješta je s djetetom dviju (slučajno odabranih) preživjelih jedinki. Više o turnirskim i ostalim vrstama selekcija u drugom dijelu.

3.5.4 Eliminacijska selekcija

Generacijski genetski algoritam u jednom koraku raspolaže s dvije populacije jedinki: jednu populaciju dobiva iz prethodnog koraka, a drugu generira jednostavna selekcija. Kad selekcija generira novu populaciju iz prethodne, prethodna populacija se briše. Tek nakon što genetski operatori izvrše svoje

Algoritam eliminacijske selekcije
- **Selektiraj *m* "loših" kromosoma**
- **Izbriši selektirane kromosome**
- **Genetskim operatorima generiraj nove kromosome**
- **Dodaj nove kromosome**

djelovanje, novonastala populacija je spremna za idući korak. To nepotrebno udvostručavanje populacije izbjegava se eliminacijskom selekcijom, odnosno eliminacijskom reprodukcijom.

Za razliku od jednostavne selekcije, eliminacijska selekcija (*steady-state selection*) ne bira *dobre* kromosome za slijedeću populaciju, već *loše* koje treba eliminirati i reprodukcijom ih zamijeniti novima. Dakle, *loši* kromosomi umiru, a njih nadomještaju djeca nastala reprodukcijom roditelja, tj. preživjelih kromosoma. Postoji mogućnost da prilikom križanja i mutacije nastaju djeca koja su identična nekoj već postojećoj jedinki. Budući da duplikati samo usporavaju genetski algoritam, neki autori predlažu genetski algoritam bez duplikata. Poslije svake reprodukcije valja provjeriti da li novonastali kromosom već postoji ili ne. Ako postoji, ponavlja se postupak sve dok se genetskim operatorima ne generira jedinka koja nema duplikata. Takva selekcija se naziva eliminacijska selekcija bez duplikata. Na taj način se eliminira nedostatak naveden pod točkom 3.

```
Eliminacijski genetski algoritam{
   generiraj početnu populaciju;
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa{
      izračunaj vjerojatnost eliminacije za svaku jedinku;
      M puta jednostavnom selekcijom odaberi i izbriši jedinku;
      parenjem preživjelih jedinki nadopuni populaciju;
   }
}
```

Slika 3.6: Genetski algoritam s eliminacijskom reprodukcijom

Algoritam selekcije loših kromosoma je sličan jednostavnoj selekciji, ali ima dvije bitne razlike:

1. Umjesto da je vjerojatnost odabira ili selekcije proporcionalna funkciji dobrote, vjerojatnost selekcije je to veća što je dobrota manja. Umjesto funkcije dobrote treba definirati *funkciju kazne*.

$$p_k \approx kazna(v_k)\,,\ k=1,2,...,VEL\_POP, \qquad (3.19)$$

$$kazna(v_k) = \max_i \{dobrota(v_i)\} - dobrota(v_k)\,., \qquad (3.20)$$

   Time je jednostavno riješen problem očuvanja kromosoma s najvećom dobrotom (opisan u poglavlju 3.5.5), jer je njegova *kazna* jednaka nuli. Stoga je i vjerojatnost eliminacije najbolje jedinke jednaka nuli. Najbolja jedinka se čuva (elitizam) kako bi se iz generacije u generaciju osiguralo da najbolja jedinka bude u najgorem slučaju jednaka kao u prošloj generaciji, ako ne i bolja. Ako se u algoritmu primijeni formula (3.20) za *funkciju kazne*, tada više nije potrebno ugrađivati dodatne mehanizme za očuvanje najbolje jedinke tijekom selekcije, jer najbolja jedinka ima *kaznu* jednaku nuli.
2. Jednom odabrani kromosom za eliminaciju se više ne može odabrati. U svakom koraku potrebno je izračunavati kumulativnu dobrotu i ukupnu dobrotu populacije, jer jednom odabrani kromosom za eliminaciju ne može više doprinositi svojom dobrotom kumulativnoj dobroti.

Osim navedenih prednosti eliminacijske selekcije bez duplikata, algoritam je jednostavan za implementaciju. Tekst programa (programski kod) je kratak i pregledan. Program je brz i daje vrlo dobre rezultate.

3.5.5 Elitizam

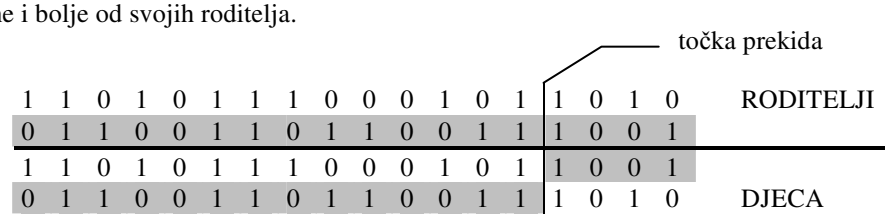
Postoji opasnost da se *dobro* rješenje dobiveno nakon puno iteracija izgubi ukoliko ga genetski operatori (npr. mutacija ili selekcija) izmijene. Stoga se javlja potreba za mehanizmom zaštite najbolje jedinke od bilo kakve izmjene ili eliminacije tijekom evolucijskog procesa. Takav mehanizam se naziva e*litizam.* Genetski algoritam s ugrađenim elitizmom, iz generacije u generaciju, asimptotski teži ka globalnom optimumu, odnosno rješenju problema. Međutim, da bi u svakom koraku evolucije zaštitili najbolju jedinku od bilo kakvih izmjena ili eliminacije, potrebno ju je u svakom koraku i pronaći. Pretraživanje ili sortiranje zahtjeva procesorsko vrijeme zbog kojeg se može znatno usporiti genetski algoritam.

3.6 Genetski operatori

Osim selekcije, reprodukcija je druga važna karakteristika genetskog algoritma. U reprodukciji sudjeluju *dobre* jedinke koje su preživjele proces selekcije. Reprodukcija je razmnožavanje s pomoću genetskog operatora *križanja*. Tijekom procesa reprodukcije dolazi i do slučajnih promjena nekih gena ili *mutacije* te zamjene gena ili *inverzije.*

3.6.1 Križanje

U procesu križanja (*crossover*)sudjeluju dvije jedinke koje se nazivaju *roditelji*. Dakle, križanje je binarni operator. Križanjem nastaje jedna ili dvije nove jedinke koje se nazivaju *djeca*. Najvažnija karakteristika križanja jest da *djeca* nasljeđuju svojstva svojih roditelja. Ako su roditelji *dobri* (prošli su proces selekcije), tada će najvjerojatnije i *dijete* biti dobro, ako ne i bolje od svojih roditelja.



Slika 3.7: Križanje s jednom točkom prekida

Križanje može biti definirano s proizvoljnim brojem prekidnih točaka. Uniformno križanje je križanje s *b-1* prekidnih točaka (*b* je broj bitova). Vjerojatnost da dijete naslijedi svojstvo jednog roditelja je 0.5, odnosno jednaka je vjerojatnost nasljeđivanja svojstava za oba roditelja. Ako se te vjerojatnosti razlikuju za pojedine gene tada se takvo uniformno križanje naziva *p-uniformno* križanje. Na primjer: ako je *p=0.3* tada je vjerojatnost da će jedan bit biti naslijeđen od prvog roditelja 30%, a od drugog 70%. Ako je vjerojatnost nasljeđivanja različita za pojedine gene, tada se zadaje maska koja definira za svaki gen posebno koja je vjerojatnost nasljeđivanja.

gen	1	2	3	4	5	6	7	8	9	...	b-3	b-2	b-1	b
p	0.1	0.5	0.9	0.1	1	1	0.5	0.1	0	....	0.7	0.7	0.6	0.1

Tablica 3.3 : Primjer maske koja određuje vjerojatnost odabira gena

Maska prikazana u tablici 3.3 određuje da se geni 5 i 6 uzimaju od prvog roditelja. Gen s rednim brojem 9 se kopira od drugog roditelja, a gen 1 će najvjerojatnije (s 90% vjerojatnosti) biti od drugog roditelja.

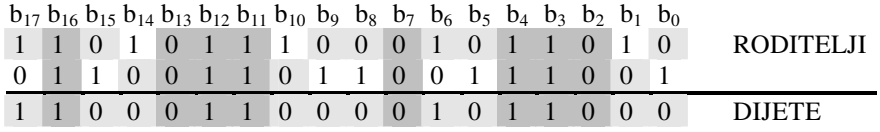
Uniformno križanje (p-uniformno križanje s p=0.5 za svaki gen) se može realizirati na više načina kako se to može vidjeti iz (3.21) i (3.22). Uniformno križanje se može izvesti i kao logička operacija nad bitovima. Potrebno je izdvojiti one bitove koji su jednaki i prepisati ih, a ostale postaviti na 0. To se postiže jednostavno logičkom funkcijom *I*: ***A I B***, gdje su ***A*** i ***B*** roditelji. Neka je ***R*** kromosom dobiven slučajnim procesom. Logičkom operacijom ***R I (A XILI B)*** dobiva se maska slučajnih bitova upravo na mjestima gdje se roditelji razlikuju. I konačno logičkom operacijom *ILI* se dobiva traženo rješenje:

$$\mathbf{DIJETE = AB + R(A \oplus B)}. \qquad (3.21)$$

odnosno:

$$\mathbf{DIJETE = AB + RA + RB}. \qquad (3.22)$$

Uniformno križanje je najlakše i najjednostavnije implementirati kao logičku operaciju, a i najbrže je, jer su logičke operacije sastavni dio strojnog jezika računala.



Slika 3.8: Primjer uniformnog križanja

U ovisnosti da li zamjena bitova kod operatora križanja ovisi o poziciji ili broju zamijenjenih bitova, definira se *pozicijska* i *distribucijska* sklonost operatora križanja. Križanje s jednom točkom prekida ima maksimalnu pozicijsku, a minimalnu distribucijsku sklonost, jer vjerojatnost zamjene nekog bita u kromosomu ovisi isključivo o poziciji tog bita. Druga krajnost je uniformno križanje koje ima minimalnu pozicijsku sklonost, a maksimalnu distribucijsku, jer svi bitovi bivaju zamijenjeni bez obzira na njihovu poziciju u kromosomu.

Križanje s jednom i dvije točke prekida čuva shemu (vidi poglavlje 4). No, kad populacija postane homogena, prostor koji pretražuje algoritam se smanjuje. Upravo je to razlog zašto se takvo križanje koristi pri većim populacijama, jer veća populacija ima i veću raznolikost shema. Druga krajnost je uniformno križanje koje s velikom vjerojatnošću lomi sheme, ali pretražuje veći prostor. Stoga se uniformno križanje koristi pri manjim populacijama.

Segmentno križanje je križanje s više točaka prekida, s tim da je broj točaka i pozicija prekida slučajna za svako pojedino križanje

Miješajuće križanje obavlja se u tri koraka. U prvom koraku izmiješaju se bitovi svakom roditelju. Drugi korak je klasično križanje s jednom ili više točaka prekida. I konačno u trećem koraku, izmiješani bitovi kod roditelja vrate se na stara mjesta (roditelji ostaju nepromijenjeni).

Pokazalo se korisnim provjeravati prije križanja da li su roditelji jednaki. S jedne strane tada nema smisla provoditi križanje - dovoljno je kopirati jedan od roditelja. S druge strane na taj način nastaje još jedan duplikat. Umjesto da se implementira posebni mehanizam otklanjanja duplikata, ovo je dobra prilika da se eliminacija duplikata izvede u sklopu operatora križanja. Dakle, ako provjera da li su roditelji jednaki daje pozitivan odgovor, mutira se jedan od roditelja, a dijete se dobiva slučajnim odabirom kromosoma (kao kod inicijalizacije).

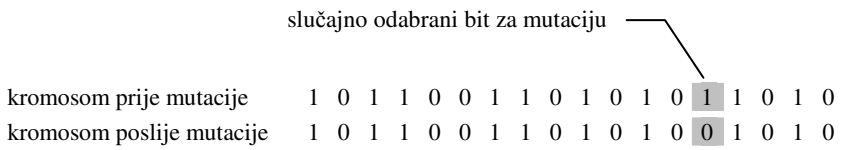
Pretpostavlja se da je upravo operator križanja to što razlikuje genetski algoritam od ostalih metoda optimiranja. To se ne može reći za operator mutacije kojega možemo sresti i kod simuliranog kaljenja i kod evolucijske strategije.

3.6.2 Mutacija

Drugi operator koji je karakterističan za genetski algoritam je mutacija ili slučajna promjena jednog ili više gena. Mutacija je unarni operator jer djeluje nad jednom jedinkom. Rezultat mutacije je izmijenjena jedinka.

Parametar koji određuje vjerojatnost mutacije p_m jednog bita je ujedno i parametar algoritma. Ako vjerojatnost mutacije teži k jedinici, tada se algoritam pretvara u algoritam slučajne pretrage prostora rješenja. S druge strane, ako vjerojatnost mutacije teži k nuli, postupak će najvjerojatnije već u početku procesa optimiranja stati u nekom lokalnom optimumu.

Jednostavna mutacija svaki bit kromosoma mijenja s jednakom vjerojatnošću p_m.



Slika 3.9: Jednostavna mutacija

Potpuna mutacija slučajnim postupkom odabire kromosom (a ne gen) za mutaciju i tada sve bitove ispremiješa. To je specijalan oblik miješajuće mutacije čija je prva granica početak kromosoma, a druga granica kraj kromosoma. Miješajuća mutacija je vrsta mutacije koja slučajnim postupkom odabire kromosom za mutaciju, prvu i drugu granicu (ili uzorak) i tada ili izmiješa gene ili ih slučajno generira (potpuna miješajuća mutacija) ili ih invertira (invertirajuća, miješajuća mutacija).

vrsta mutacije	slučajno odabran jedan gen npr. peti	slučajno generirana maska: 001001111000	svi geni
jednostavna mutacija	1101**0**0001011	11**1111110**011	**001001110100** (potpuna inverzija)
miješajuća mutacija	–	11**0**11**0100**011	**011110100101** (broj jedinica i nula ostaje isti)
potpuna miješajuća mutacija	–	11**1**11**0011**011	**011101011010** (potpuno slučajni kromosom)
invertirajuća miješajuća mutacija	–	11**1**11**1110**011	**001001110100** (potpuna inverzija)

Tablica 3.4 : Različite vrste mutacija nad kromosomom: **1 1 0 1 1 0 0 0 1 0 1 1**

Evolucijski programi najčešće ne koriste binarni prikaz. U tom slučaju umjesto parametra p_m (vjerojatnost mutacije jednog gena odnosno bita), definira se vjerojatnost mutacije kromosoma p_M. Odnos između ta dva parametra je sljedeći:

$$p_M = 1-(1-p_m)^n, \qquad (3.22)$$

gdje je n broj bitova koji čine kromosom. Primjerice, neka je vrijednost parametra p_m=0.01 (jedan od sto bitova će biti promijenjen) i neka je kromosom predstavljen nizom od n=32 bita. Tada je p_M =0.275 . Dakle, 27 kromosoma od 100 će biti izmijenjeni.

Mutacijom se pretražuje prostor rješenja i upravo je mutacija mehanizam za izbjegavanje lokalnih minimuma. Naime, ako cijela populacija završi u nekom od lokalnih minimuma, jedino slučajnim pretraživanjem prostora rješenja pronalazi se bolje rješenje. Dovoljno je da jedna jedinka (nastala mutacijom) bude bolja od ostalih, pa da se u nekoliko slijedećih generacija, sve jedinke *presele* u prostor gdje se nalazi bolje rješenje.

Uloga mutacije je i također i u obnavljanju *izgubljenog* genetskog materijala. Dogodi li se, npr. da sve jedinke populacije imaju isti gen na određenom mjestu u kromosomu, samo križanjem se taj gen nikad ne bi mogao promijeniti. Ako je riječ o binarnom prikazu kromosoma, time je izgubljeno čak pola prostora pretraživanja.

3.7 Parametri algoritma

Bez obzira o kakvom se genetskom algoritmu radi, algoritam ima slijedeće parametre: veličina populacije, broj generacija ili iteracija i vjerojatnost mutacije. Za generacijski genetski algoritam potrebno je još navesti i vjerojatnost križanja. Broj jedinki za eliminaciju zadaje se umjesto vjerojatnosti križanja kod eliminacijskog genetskog algoritma. Vjerojatnost mutacije ovisi o broju jedinki za eliminaciju, jer upravo je toliko jedinki potrebno generirati križanjem. Uobičajene vrijednosti parametara za algoritme s "malim" i "velikim" brojem jedinki u jednoj populaciji dane su u tablici:

Parametri	oznaka	*"mala"* populacija	*"velika"* populacija
Veličina populacije	VEL_POP	30	100
Vjerojatnost mutacije	p_m	0.01	0.001
Vjerojatnost križanja	p_c	0.9	0.6
Broj jedinki za eliminaciju	M	VEL_POP/2	VEL_POP/4

Tablica 3.5: Popis parametara genetskog algoritma s generacijskom i eliminacijskom reprodukcijom

Za različite vrijednosti parametara algoritma, algoritam daje različite rezultate: brže ili sporije dolazi do boljeg ili lošijeg rješenja. Problem postavljanja djelotvornih vrijednosti parametara GA javio se sa prvim primjenama algoritma u optimiranju. Optimiranje parametara genetskog algoritma je složen postupak i zahtjeva izvođenje velikog broja eksperimenata. Dovoljno je optimizirati slijedeće parametre: veličina populacije, vjerojatnost mutacije, vjerojatnost križanja i broj jedinki za eliminaciju. Kako bi se riješio ovaj složeni problem optimiranja parametara, može se realizirati tzv. genetski algoritam nad genetskim algoritmom. Grefenstette je predložio da se za izbor vjerojatnosti križanja i mutacije upotrebljava posebni genetski algoritam čije bi izvršavanje teklo paralelno sa glavnim GA koji radi na rješavanju. Jedinka predstavlja skup vrijednosti parametara algoritma. Funkcija dobrote može biti određena jednadžbom:

$$dobrota = \frac{a}{T_{uk}} + b \cdot S \,, \qquad (3.23)$$

gdje je T_{uk} ukupno vrijeme trajanja optimiranja uz pomoć genetskog algoritma, a S je srednje odstupanje od točnog rješenja. a i b su parametri koji se zadaju i koji će ovisiti o tome da li je potrebno da algoritam brže radi i/li daje točnije rješenje. Nedostatak tog pristupa je da se često pokazuje nezgrapnim za provedbu na računalu.

Parametri genetskog algoritma ne moraju biti konstantni za vrijeme evolucije. Dinamički parametri mogu biti funkcija vremena ili broja iteracije (što je najčešći slučaj) ili mogu biti funkcija veličine raspršenosti rješenja. Ako su rješenja raspršena, valja povećati učestalost križanja, a smanjiti mutaciju. S druge strane, ako su rješenja uniformna potrebno je učiniti suprotno: povećati učestalost mutacije, a smanjiti učestalost križanja, jer križanjem dvaju istih rješenja neće se dobiti ništa novo, već samo treći, isti, kromosom.

Za optimiranje višemodalnih funkcija, genetski algoritmi trebaju posjedovati dvije važne karakteristike. Prva je sposobnost konvergiranja k optimumu (lokalni ili globalni) nakon pronalaženja područja u kojemu se isti nalazi. Druga je karakteristika sposobnost da se konstantno istražuju nova područja problemskog prostora u potrazi za globalnim optimumom. Nije teško postići samo jednu karakteristiku u isto vrijeme. Ako GA upotrebljava samo križanje, on će moći konvergirati k ekstremu kod koga se trenutno zatekao (misli se na članove populacije). Ako pak imamo samo mutaciju kao jedini genetski operator, GA će nakon dovoljnog broja generacija vrlo vjerojatno naći područje globalnog optimuma. Strategije prilagodbe traže kompromis između ove dvije krajnosti mijenjajući vjerojatnost mutacije i križanja ovisno o trenutnom stanju genetskog algoritma; zaštititi program od prerane konvergencije ili ga usmjeriti ka dijelu područja pretraživanja koje obećava.

3.8 Prilagodljivi genetski algoritam (AGA)

Autor ovog potpoglavlja je Domagoj Jakobović.

Prilagodljivi genetski algoritam (*AGA - Adaptive Genetic Algorithm*) bio je predložen 1994 godine (Srinivas i Patnaik). AGA je namijenjen radu u okruženju generacijskog genetskog algoritma.

Da bi se vjerojatnosti križanja i mutacije pravovaljano mijenjali potrebno je na neki način detektirati trenutno stanje populacije GA, da li konvergira prema nekom optimumu. Način koji koristi AGA je promatranje odnosa prosječne dobrote $\overline{D}$ i dobrote najboljeg člana populacije $D_{\max}$. Izraz $D_{\max} - \overline{D}$ će vjerojatno biti manji za populaciju koja je konvergirala prema optimumu nego za populaciju rasutu u području rješenja. Autori su tu osobinu uočili u svim pokusima s GA. Vrijednost spomenutog izraza koristi se kao pokazatelj stupnja konvergencije genetskog algoritma, a p_m i p_c se mijenjaju ovisno o njoj. Cilj je da se p_m i p_c povećaju kada GA konvergira ka lokalnom optimumu i obratno.

To nije sve što AGA postiže. Kada, recimo, populacija konvergira ka lokalnom optimumu, povećane vjerojatnosti križanja i mutacije mogu pokvariti rješenja bliska optimumu, pa populacija ne bi morala nikada niti doseći optimum.

Zato se zaštićuju *dobra* rješenja u populaciji tako da su p_m i p_c niži za njih a viši za lošija rješenja. Dok su dobri kromosomi zaslužni za točniju konvergenciju, lošiji su zaduženi za zaštitu od zaglavljivanja na lokalnom optimumu. Matematički formulirano, vrijednosti p_m i p_c se za svakog člana populacije izračunavaju pomoću slijedećih formula:

$$p_c = k_1(D_{\max} - D')/(D_{\max} - \overline{D}), \qquad k_1 \le 1.0\,, \tag{3.24}$$

$$p_m = k_2(D_{\max} - D)/(D_{\max} - \overline{D}), \qquad k_2 \le 1.0 \tag{3.25}$$

gdje je D dobrota kromosoma koji se izabire za mutaciju, a D' dobrota boljega od dva rješenja odabrana za križanje. Primijetimo da su p_m i p_c jednaki nuli za najbolje rješenje (dobili smo elitizam, što i ovdje pokazuje da je to vrijedna ideja), kao i to da je p_c=k_1 za član sa $D' = \overline{D}$, a p_m=k_2 za član sa $D = \overline{D}$. Za kromosome sa dobrotom manjom od prosječne, vrijednosti p_m i p_c bi mogle postati veće od 1.0 . Zato se uvode sljedeća ograničenja:

$$p_c = k_3, \quad D' \le \overline{D}$$

$$p_m = k_4, \quad D \le \overline{D}$$

gdje su k_3, k_4 manji ili jednaki 1.0 .

Kako su izrazi sada postavljeni, najbolje rješenje se prenosi iz generacije u generaciju nepromijenjeno. Uz mehanizam odabira to može uzrokovati prekomjernu brojnost 'najboljih' rješenja koja se ne bi uopće bila obrađivana. Zato se uvodi 'obavezna' vjerojatnost mutacije od 0.005 za svako rješenje u AGA.

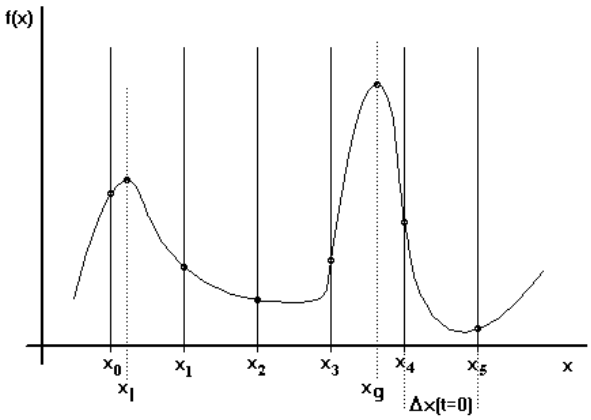
Ostaje nam izbor vrijednosti za konstante k_1, k_2 k_3 i k_4. Cilj je djelovanja AGA da spriječi populaciju da zaglavi na lokalnom optimumu, stoga se rješenja sa dobrotom manjom od $\overline{f}$ namjenjuju pretraživanju prostora rješenja u potrazi za globalnim optimumom. Takva rješenja trebaju biti sasvim 'potrgana' te se za njih postavlja vrijednost k_4=0.5. Isto možemo reći i za kromosome sa prosječnom dobrotom, pa se i konstanti k_2 pridjeljuje vrijednost 0.5; time su definirane konstante vezane uz mutaciju. Iz sličnog zaključivanja postavljaju se konstante k_1 i k_3 na vrijednost 1.0. Ovo osigurava da sva rješenja sa dobrotom jednakom ili manjom od prosječne sudjeluju u križanju.

3.9 Grubo i fino podešavanje rješenja

Pokazalo se da genetski algoritam daje slabe rezultate pri finom podešavanju rješenja. Holland predlaže da se genetski algoritam koristi kao pretprocesor u procesu optimiranja funkcije cilja prije nego što se pokrene glavni mehanizam pretrage. Za lokalno pretraživanje, odnosno za pronalaženje rješenja s velikom preciznošću, genetski algoritam zahtjeva odgovarajuću (veliku) dužinu kromosoma. Na primjer, za jednodimenzijski problem nad jediničnim intervalom i preciznošću 10^{-3} potrebno je 10 bitova, a za preciznost 10^{-6} potreban je barem dvostruki broj bitova: 20 (Tablica 3.1). Takvim povećanjem preciznosti prostor pretrage se povećao za 2^{10} točaka, što je ekvivalentno dvodimenzijskom problemu s binarnim prikazom rješenja i dužine kromosoma od 10 bitova. Dakle, jednodimenzijski problem koji se rješava uz pomoć genetskog algoritma gdje je kromosom dužine $N*b$ bitova, ekvivalentan je N-dimenzijskom problemu gdje je kromosom dužine b bitova.

S jedne strane može se odustati od finog podešavanja rješenja i primijeniti genetski algoritam, kao pretprocesor ili s druge strane, mogu se na razne načine poboljšati svojstva finog podešavanja genetskog algoritma. Tako je, npr., vrlo izraženi problem finog podešavanja rješenja kod genetskog algoritma s brojem s pomičnim zarezom, riješen dinamičkim operatorom mutacije. Ideja je posuđena od simuliranog kaljenja, gdje s vremenom (s brojem iteracija) temperatura, kao parametar algoritma, pada. Na taj se način prostor pretrage smanjuje. Smanjenjem prostora pretrage algoritam bolje podešava rješenje uz pretpostavku da je identificirano područje globalnog optimuma, odnosno da se globalni optimum nalazi u prostoru pretrage.

Problem finog podešavanja rješenja kod genetskog algoritma s binarnim prikazom rješenja može se riješiti dinamičkom dužinom kromosoma. Na taj način osigurana je i dinamička preciznost rješenja. Na početku procesa optimiranja dužina kromosoma, a time i preciznost, je mala. U tom prvom dijelu optimiranja nastoji se samo identificirati manje područje (manji interval) unutar kojeg se nalazi rješenje. U drugom, završnom dijelu optimiranja provodi se fino podešavanje rješenja povećavanjem preciznosti, tj. produžavanjem kromosoma. Tijekom vremena postupak produžavanja kromosoma je obično linearan. Pred genetski algoritam se postavlja novi problem: kako odrediti početnu preciznost, a da se zbog premale preciznosti ne identificira umjesto globalnog, lokalni optimum. Navedeni problem može se ilustrirati na funkciji cilja koja je prikazana na slici 3.12, a preciznost je određena odsječcima na osi x.



Slika 3.10: Primjer funkcije cilja i suviše male početne preciznosti

Globalni optimum je u intervalu (x_3, x_4), a rješenje koje pronalazi genetski algoritam s prikazanom početnom preciznošću Δx_0 je x_l - lokalni optimum u blizini x_1, jer je $\max_{i=0,1,2,3,4,5} \{f(x_i)\} = f(x_0)$.

4. TEOREM SHEME I HIPOTEZA BLOKOVA

Teorijske osnove genetskog algoritma (teorem sheme i hipoteza blokova) odnose se na genetski algoritam s binarnim prikazom.

Da bi vrijedili teorem sheme i hipoteza blokova, potrebno je zadovoljiti slijedeće pretpostavke:

- populacija je neograničena,
- funkcija dobrote vjerno odražava problem i
- geni u kromosomu su međusobno nezavisni.

4.1 Teorem sheme

Shema je uzorak ili skup rješenja koja su međusobno slična. Sličnost se odnosi na jednakost gena na pojedinim mjestima kromosoma. Ostali geni mogu biti ili isti ili različiti i označavaju se zvjezdicom: "*". Primjerice, shema `*101*1` predstavlja skup od četiri rješenja:

0101**0**1,
0101**1**1,
1101**0**1,
1101**1**1.

Zvijezdica "*" označava "*bilo što*" ili "*bilo koji znak*", a u ovom slučaju to je 0 ili 1.

Neka je kromosom duljine n, a shema neka sadrži r znakova "*". Takva shema predstavlja 2^r rješenja. S druge strane, jedno rješenje može biti predstavljeno s 2^n shema. Ukupan broj shema kod binarnog prikaza rješenja iznosi 3^n, jer jedan znak sheme može biti "*", "0" ili "1".

Red sheme S (oznaka $o(S)$) je broj nepromjenjivih (fiksnih) gena, odnosno broj svih onih znakova sheme S koji nisu "*":

$$o(S)=n-r. \qquad (4.1)$$

Na primjer, o(*1*001)=4 i o(****11******)=2.

Definirana dužina sheme ili definirajući razmak (oznaka $\delta(S)$) je udaljenost između prve i zadnje nepromjenjive znamenke. Na primjer, δ(*01110011*)=9-2=7, δ(10**0)=5-1=4 i δ(**1**)=3-3=0.

Teorem sheme. Broj jedinki koje sadrže shemu niskog reda, kratke definirane dužine i iznadprosječne dobrote raste eksponencijalno.

Formalni opis teorema sheme prikazan je slijedećom nejednakošću:

$$N(S,t+1) \geq N(S,t)\frac{\overline{D_S}}{\overline{D}}\left[1-\frac{\delta(S)}{n-1}p_c - o(S)p_m\right], \qquad (4.2)$$

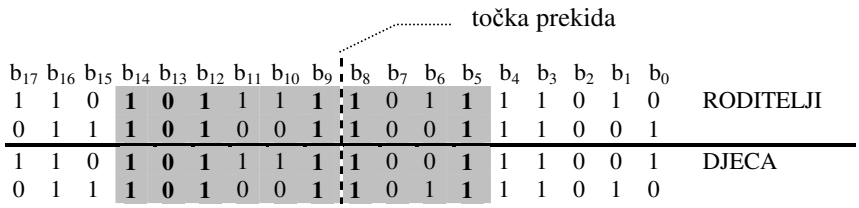
gdje su:

$N(S,t)$	- očekivani broj jedinki koje su podskup sheme S u generaciji t
$\overline{D_S}$	- prosječna dobrota sheme
$\overline{D}$	- prosječna dobrota populacije (3.12)
$\delta(S)$	- definirana dužina sheme S
$o(S)$	- red sheme S
p_c	- vjerojatnost križanja
p_m	- vjerojatnost mutacije
n	- dužina kromosoma

Vjerojatnost gubitka sheme zbog operatora križanja je proporcionalna vjerojatnosti križanja i definiranoj dužini sheme, a obrnuto je proporcionalna broju mogućih prekidnih točaka (n-1) i iznosi: $p_{gc}=\frac{\delta(S)}{n-1}p_c$.

Točnije, vjerojatnost gubitka sheme zbog operatora križanja je manja ili jednaka p_{gc}, jer postoji mogućnost da oba roditelja sadrže shemu S. Slučajnim odabirom točke prekida shema S ostaje sačuvana, jer križanjem nastaje ista shema iz istih dijelova koje sadrže roditelji. Na primjer, promotrimo shemu 101**11**1 koju sadrže oba roditelja. Neka je točka prekida između dvije jedinice unutar sheme. Nakon križanja shema je ostala sačuvana (Slika 4.1).

Svako križanje, osim miješajućeg križanja, (npr. uniformno križanje ili križanje s više točaka prekida) čuva shemu ako je sadrže oba roditelja.



Slika 4.1:Shema je sačuvana nakon križanja s jednom točkom prekida ako oba roditelja sadrže istu shemu

Vjerojatnost gubitka sheme zbog mutacije je proporcionalna vjerojatnosti mutacije i redu sheme: $p_{gm} = o(S)p_m$.

Treba spomenuti da u izrazu (4.2) u uglatoj zagradi nedostaje četvrti član koji se zanemaruje. Ispravno, spomenuti izraz glasi:

$$N(S,t+1) \geq N(S,t)\frac{\overline{D_S}}{\overline{D}}\left[1-\frac{\delta(S)}{n-1}p_c - o(S)p_m + \frac{\delta(S)}{n-1}o(S)p_m p_c\right], \qquad (4.3)$$

4.2 Hipoteza građevnih blokova

Rad genetskog algoritma može se predstaviti kao natjecanje između shema gdje kratke sheme niskog reda s nadprosječnom dobrotom pobjeđuju. Sastavljajući i nizajući takve sheme algoritam dolazi do rješenja. Kratke sheme niskog reda s iznadprosječnom dobrotom nazivaju se građevni blokovi.

Hipoteza građevnih blokova. Genetski algoritam pretražuje prostor rješenja nizajući sheme niskog reda, kratke definirane dužine i iznadprosječne dobrote, zvane građevni blokovi.

U slučaju da su geni u kromosomu međusobno zavisni, hipoteza građevnih blokova će i tada vrijediti ako su ti zavisni geni *blizu*.

S druge strane, građevni blokovi mogu navesti genetski algoritam na krivo rješenje, tj. da konvergira lokalnom optimumu. Ta se pojava naziva *decepcija*[3], a funkcija koja izaziva tu pojavu *decepcijska* ili *varajuća* funkcija.

Najjednostavniji primjer decepcijske funkcije je tzv. minimalni decepcijski problem. Dvobitna funkcija koja izaziva minimalni decepcijski problem je određena nejednakostima:

```
f(11)>f(00),      f(*0)>f(*1),
f(11)>f(01),      f(0*)>f(1*).
f(11)>f(10),
```

0* i ***0** su kratke sheme niskog reda i iznadprosječne dobrote: f(*0)>f(*1) i f(0*)>f(1*), ali ne sadrže niz **11** za koji se postiže optimum.

Potpun decepcijski problem je takav problem gdje sve kratke sheme niskog reda koje sadrži optimalno rješenje imaju ispodprosječnu dobrotu u odnosu na ostale sheme s istim nepromjenjivim genima.

Primjer potpunog decepcijskog problema. Neka se kromosom sastoji od niza jedinica i nula duljine *b*. Funkcija *f* definirana nad kromosomom vraća broj jedinica, a ako su sve nule vraća vrijednost *2b*. Globalni optimum se postiže upravo s kromosomom koji sadrži sve nule i niti jednu jedinicu: *f(0000..0000)=2b*, a lokalni optimum je kromosom sa svim jedinicama: *f(1111....1111)=b.* Svaka kratka shema niskog reda koju sadrži optimalno rješenje (npr. *00*0* ili *0*0*) ima ispodprosječnu dobrotu u odnosu na ostale sheme s istim fiksiranim genima.

4.3 Broj shema

Broj shema *#S* nekog prikaza određen je izrazom:

$$\#S = (c+1)^b \qquad (4.4)$$

gdje je *c* kardinalni broj gena (kod binarnog prikaza *c=2*, a kod trinarne *c=3*), a *b* je broj bitova.

Promatraju li se druge vrste prikaza osim binarnog {0,1}, tzv. ne-binarne vrste prikaza, Primjerice trinarni prikaz {0,1,2} valja preciznije definirati znak "*". Treba uvesti oznaku za, npr., bilo koji znak 0 ili 1, ali ne i 2: "$*_{01}$". Po takvoj konvenciji znak "*" je ekvivalentan oznaci "$*_{012}$". U tom slučaju broj shema *#S* iznosi:

$$\#S = (2^c - 1)^b \qquad (4.5)$$

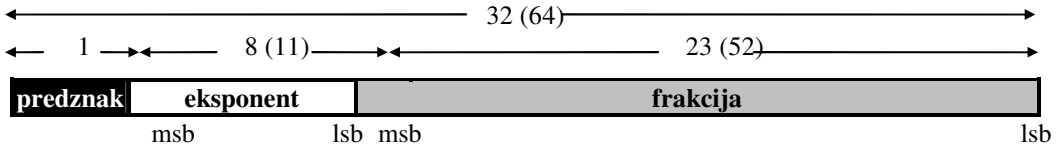
Izraz (4.4) vrijedi samo za binarni prikaz, a ne i za ostale vrste prikaza, dok izraz (4.5)vrijedi općenito.

[3] *deceive,* eng. - varati, zavesti, *deception*, eng. - varka, prijevara, *decipio,* lat. - prevariti

5. KROMOSOM KAO BROJ S POMIČNOM TOČKOM

5.1 Prikaz broja s pomičnom točkom

Neka je zadatak optimizirati funkciju cilja realne varijable. U tom slučaju kromosom može biti uobičajno niz jedinica i nula, tj. cijeli broj, koji predstavlja realan broj u zadanom intervalu, zapisan binarno kao što je već opisano. Alternativno rješenje je broj s pomičnom točkom. Za takav prikaz u genetski algoritam nije potrebno ugrađivati poseban mehanizam dekodiranja kakav mora biti prisutan pri binarnom prikazu, jer kromosom i jest realan broj (samo je zapisan kao broj s pomičnom točkom prema IEEE 754-1985 standardu) i može se direktno uporabiti za izračunavanje vrijednosti funkcije cilja.



Slika 5.1: Broj s pomičnom točkom jednostruke (dvostruke) preciznosti prema IEEE 754-1985 standardu

Broj s pomičnom točkom zapisan je u slijedećem formatu: prvi bit je predznak broja. Slijedi eksponent od 8, odnosno kod broja dvostruke preciznosti, 12 bitova. Ostali bitovi su frakcija koja čini signifikant. Signifikant je jedna jedinica (skriveni bit) iza koje slijedi frakcija.

5.2 Posebno definirani genetski operatori

Za genetski algoritam s kromosomom kao brojem s pomičnom točkom potrebno je posebno definirati genetske operatore.

Križanje. Križanje treba omogućiti prenošenje svojstava iz generacije u generaciju. Operator križanja daje kao rezultat slučajan broj u intervalu čije granice određuju roditelji:

$$x_{dijete} = \text{slučajan\_broj\_u\_intervalu}(x_{roditelj1}, x_{roditelj2}). \quad (5.1)$$

Mutacija. Mutacija je slučajna promjena kromosoma. Kod binarnog prikaza, mutacija je slučajna promjena jednog gena, odnosno jednog bita: ako je 0 postaje 1 i obrnuto. U ovom slučaju operator mutacije može biti definiran kao slučajan broj unutar cijelog zadanog intervala, odnosno cijele domene:

$$x_k' = \text{slučajan\_broj\_u\_intervalu}(DG, GG). \quad (5.2)$$

Takvim operatorom mutacije pretražuje se cijeli prostor rješenja tijekom cijelog evolucijskog procesa. Na kraju procesa pretrage, kad je identificirano područje globalnog optimuma, potrebno je izvršiti fino podešavanje rješenja. Razlog lošem ponašanju algoritma s ovakvom jednostavnom mutacijom leži u maloj vjerojatnosti p_δ da poslije mutacije novo rješenje postane element malog intervala δ unutar domene ograničene donjom i gornjom granicom:

$$p_\delta = \frac{\delta}{GG - DG}, \quad (5.3)$$

gdje je $\delta << (GG-DG) \Rightarrow p_\delta \cong 0$.

U svrhu poboljšanja učinkovitosti mutacije u području finog podešavanja rješenja definira se dinamički operator mutacije. Vjerojatnost mutacije je konstantna, ali se prostor djelovanja mutacije s vremenom (s brojem generacija ili iteracija) smanjuje. Ako je odabran kromosom x_k za mutaciju, tada se novi kromosom izračunava s pomoću slijedeće formule:

$$x_k' = \text{slučajni\_broj\_u\_intervalu}(DGM, GGM), \quad (5.4)$$

gdje su dinamičke granice *DGM* i *GGM* određene slijedećim jednadžbama:

$$DGM = \max\{DG, x_k - (GG - DG)r(t)\}, \qquad GGM = \min\{GG, x_k + (GG - DG)r(t)\}, \quad (5.5), (5.6)$$

$$r(t) = 1 - s^{\left(1 - \frac{t}{T}\right)^b}, \quad (5.7)$$

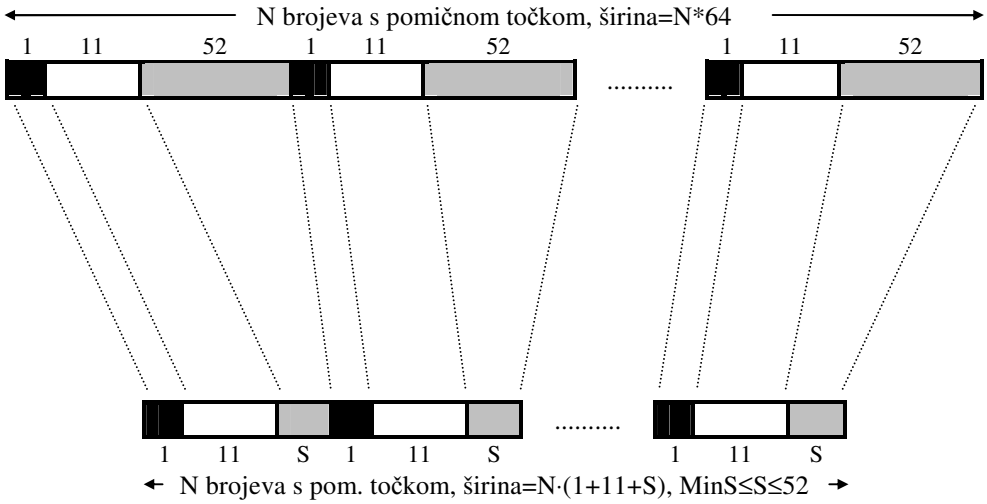
s je slučajni broj iz intervala *[0,1]*, b je sistemski parametar koji određuje stupanj ovisnosti o broju iteracije (u ovom slučaju $b=5$) i T je ukupan broj iteracija.

Funkcija *r(t)* omogućava dinamički prostor djelovanja operatora mutacije. Na početku procesa evolucije kad *t* ima malu vrijednost, prostor djelovanja mutacije je unutar cijelog intervala *[DG,GG]*. Dok pri kraju evolucije (*t→T)* prostor djelovanja mutacije teži k nuli. Opisanim postupkom se postiže djelotvornije fino podešavanje rješenja.

5.3 Uobičajeni genetski operatori nad brojem s pomičnom točkom

Dosad su bile prikazane dvije vrste prikaza: binarni i broj s pomičnom točkom. Za svaku vrstu prikaza posebno su definirani genetski operatori. Genetski algoritam s binarnim prikazom ima implementirane klasične genetske operatore nad nizom bitova. S druge strane, genetski algoritam s brojem s pomičnom točkom ima posebne operatore križanja i mutacije koji djeluju nad realnim brojevima. Eksperimentalnom promjenom prikaza (umjesto binarnog uporabljen je prikaz brojem s pomičnom točkom) dobili su se slični rezultati pri optimiranju funkcija cilja više varijabli, ali za kraće vrijeme optimiranja. U ovom poglavlju razmatra se ideja da se broj s pomičnom točkom tretira kao niz bitova. Nad nizom bitova je moguće primijeniti klasične genetske operatore: križanje s jednom ili više točaka prekida te mutacija kao slučajna promjena jednog bita.

Problem finog podešavanja rješenja rješava se preslikavanjem. Naime, tijekom cijelog procesa optimiranja nije potrebno djelovati genetskim operatorima na sve bitove broja s pomičnom točkom: u početku procesa dovoljno je promatrati samo eksponent i dio frakcije (i to onaj dio frakcije koji ima veću težinu) kako je prikazano na slici:



Slika 5.2: Postupak preslikavanja vektora brojeva s pomičnom točkom u kraći kromosom

Nadalje, valja primijetiti da je takvim prikazom realnog broja gustoća realnih brojeva, koji se daju prikazati, najveća oko nule. Zamislimo realni pravac na kojem su označeni svi realni brojevi koji se daju prikazati opisanim standardom (s pomoću broja s pomičnom točkom). Kako se udaljavamo od nule na spomenutom pravcu, gustoća brojeva pada s potencijom broja 2. Na primjer, u intervalu [0,2) ima isto toliko brojeva koliko i u intervalu [2,*MAX_FP*], gdje je *MAX_FP* najveći realan broj koji se može zapisati kao broj s pomičnom točkom. Ako generiramo početnu populaciju takvim slučajnim procesom da svi bitovi imaju jednaku vjerojatnost da postanu nula ili jedinica (kao što je to slučaj s binarnim prikazom) tada taj generator slučajnih brojeva nema uniformnu razdiobu. Na primjer veća je vjerojatnost da slučajan broj bude u intervalu [0,1], nego u intervalu [100,101]. Dakle, treba drugačije definirati generator slučajnih brojeva - tako da ima uniformnu razdiobu u intervalu [dg,gg]. Ugrađeni generator slučajnih brojeva glasi:

$$x_{sluc} = dg + \frac{rand()}{MAX\_SLUC}(gg - dg), \qquad (5.8)$$

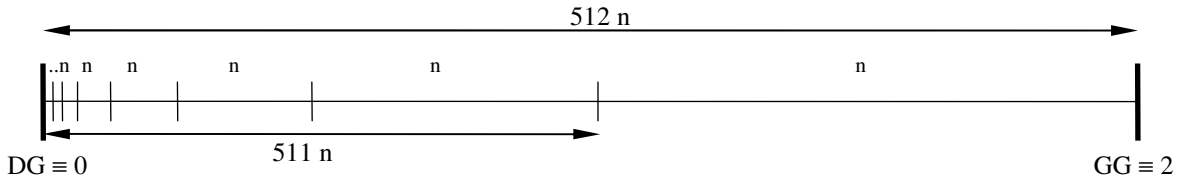
gdje funkcija *rand()* vraća slučajan cijeli broj u intervalu [0,MAX_SLUC] i to po uniformnoj.

U početku procesa optimiranja nisu važni posljednji bitovi frakcije. Stoga nije ni potrebno na njih djelovati genetskim operatorima. U svrhu poboljšanja performansi algoritma realiziran je genetski algoritam s promjenjivom dužinom kromosoma. Na početku procesa kromosom je najmanji i sastoji se od *N* skraćenih brojeva s pomičnom točkom. Skraćeni broj s pomičnom točkom se sastoje od predznaka, eksponenta i dijela frakcije sa značajnijim bitovima. Pri kraju procesa optimiranja kromosom je pune duljine i sastoji se od *N* brojeva s pomičnom točkom. Na taj način je proces pretrage podijeljen na dva dijela. U prvom dijelu se vrši detekcija područja u kojem se nalazi globalni optimum, a u drugom dijelu se vrši fino podešavanje rješenja. Dakle, na početku procesa optimiranja *S* je najmanji i iznosi *MinS*, a pri kraju procesa *S* je jednak cijeloj dužini frakcije.

Eksperimentiranje je pokazalo da opisani genetski algoritam s uobičajenim genetskim operatorima nad brojem s pomičnom točkom ne daje očekivano bolje rezultate. Kao što je prikazano na slici 5.4, problem je u nejednakoj gustoći brojeva na brojevnom pravcu. Ako donju granicu predstavlja broj 0, a gornju granicu broj 2, tada gustoća

brojeva raste s potencijom broja 2 kako se približavamo donjoj granici. Budući da je mutacija slučajna promjena jednog ili više bitova, u ovom slučaju bitova eksponenta, mutacija je generator slučajnih brojeva. Samo što u ovom slučaju to nije uniformna razdioba. Na primjer, ako se promatra devet bitova eksponenta, vjerojatnost da slučajno generirani broj bude u intervalu [(GG-DG)/2,GG] je 510 puta manja od vjerojatnosti da slučajni broj bude u duplo manjem intervalu [DG,(GG-DG)/4]:

$$p_{\left[\frac{GG-DG}{2},GG\right]} = \frac{1}{512} \text{ i } p_{\left[\frac{gg-dg}{2},gg\right]} = \frac{1}{512}.$$



Slika 5.3: Gustoća brojeva raste s potencijom broja 2 kako se približavamo 0, odnosno donjoj granici. (Promatra se 9 bitova eksponenta (2^9=512)).

Stoga je potrebno definirati takav operator mutacije da bude jednaka vjerojatnost generiranja slučajnog broja duž cijelog intervala [DGM,GGM] prema formuli (5.4). Genetski algoritam s takvim operatorom mutacije je malo sporiji, ali su zato rezultati znatno bolji - sumjerljivi s rezultatima dobivenih s pomoću genetskog algoritma s binarnim prikazom.

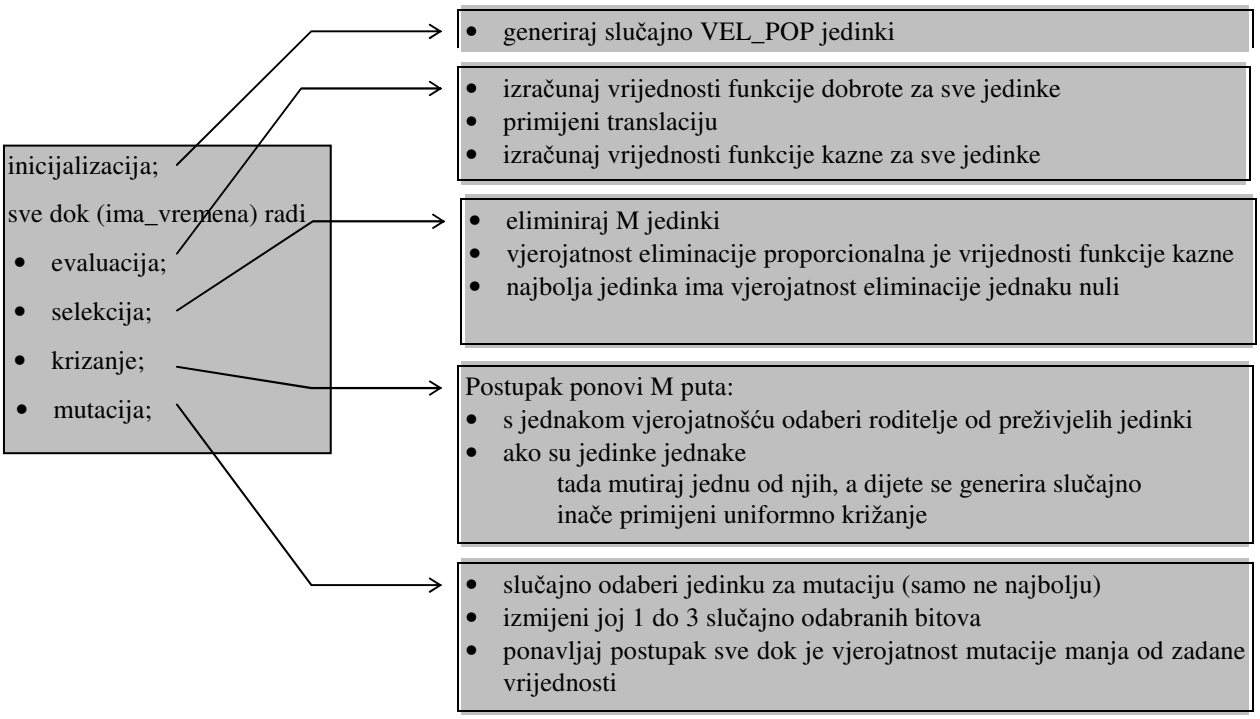
Prednost ovakvog pristupa problemu leži u mogućnosti da se postignu bolji rezultati u finom podešavanju rješenja dinamičkom dužinom kromosoma, a da se pri tom ne trebaju obavljati nikakve računske operacije. Samo je potrebno mijenjati prostor djelovanja genetskih operatora. U početku procesa optimiranja genetski operatori djeluju nad manjim brojem bitova (manja preciznost). S vremenom ili s brojem iteracija, genetski operatori djeluju nad većim brojem bitova (povećava se preciznost). I konačno, pri kraju procesa optimiranja, genetski operatori djeluju nad svim bitovima kromosoma (najveća preciznost). S tim da su svi genetski operatori (osim operatora mutacije nad eksponentom) uobičajeni, kao što su uniformno križanje i jednostavna mutacija.

6. PRIMJERI

6.1 Jednostavni eliminacijski genetski algoritam

vrsta GA	eliminacijski, M=50%
veličina populacije	100
prikaz	binarni, broj bitova=32
ispravak funkcije dobrote	translacija
eliminacija duplikata	da, prilikom križanja
križanje	uniformno, p_c=0.5
mutacija	jednostavna, p_m=0.01

Tablica 6.6: Vrsta i parametri ugrađenog genetskog algoritma

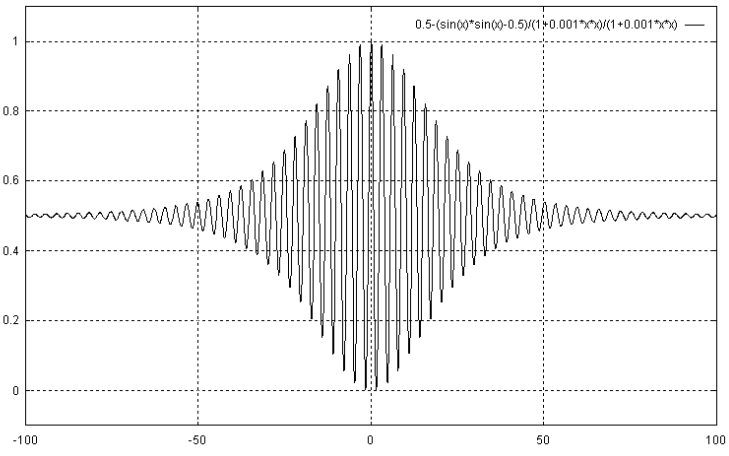


Slika 6.1: Ugrađeni genetski algoritam

6.2 Primjer rada GA

Rad genetskog algoritma (korak po korak) prikazan je na jednostavnom primjeru optimiranja funkcije cilja jedne realne varijable. Zadatak je pronaći globalni maksimum funkcije prikazane na slici 6.1.

Globalni optimum postiže se u točki (0,1). Koristi se binarni prikaz duljine kromosoma od 32 bita čime je postignuta preciznost na sedam decimala. Broj jedinki[4] koje čine populaciju je 16. Broj jedinki za eliminaciju je postavljen na 8, a broj selektiranih kromosoma za mutaciju je 5. Kad se već odabere kromosom za mutaciju, mutira se slučajnim odabirom



Slika 6.2: Ispitna funkcija $f(x) = 0.5 - \frac{\sin^2(x) - 0.5}{\left(1 + 0.001 \cdot x^2\right)^2}$

[4] Odabrana veličina populacije je suviše mala, ali je pogodna za demostraciju rada genetskog algoritma.

jedan do tri gena.

Prilikom inicijalizacije generira se početna populacija s pomoću generatora slučajnih brojeva:

rbr	kromosom	x	f(x)
0	10001010011100000111111010011100	8.1558	0.1381
1	11000001001000011101000011011110	50.8844	0.5127
2	01110010101111011111001011111000	-10.3578	0.3813
3	10011110101110000110010000001010	24.0002	0.3712
4	10101100101110100000010010110100	34.9427	0.5727
5	10001111110110110011100100010110	12.3878	0.8521
6	11001000110000000110001011010000	56.8371	0.5234
7	01010100111001011101001100000010	-33.6736	0.4785
8	11100001101100011101010001001100	76.3239	0.4970
9	11000110001011101110110011001110	54.8307	0.4702
10	00100011011101011111100000101000	-72.2962	0.5129
11	00100001110001110011100101111010	-73.6108	0.4890
12	10000011101100110000010101100100	2.8901	0.9308
13	00101110001101100010010000000110	-63.8973	0.4897
14	10001110101010111000101100000000	11.4610	0.2667
15	10000101011001001000111101110010	4.2131	0.2386

Tablica 6.1: Slučajno generirana populacija od 16 jedinki s evaluacijskim vrijednostima

Prva iteracija.

1. korak: **evaluacija.** Izračunavaju se vrijednosti kromosoma *x* prema formuli (3.2). Dobivena vrijednost se uvrštava u funkciju cilja i dobiva se *f(x)*.

2. korak: **procjena.** Identificira se najbolja i najlošija jedinka i izvrši se translacija. U ovom slučaju najbolja jedinka je kromosom s rednim brojem 12, a najlošija je kromosom s rednim brojem 0. Translacija se obavlja prema formuli (3.18) i rezultat su dobrote *d*. Po toj formuli najlošija jedinka ima dobrotu 0. Budući da se radi o eliminacijskoj selekciji, izračunaju se *kazne*.

3. korak: **selekcija.** Generira se *M* slučajnih brojeva u intervalu *[0,max{kp}],* gdje je *max{kp}* najveća vrijednost kumulativne dobrote. U prvom koraku, prije prve eliminacije, najveća vrijednost kumulativne dobrote iznosi 7.16 (Tablica 6.2). Ako je slučajno generirani broj *r* u intervalu $[kp_{i-1},kp_i]$ tada se *i*-ti kromosom briše. S pomoću generatora slučajnih brojeva generiran je broj 2.343. Za broj r=2.343 briše se jedinka s rednim brojem 4. Nakon svake eliminacije potrebno je ponovno izračunati kumulativnu dobrotu za sve jedinke koje slijede. U ovom slučaju eliminirana je jedinka s rednim brojem 4 i treba izračunati kumulativne dobrote jedinkama koje slijede: s rednim brojevima 5 do 15. Tek kada se izračuna novi (manji) *max{kp}*, generira se novi slučajni broj u novom intervalu. Vrijednost *max{kp}* je manja od one u prethodnom koraku za iznos *kazne* eliminirane jedinke. Generirani su još slijedeći slučajni brojevi: 0.17, 3.059, 4.261, 0.178, 3.002, 1.692 i 1.462. Postupak se ponavlja za svaki *r*. Na taj način u navedenom primjeru eliminirane su slijedeće jedinke 4, 0, 9, 14, 1, 11, 7 i 6.

Rb	d	kazna	kp	kromosom	x	f(x)	križanje
0	**0.00**	**0.79**	**0.79**	**10001010011100000111111010011100**	**8.1558**	**0.1381**	**10+2**
1	**0.37**	**0.42**	**1.21**	**11000001001000011101000011011110**	**50.8844**	**0.5127**	**8+2**
2	0.24	0.55	1.76	01110010101111011111001011111000	-10.3578	0.3813	
3	0.23	0.56	2.32	10011110101110000110010000001010	24.0002	0.3712	
4	**0.43**	**0.36**	**2.67**	**10101100101110100000010010110100**	**34.9427**	**0.5727**	**13+12**
5	0.71	0.08	2.75	10001111110110110011100100010110	12.3878	0.8521	
6	**0.39**	**0.41**	**3.16**	**11001000110000000110001011010000**	**56.8371**	**0.5234**	**12+2**
7	**0.34**	**0.45**	**3.61**	**01010100111001011101001100000010**	**-33.6736**	**0.4785**	**3+8**
8	0.36	0.43	4.05	11100001101100011101010001001100	76.3239	0.4970	
9	**0.33**	**0.46**	**4.51**	**11000110001011101110110011001110**	**54.8307**	**0.4702**	**13+10**
10	0.37	0.42	4.92	00100011011101011111100000101000	-72.2962	0.5129	
11	**0.35**	**0.44**	**5.37**	**00100001110001110011100101111010**	**-73.6108**	**0.4890**	**3+12**
12	0.79	0.00	5.37	10000011101100110000010101100100	2.8901	0.9308	
13	0.35	0.44	5.81	00101110001101100010010000000110	-63.8973	0.4897	
14	**0.13**	**0.66**	**6.47**	**10001110101010111000101100000000**	**11.4610**	**0.2667**	**10+13**
15	0.10	0.69	7.16	10000101011001001000111101110010	4.2131	0.2386	

Tablica 6.2: Stanje populacije nakon selekcije, ali prije križanja (crno su označene jedinke za eliminaciju).

4. korak: **križanje**. Križanjem preostalih jedinki popunjavaju se eliminirane jedinke. Veličina populacije je tijekom cijelog procesa optimiranja jednaka. Jedinke koje su preostale imaju međusobno jednaku vjerojatnost križanja. U prvom od *M* koraka se nadomješta jedinka s rednim brojem 4 i to križanjem slučajno odabranih jedinki 13 i 12. Jedinka pod rednim brojem 0 dobiva se križanjem slučajno odabranih jedinki 10 i 2, itd.

5. korak: **mutacija.** Selektira se *BR_M=5* puta jedna jedinka za mutaciju. Jedna jedinka može više puta biti odabrana za mutaciju. Sve jedinke imaju istu vjerojatnost mutacije osim najbolje čija je vjerojatnost mutacije 0 (elitizam). Slučajnim odabirom selektirane su slijedeće jedinke za mutaciju: 2, 11, 2, 5 i 7 (jedinka s rednim brojem 2 je dva puta selektirana). Mutirani bitovi su u tablicama napisani masno.

rbr	kromosom	križanje
0	01100010011111101111110001010100**0**	10+2
1	01100011101110011101011011000**0**	8+2
2	01110**0**10101111011111001**1**11**1**10**0**0	
3	10011110101110000110010000001010	
4	10101111001100100010010100100100	13+12
5	10001111110110**1**10011100100010110	
6	10100011101111110010010101110000	12+2
7	11010100101110010111010**1**00001110	3+8
8	11100001101100011101010001001100	
9	00101111001101000011110000101010	13+10
10	00100011011101011111100000101000	
11	100**1**10101011001000000101010010**1**0	3+12
12	10000011101100110000010101100100	
13	00101110001101100010010000000110	
14	00100110001101100011010000100100	10+13
15	10000101011001001000111101110010	

Tablica 6.3: Populacija nakon prve iteracije.

Druga iteracija

Nakon prve iteracije nije postignut bolji rezultat. Najbolja jedinka je ostala ista, ali je prosječna vrijednost funkcije cilja populacije porasla s 0.4828 na 0.5212.

rbr	d	kazna	kromosom	x	f(x)	križanje
0	**0.15**	**0.54**	**01100010011111101111110001010100**	**-23.0531**	**0.3880**	**4+8**
1	**0.48**	**0.21**	**01100011101110011101011011000**	**-22.0891**	**0.7215**	**12+5**
2	**0.12**	**0.58**	**01110110101111101111001001101010**	**-7.2328**	**0.3543**	**7+11**
3	0.13	0.56	10011110101110000110010000001010	24.0002	0.3712	
4	0.25	0.44	10101111001100100010010100100100	36.8718	0.4925	
5	0.61	0.08	10001111110110100011100100010110	12.3847	0.8513	
6	**0.38**	**0.31**	**10100011101111110010010101110000**	**27.9271**	**0.6213**	**4+8**
7	0.28	0.42	11010100101110010111010100001110	66.1910	0.5157	
8	0.26	0.43	11100001101100011101010001001100	76.3239	0.4970	
9	**0.28**	**0.41**	**00101111001101000011110000101010**	**-63.1218**	**0.5168**	**11+15**
10	**0.27**	**0.42**	**00100011011101011111100000101000**	**-72.2962**	**0.5129**	**5+3**
11	0.10	0.59	10011010101100100000010101001010	20.8558	0.3436	
12	0.69	0.00	10000011101100110000010101100100	2.8901	0.9308	
13	**0.25**	**0.44**	**00101110001101100010010000000110**	**-63.8973**	**0.4897**	**11+3**
14	**0.25**	**0.44**	**00100110001101100011010000100100**	**-70.1471**	**0.4932**	**3+12**
15	0.00	0.69	10000101011001001000111101110010	4.2131	0.2386	

Tablica 6.4: Populacija u drugoj iteraciji prije križanja

Već u drugoj iteraciji postiže se poboljšanje. Jedinka s rednim brojem 1 dobivena križanjem jedinki 12 i 5 postaje najbolja. Neke jedinke (2,6,10 i 13) koje su bile pošteđene eliminacije u prvoj iteraciji sada su eliminirane.

Do 278. generacije najbolja jedinka populacije predstavljala je prvi lokalni optimum, tj. maksimum: `01111100000000000000000000000000` (x=-3.12499997744). U 279-toj generaciji pronađeno je područje globalnog optimuma. Preostali dio procesa do 338. generacije je fino podešavanje rješenja. I konačno u 338. generaciji pronađeno je rješenje (globalni optimum): `10000000000000000000000000000000` (x≅0).

rbr	d	kazna	kromosom	x	f(x)	križanje
0	0.39	0.45	11100001001100001011010000100100	75.9299	0.5053	
1	0.84	0.00	10000111111100110001000100000110	6.2105	0.9587	
2	**0.52**	**0.32**	**10011100101110000100010101001110**	**22.4373**	**0.6388**	**3+10**
3	0.26	0.59	10011110101110000110010000001010	24.0002	0.3712	
4	0.38	0.47	10101111001100100010010100100100	36.8718	0.4925	
5	0.74	0.11	10001111110110100011100100010110	12.3847	0.8513	
6	**0.32**	**0.52**	**10101101101100011101010100000100**	**35.6990**	**0.4368**	**10+5**
7	**0.40**	**0.44**	**11010100101110010111010100001110**	**66.1910**	**0.5157**	**3+0**
8	**0.38**	**0.46**	**11100001101100011101010001001100**	**76.3239**	**0.4970**	**15+3**
9	**0.10**	**0.74**	**10000001011001011000011101000010**	**1.0911**	**0.2137**	**12+1**
10	0.25	0.60	10001110110110000110111000011110	11.5980	0.3610	
11	**0.23**	**0.62**	**10011010101100100000010101001010**	**20.8558**	**0.3436**	**0+1**
12	0.82	0.03	10000011101100110000010101100100	2.8901	0.9308	
13	**0.25**	**0.60**	**10011110101100000110010100001010**	**23.9758**	**0.3636**	**10+1**
14	**0.53**	**0.31**	**10011111101110010000010100110010**	**24.7834**	**0.6469**	**15+3**
15	0.00	0.84	10001101111001001000111100110010	10.8538	0.1158	

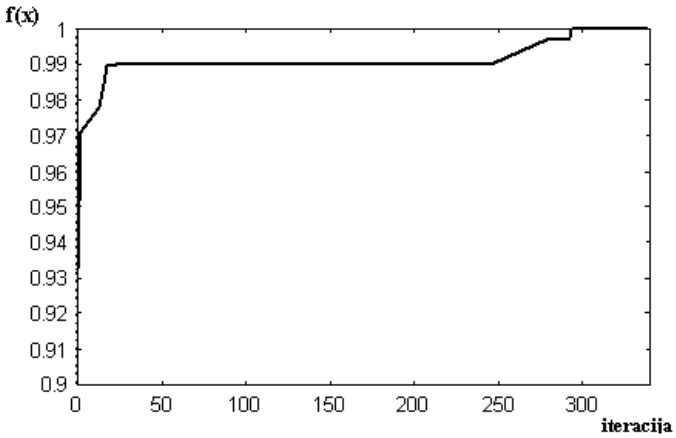
Tablica 6.5: Populacija u trećoj iteraciji prije križanja

iteracija	x	f(x)	kromosom
0	-6.09471749191	0.93219392743	01111000001100101110001000001010
2	-2.99842290184	0.97113253692	01111100001010010111101000001110
13	-3.02607852570	0.97792283178	01111100001000000110101000100010
14	-3.04753526651	0.98218118643	01111100000110010110001000110110
17	-3.12077973576	0.98997638820	01111100000000010110001000000101
23	-3.12135082277	0.98999591769	01111100000000010011001000011101
24	-3.12135110216	0.98999592708	01111100000000010011001000010111
26	-3.12137457149	0.98999671596	01111100000000010011000000011111
27	-3.12154258208	0.99000233173	01111100000000010010001000000111
31	-3.12364015335	0.99006777924	01111100000000000111001000010010
32	-3.12430553677	0.99008673589	01111100000000000011101001000001
33	-3.12440393100	0.99008946536	01111100000000000011001000000000
35	-3.12454623709	0.99009337933	01111100000000000010011000010000
40	-3.12468700649	0.99009721193	01111100000000000001101001000001
42	-3.12468924167	0.99009727247	01111100000000000001101000010001
45	-3.12468998673	0.99009729265	01111100000000000001101000000001
46	-3.12497590276	0.99010495561	01111100000000000000001000000101
49	-3.12497608902	0.99010496055	01111100000000000000001000000001
53	-3.12498782368	0.99010527162	01111100000000000000000100000101
54	-3.12499974461	0.99010558735	01111100000000000000000000000101
55	-3.12499993088	0.99010559229	01111100000000000000000000000001
246	-3.12499997744	0.99010559352	01111100000000000000000000000000
279	0.05464351272	0.99701408914	10000000000100011110011111010100
288	0.05268848037	0.99722373129	10000000000100010100001111010100
293	0.04963653629	0.99753578537	10000000000100000100001111010000
294	0.00081167091	0.99999934053	10000000000000000100010000010110
305	0.00001883600	0.99999999964	10000000000000000000000110010100
308	0.00001296867	0.99999999983	10000000000000000000000100010110
309	0.00001287553	0.99999999983	10000000000000000000000100010100
316	0.00000700820	0.99999999995	10000000000000000000000010010110
318	0.00000104774	1.00000000000	10000000000000000000000000010110
326	0.00000086147	1.00000000000	10000000000000000000000000010010
332	0.00000011642	1.00000000000	10000000000000000000000000000010
338	0.00000002328	1.00000000000	10000000000000000000000000000000

Tablica 6.6: Primjer cijelog evolucijskog procesa

Slika 6.2 je grafički prikaz eksperimentalnih rezultata iz tablice 6.6. Na x osi je broj iteracije ili generacije, a na y osi je vrijednost funkcije cilja koja se dobije evaluacijom najbolje jedinke u toj iteraciji. Na toj slici je zorno prikazan prijelaz s lokalnog optimuma *(f(x)≅0.99)* na globalni *(f(x)=1)* koji se dogodio oko 250-te iteracije.

Svi duplikati se ne eliminiraju u svakoj iteraciji. Detekcija i eliminacija duplikata odvija se u procesu križanja. Ako su roditelji isti, tada se jedan od roditelja mutira, a dijete je rezultat slučajnog procesa (kao kod inicijalizacije). Na taj način proširuje se prostor pretrage.



Slika 6.3: Genetski algoritam koji iz generaciju u generaciju čuva najbolju jedinku asimptotski teži ka globalnom optimumu

6.3 Određivanje koeficijenata trigonometrijskih aproksimacijskih funkcija

Zadan je skup točaka $T = \{(x_1, y_1),(x_2, y_2),(x_3, y_3),....,(x_P, y_P)\}$ i funkcija *g(x)* s neodređenim koeficijentima $a_0,a_1,a_2,...a_{N-1}$, gdje je $a_i \in [dg_i, gg_i]$. Ako je za zadani skup točaka os *x* vrijeme (*x=t*), tj. zadan je niz vrijednosti u određenim vremenskim trenucima, tada je skup točaka *T* skup vrijednosti u diskretnim vremenskim trenucima i naziva se *vremenski niz*, odnosno $T = \{(t_1, y_1),(t_2, y_2),(t_3, y_3),....,(t_P, y_P)\}$. Zadatak je odrediti s pomoću genetskog algoritma *N* koeficijenata a_0 do a_{N-1} tako da kvadrat odstupanja funkcije *g(t)* u zadanim točkama bude minimalan. Za genetski algoritam to je N-dimenzijski problem i u kromosomu je zapisan vektor koeficijenata $\vec{a}$. Funkcija cilja *f* koju treba minimizirati glasi:

$$f(\vec{a}) = f(a_0, a_1,....,a_{N-1}) = \sum_{i=1}^{P} (g(x_i) - y_i)^2 \,. \qquad (6.1)$$

Koeficijenti $a_0,a_1,a_2,...a_{N-1}$ koji određuju funkciju *g(t)* su nepoznanice za funkciju cilja *f*. Rješenje zadanog problema je jedna ili više točaka u *N* dimenzijskom prostoru.

Optimizacijski problem koji treba riješiti glasi:

$$\min\{f(a_0, a_1, ..., a_{N-1})\} = ?, \quad (6.2)$$

uz ograničenja:

$$a_i \in [dgi, ggi]. \quad i=0,1,2,...N-1. \quad (6.3)$$

Neka je zadan je vremenski niz; skup od 20 točaka:

T = { (1,1),(4,50),(5,51),(7,52),(9,60),(11,71),(14,76),(15,73),(16,79),(17,68),(20,55),
(22,57),(25,80),(27,82),(29,78),(31,100),(34,90),(36,85),(38,80),(40,78) }

Funkcija *g* kojoj treba odrediti 20 koeficijenta $a_0, a_1, a_2, ... a_{19}$ glasi:

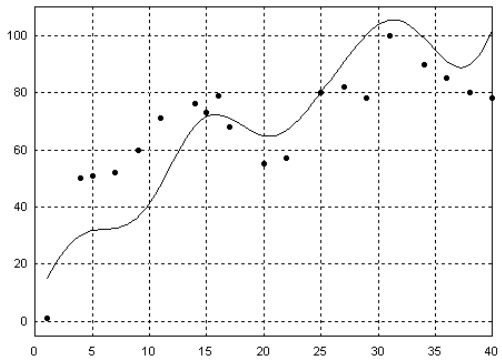
$$a_0 + a_1 x + \sum_{i=1}^{6} \left(a_{3i-1} \sin(a_{3i} x + a_{3i+1})\right) \quad (6.4)$$

$$= a_0 + a_1 t + a_2 \sin(a_3 t + a_4) + a_5 \sin(a_6 t + a_7) + a_8 \sin(a_9 t + a_{10}) + a_{11} \sin(a_{12} t + a_{13}) + a_{14} \sin(a_{15} t + a_{16}) + a_{17} \sin(a_{18} t + a_{19}).$$

Funkcija *g* je suma konstantnog člana (a_0), linearnog člana ($a_1 t$) te 6 sinus članova određenih amplitudama ($a_2, a_5, a_8, a_{11}, a_{14}$ i a_{17}), frekvencijama ($a_3, a_6, a_9, a_{12}, a_{15}$, i a_{18}) i faznim pomacima ($a_4, a_7, a_{10}, a_{13}, a_{16}, a_{19}$). Koeficijenti se nalaze unutar zadanih granica:

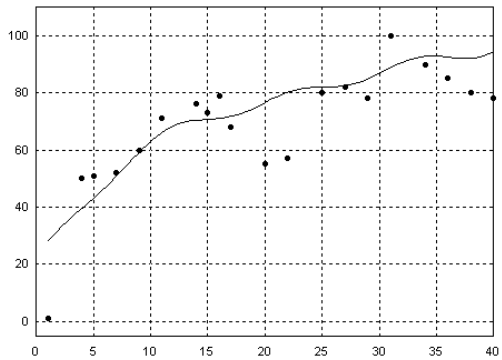
$$a_0, a_1, a_2, a_5, a_8, a_{11}, a_{14}, a_{17} \in [-10, 10],$$
$$a_3, a_6, a_9, a_{12}, a_{15}, a_{18} \in [0, 2],$$
$$a_4, a_7, a_{10}, a_{13}, a_{16}, a_{19} \in [-5, 5].$$

6.3.1 Eksperimentalni rezultati



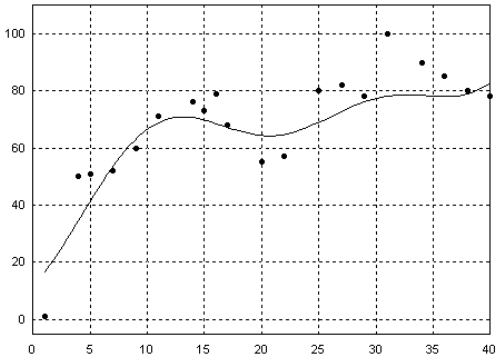
Slika 6.4: Rješenje u 21. iteraciji
g(t)=8.055+2.937·t-2.361·sin(0.644·t-4.445)-
-9.070·sin(0.440·t-2.338)-
-9.994·sin(0.113·t+3.791)+
+4.818·sin(0.322·t-1.244)+
+5.167·sin(0.140·t+4.744)-
-1.738·sin(0.349·t+0.106)

$f(\vec{a}) = 4076.8$



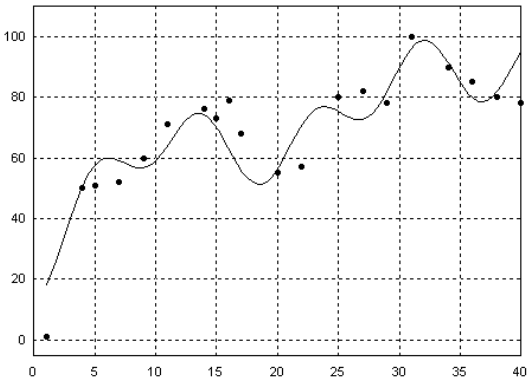
Slika 6.5: Rješenje u 62. iteraciji
g(t)=9.221+2.614·t-1.905·sin(0.577·t-2.007)+
+2.251·sin(0.188·t+0.006)-
-9.810·sin(0.081·t+3.802)+
+8.684·sin(0.092·t-0.224)-
-9.733·sin(0.077·t+4.590)-
-1.421·sin(0.290·t+1.978)

$f(\vec{a}) = 2690.1$



Slika 6.6: Rješenje u 76. iteraciji
g(t)=9.304+2.653·t+2.939·sin(0.140·t+0.505)-
-8.764·sin(0.188·t-4.843)-
-9.816·sin(0.084·t+3.956)-
-2.565·sin(0.076·t-0.926)-
-9.735·sin(0.078·t-1.543)-
-9.545·sin(0.290·t+1.964)

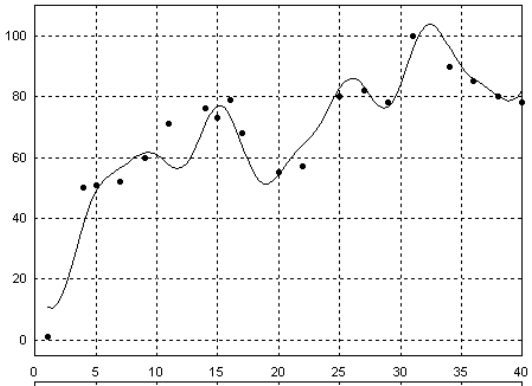
$f(\vec{a}) = 1780.7$



Slika 6.7: Rješenje u 93. iteraciji

$g(t)=7.892+2.653\cdot t+4.160\cdot\sin(0.139\cdot t+0.543)-$
$-8.755\cdot\sin(0.690\cdot t+1.422)-$
$-9.191\cdot\sin(0.084\cdot t+3.958)-$
$-3.816\cdot\sin(0.009\cdot t-0.906)-$
$-9.828\cdot\sin(0.070\cdot t-1.582)-$
$-9.545\cdot\sin(0.298\cdot t+1.969)$

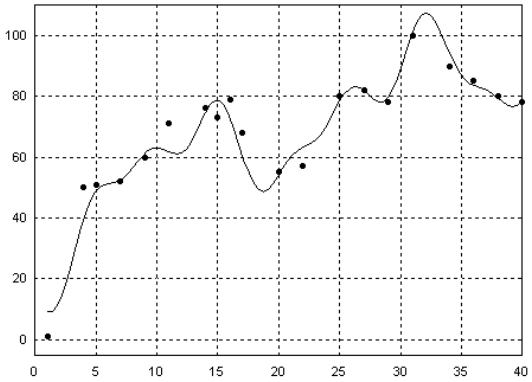
$f(\vec{a})=1491.3$



Slika 6.8: Rješenje u 234. iteraciji

$g(t)=9.299+2.643\cdot t+4.189\cdot\sin(1.139\cdot t+3.005)-$
$-8.755\cdot\sin(0.703\cdot t+0.162)-$
$-9.817\cdot\sin(0.086\cdot t+3.960)-$
$-2.566\cdot\sin(0.327\cdot t-0.066)-$
$-9.987\cdot\sin(0.078\cdot t-1.856)-$
$-9.545\cdot\sin(0.305\cdot t+1.883)$

$f(\vec{a})=651.8$



Slika 6.9: Rješenje u 1000. iteraciji

$g(t)=9.963+2.584\cdot t+5.000\cdot\sin(1.141\cdot t+2.987)-$
$-7.502\cdot\sin(0.703\cdot t+0.475)-$
$-9.661\cdot\sin(0.086\cdot t+3.960)-$
$-4.599\cdot\sin(0.344\cdot t-0.027)-$
$-10.000\cdot\sin(0.076\cdot t-1.857)-$
$-9.961\cdot\sin(0.302\cdot t+1.878)$

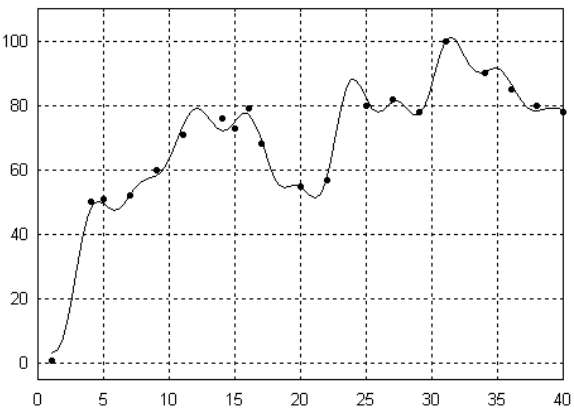
$f(\vec{a})=446.7$

6.3.2 Zapažanja

Povećanjem broja iteracija postiglo se jedva zamjetno poboljšanje učinkovitosti algoritma. Genetski algoritam i dalje pronalazi aproksimacijsku funkciju koja prolazi kroz ili blizu oko četrnaest od dvadeset točaka. Za takvo loše ponašanje genetskog algoritma moguća su tri razloga:

1. Aproksimacijska funkcija ima suviše malo sinus članova.
2. Parametri genetskog algoritma nisu dobro podešeni.
3. Postavljene su krive granice koeficijenata.

Otklanjanjem svih triju navedenih uzroka poboljšala se učinkovitost genetskog algoritma.



Slika 6.10: Rješenje dobiveno za aproksimacijsku funkciju koja ima više (9 umjesto 6) sinus članova. Broj iteracija je postavljen na 5 000.

$g(t)=10.000+2.500\cdot t-10.000\cdot\sin(0.093\cdot t-2.540)+$
$+1.249\cdot\sin(1.849\cdot t+1.757)+$
$+8.032\cdot\sin(0.656\cdot t-1.680)-$
$-3.869\cdot\sin(0.375\cdot t+0.503)+$
$+9.961\cdot\sin(0.297\cdot t+4.764)-$
$-2.495\cdot\sin(0.004\cdot t+4.681)+$
$+5.000\cdot\sin(0.938\cdot t+4.063)-$
$-4.286\cdot\sin(1.609\cdot t-1.855)-$
$-10.000\cdot\sin(0.106\cdot t-2.672)$

$f(\vec{a})=56.4$

Povećanjem broja iteracija, genetski algoritam postiže bolje rezultate, ali proporcionalno povećanju iteracija povećava se i vrijeme izvođenja algoritma. Povećanjem dimenzije problema, genetski algoritam u načelu postiže slabije rezultate, jer ima za odrediti više nepoznanica. Međutim, mada je povećana dimenzija problema od 20 na 29

(povećan je broj sinus članova sa 6 na 9) i mada je smanjen broj iteracija s 50 000 na 5 000, rezultati optimiranja su znatno bolji. Razlog takvom poboljšanju učinkovitosti algoritma leži u povećanju broja sinus članova aproksimacijske funkcije.

Povećanjem broja bitova kod binarnog prikaza postigli su se bolji rezultati, mada su se očekivali lošiji. Naime, za genetski algoritam je isto da li se poveća broj bitova ili dimenzija problema Nakon analize ustanovljeno je da razlog takvom ponašanju algoritma leži u vjerojatnosti mutacije. Vjerojatnost mutacije jednog bita p_m proporcionalna je vjerojatnosti mutacije jedinke p_M i vjerojatnosti mutacije jednog gena p_b:

$$p_m = p_M \cdot p_b , \tag{6.5}$$

gdje p_M ovisi o parametru *BR_M* algoritma (broj jedinki za eliminaciju):

$$p_M = \frac{BR\_M}{VEL\_POP} \tag{6.6}$$

i iznosi 0.25.

Vjerojatnost mutacije jednog gena p_b odabrane jedinke za mutaciju je kvocijent broja mutiranih gena m i ukupnog broja bitova b_i koji predstavljaju vrijednost jedne varijable:

$$p_b = \frac{m}{b_i} . \tag{6.7}$$

Za 20 bitova po varijabli, vjerojatnost mutacije jednog bita odabrane jedinke za mutaciju, iznosi 2/20=0.1 (1 do 3 bita se mutiraju - dakle prosječno 2 bita po mutaciji). I konačno, vjerojatnost mutacije gena p_m iznosi 0.25*0.1=0.025. Za 26 bitova po varijabli kod binarnog prikaza vjerojatnost mutacije gena je manja i iznosi 0.25*2/26=0.0192.

Značajnijom promjenom vjerojatnosti mutacije smanjuje se učinkovitost algoritma. Na primjer, postavljen je broj mutacija na *VEL_POP/20* ili $p_M=0.05$ ($p_m=0.0038$) i genetski algoritam je počeo konvergirati već u nekoliko desetaka iteracija prema lokalnom minimumu. Iz tog minimuma nije se više mogao izvući upravo zbog suviše male vjerojatnosti mutacije. Druga krajnost je suviše velika vjerojatnost mutacije. Na primjer, vjerojatnost mutacije jedinke postavljena je na $p_M=0.8$ ($p_m=0.062$) i proces optimiranja se gotovo pretvorio u slučajno pretraživanje prostora rješenja. Domena se sastoji od 2^{bN} točaka, što u ovom konkretnom slučaju iznosi $2^{26*29}=2^{754}$. Dakle, ako postoji samo jedno rješenje, vjerojatnost pronalaženja tog rješenja slučajnim procesom iznosi 2^{-754}.

7. ZAKLJUČAK

7.1 Stupnjevi slobode

Za određeni optimizacijski problem koji želimo riješiti pomoću GA potrebno je odrediti slijedeće:

- funkciju dobrote ili kazne
- selekciju (jednostavna, turnirska, eliminacijska, ...)
- predstavljanje (binarno, s bazom većom od 2, FP, ...)
- definirati uvjet završetka evolucijskog procesa (vrijeme, uniformnost,...)
- generacijski procjep (koliko se jedinki direktno prenosi iz generacije u generaciju)
- genetske operatore
    - ◊ križanje (s jednom ili više točaka prekida, uniformno, posebno definirano...)
    - ◊ mutacija (jednostavna, miješajuća, invertirajuća,...)
- parametre
    - ◊ veličina populacije
    - ◊ vjerojatnost križanja
    - ◊ vjerojatnost mutacije
    - ◊ broj jedinki za eliminaciju kod eliminacijske selekcije

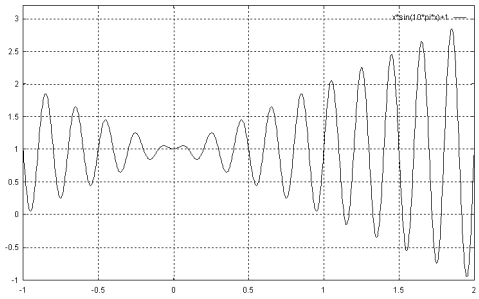
7.2 Za i protiv GA

ZA	PROTIV
♦ funkcija *f* koju treba optimizirati je potpuno proizvoljna, tj. nema posebnih zahtjeva kao što su neprekinutost, derivabilnost i sl.	◊ teško se definira *dobra* funkcija dobrote na temelju *f* ◊ potrebno je prilagoditi GA zadanim ograničenjima ◊ problem decepcijske funkcije
♦ primjenjiv je na veliki broj problema	◊ često je potrebna prilagodba problema algoritmu
♦ struktura algoritma nudi velike mogućnosti nadogradnje i povećanja efikasnosti algoritma jednostavnim zahvatima (puno stupnjeva slobode)	◊ teško je postaviti dobre parametre (velik utjecaj parametara na efikasnost)
♦ jednostavnim ponavljanjem postupka se može povećati pouzdanost rezultata ♦ ako već ne nađe rješenje (globalni optimum), daje nekakvo *dobro* rješenje koje može zadovoljiti ♦ kao rezultat daje skup rješenja, a ne jedno rješenje	◊ ne može se postići 100% pouzdanost rješenja ◊ konvergencija je znatno sporija od ostalih numeričkih metoda
♦ rješava sve probleme koji se mogu predstaviti kao optimizacijski, bez obzira da li funkcija *f* koju treba optimizirati ima za argumente realne brojeve ili bitove ili znakove ili bilo koju vrstu informacije ♦ vrlo jednostavno je primjenjiv na višemodalnim (višedimenzijskim) problemima	◊ potrebno je posebno definirati genetske operatore za posebne vrste prikaza ◊ zbog stohastičnosti nikad ne znamo prirodu nađenog rješenja
♦ jednostavnost ideje i dostupnost programske podrške	◊ zbog izvođenja velikog broja računskih operacija GA je spor, traži se velika procesorska snaga

PRILOG: 1D ispitne funkcije

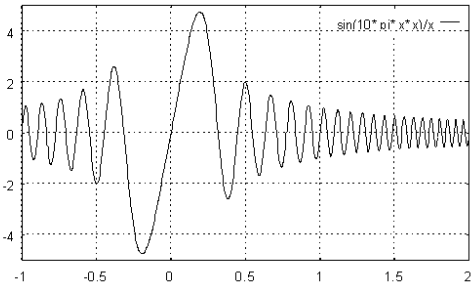
$$f_0(x) = 1 + x\sin(10\pi x)$$

$$\max\{f_0(x)\} = f_0(1.85047) = 2.850274$$



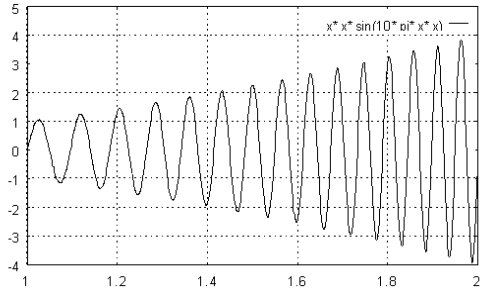
$$f_1(x) = \frac{\sin(10\pi x^2)}{x}$$

$$\max\{f_1(x)\} = f_1(0.19262) = 4.771199$$



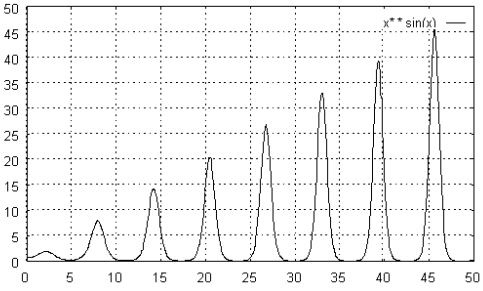
$$f_2(x) = x^2\sin(10\pi x^2)$$

$$\max\{f_2(x)\} = f_2(1.96221) = 3.8650132$$



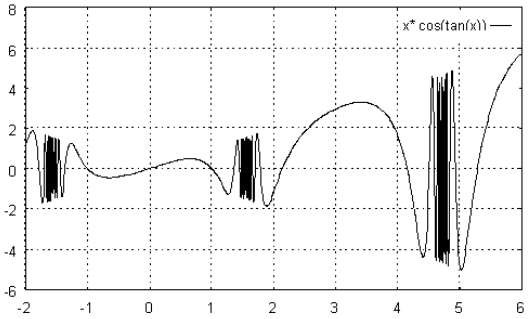
$$f_3(x) = x^{\sin(x)}$$

$$\max\{f_3(x)\} = f_3(45.55883) = 45.555967$$



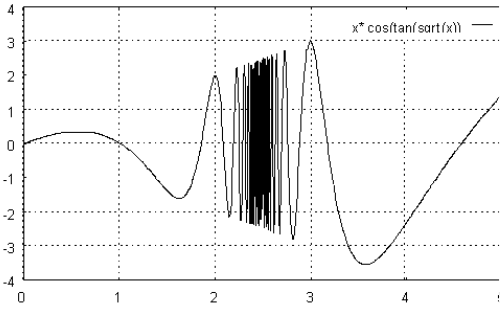
$$f_4(x) = x\cos[\tan(x)]$$

$$\max\{f_4(x)\} = f_4(6) = 5.747734$$



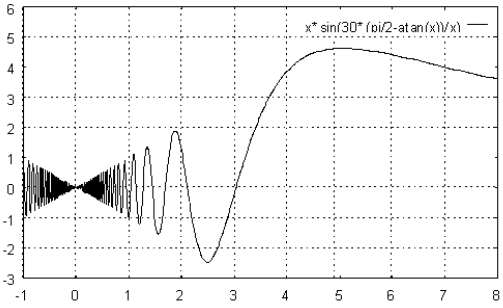
$$f_5(x) = 0.5 - \frac{\sin^2(x) - 0.5}{\left(1 + 0.001 * x^2\right)^2}$$

$$\max\{f_5(x)\} = f_7(0) = 1$$



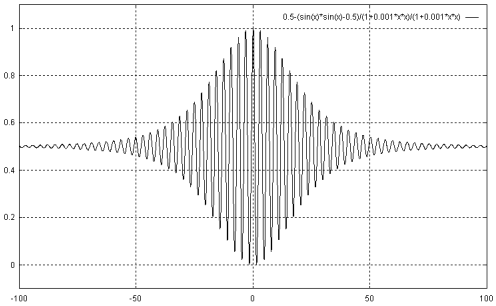
$$f_6(x) = x\sin\left\{\frac{30}{x}\left[\frac{\pi}{2} - \arctan(x)\right]\right\}$$

$$\max\{f_6(x)\} = f_6(5.05567) = 4.632549$$



$$f_7(x) = 0.5 - \frac{\sin^2(x) - 0.5}{\left(1 + 0.001 * x^2\right)^2}$$

$$\max\{f_7(x)\} = f_7(0) = 1$$



LITERATURA

[1] -, An American National Standard: IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.

[2] L. Davis, *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, New York, 1991.

[3] J.L.R. Filhio, P.C. Treleaven, C. Alippi, *Genetic-Algorithm Programming Environments,* Computer, June 1994., 28-43

[4] P. Field, *A Multary Theory for Genetic Algorithms: Unifyng Binary and Nonbinary Problem Representations*, University of London, 1995., dostupno na ftp adresi: ftp.dcs.qmw.ac.uk/applied_logic/pg/Field/MultaryTheory.ps.Z

[5] A.S. Glassner, *Principles of Digital Image Synthezis*, Morgan Kaufmann Pubishers, San Francisco, California, 1995., 243-298

[6] A. Graps, *An introduction to Wavelets*, IEEE Copmputional Science & Engineering, Summer 1995.

[7] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[8] I. Ivanšić, *Fourierov red i integral,* Sveučilišna naklada Liber, Zagreb, 1987

[9] T. C. Jones, *One Operator, One Landscape*, Twelfth International Conference on Machine Learning, Santa Fe Institute, Santa Fe, siječanj 1995., dostupno na ftp adresi: ftp.santafe.edu:pub/terry/oool.ps.gz

[10] T. C. Jones, *A description of Holland's royal road function,* Technical Report 94-11-059, Santa Fe Institute, Santa Fe, NM, prosinac 1994., dostupno na ftp adresi: ftp.santafe.edu:pub/terry/jhrr.tar.gz

[11] T. C. Jones, S. Forrest, *Genetic Algorithm and Heuristic Search*, International Joint Conference on Artificial Intelligence, Santa Fe Institute, Santa Fe, siječanj 1995.

[12] T. C. Jones, G.J.E. Rawlins *Reverse Hillclimbing, Genetic Algorithms and the Busy Beaver Problem, U S. Forrest*, urednik, Genetic Algorithms: Proceedings of the Fifth International Conference (ICGA 1993.), 70-75, San Mateo, CA, 1993. Morgan Kaufmann

[13] P.J.M. van Laarhoven, E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel Publishing Company, 1987.

[14] V. Maniezzo, *Genetic Evolution of the Topology and Weight Distribution of Neural Networks,* IEEE Transactions on Neural Networks, Vol. 5. No. 1, siječanj 1994.

[15] *Medicinska enciklopedija*, Leksikografski zavod, Zagreb, 1967.

[16] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, 1994.

[17] José L. Ribeiro Filho, Philip C. Treleaven, Cesare Alippi: *Genetic-Algorithm Programming Environments*, IEEE Computer, June 1994.

[18] R. Sauer, I. Szabo, *Mathematische Hilfsmittel des Ingenieurs,* Springer-Verlag, Berlin und Heidelberg, 1968.

[19] E. Schoeneburg, F. Heinzmann, S. Feddersen, *Genetische Algorithmen und Evolutionsstrategien,* Addison-Wesley, 1995.

[20] M. Srinivas, L.M. Patnaik, *Genetic Algorithms: A Survey*, Computer, June 1994., 17-26

[21] M. Srinivas, L. M. Patnaik: *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*, IEEE Trans. Systems, Man and Cybernetics, Apr. 1994.

[22] J. Stouer, R. Bulirsch, *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1976.

[23] E.J. Stollnitz, T.D. DeRose, D.H. Salesin, *Wavelets for Computer Graphics: A Primer, Part 1 & 2,* IEEE Computer Graphics and Applications, May and July 1995.

[24] S. Turk, L. Budin, *Analiza i projektiranje računalom*, školska knjiga Zagreb, 1989.