PRODUŽENI KOMPAKTNI GENETSKI ALGORITAM (ECGA)

EXTENDED COMPACT GENETIC ALGORITHM (ECGA)

Stručni članak

Prof. VŠ dr. Hidajet Salkić[*]
Pred. VŠ mr. Nermin Palić[*]

Sažetak

U ovom radu je predstavljen genetski algoritam kao optimizacijski metod nalaženja globalnog minimuma ili maksimuma funkcija pri matematičkom modeliranju tehničkih sistema i procesa. Optimizacijski metod za nalaženje mimimuma i maksimuma su logički ekvivalenti, a optimizacijske tehnike mogu biti isto korisne ako se zahtijeva ili minimumi ili nalaženje maksimuma. Nema optimizacijskog algoritma koji može garantovati nalaženje dobrog rješenja problema ako funkcija pogodno ne reflektuje rješenje.

Popularnost Genetskog algoritma motivirana je brojnim faktorima koji uključuju:

- *Evolucija je poznato uspješna, robusan metod za adaptaciju unutar bioloških sistema;*
- *Genetski algoritmi mogu pretraživati prostor koji sadrži kompleksno uzajamne dijelove gdje uticaj svakog dijela nad pogodnost hipoteze može biti težak za modeliranje.*

Genetski algoritmi su jednostavni za paralelizaciju i mogu iskoristiti prednost u opadanju cijene računarske snage.

Ključne riječi: uniformno ukrštanje, šema teorem, nagomilavanje, genetski algoritam.

Abstract

In this article the genetic algorithm has been presented as the optimization method of the research of global minimum or maximum of functions during matemathic modeling of technical systems and processes.

[*] *Visoka škola „CEPS – Centar za poslovne studije“, JP Elektroprivreda BiH*
E-mail: h.salkic@elektroprivreda.ba
[*] *Visoka škola „CEPS – Centar za poslovne studije“ Kiseljak*
E-mail: nermin.visokaskola@gmail.com

The optimization method for finding the mimimum and the maximum are logical equivalents, and the optimization techniques can also be useful if either a minimum or a maximum is required. There is no optimization algorithm that can guarantee finding a good problem solution if the function does not adequately reflect the solution.

Popularity of Genetic algorithm has been motivated by numerous factors that included:

- *Evolution is very well successful, robust method for adaptation of inner biological systems*
- *Genetic algorithms are able to search space that includes complex mutual parts where influence of each part to over the convenience of a hypothesis can be difficult for modeling.*

Genetic algorithms are simple for parallelization and could use advantage of decreasing computer's power price.

Keywords: uniform crossing, schema theorem, accumulation, genetic algorithm.

1. UVOD

Optimizacija po sebi je više fleksibilna nego što je prepoznata kao takva. Problem optimizacije može se definisati ovako:

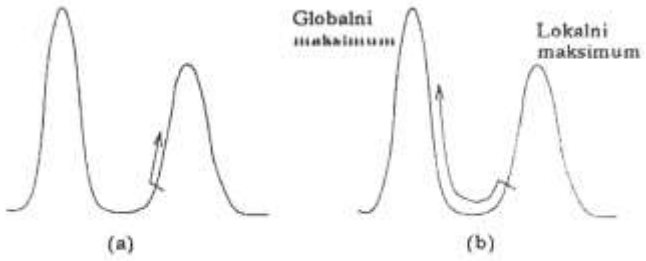
Za proizvoljno zadati konačan domen D i funkciju f: D→ R, R je skup realnih brojeva. Potrebno je pronaći najbolju vrijednost u D. Nalaženje najbolje vrijednosti u D podrazumijeva nalaženje x ϵ D takoda vodi minimumu (minimizacija funkcije) ili maksimumu (maksimiziranje funkcije) funkcije **f**:

$$f_{\min}(x) = \min_{x \varepsilon D} f(x)$$
$$f_{\max}(x) = \min_{x \varepsilon D} f(x) \qquad (1.1)$$

Optimizacijski metod za nalaženje mimimuma i maksimuma su logički ekvivalenti, a optimizacijske tehnike mogu biti isto korisne ako se zahtijeva ili minimumi ili nalaženje maksimuma. Nema optimizacijskog algoritma koji može garantovati nalaženje dobrog rješenja problema ako funkcija pogodno ne reflektuje rješenje. Dizajn funkcije je ključni faktor u izvođenju bilo kojeg optimizacijskog algoritma (slično, odgovarajuća selekcija karakteristika je neophodna za uspješno klasificiranje).

Većina konvencionalnih pristupa optimizaciji koristi metode bazirane na izračunavanju koje se mogu uporediti sa penjanjem uz brežuljak (u

slučaju maksimuma) – gradijent funkcije daje strmiji smjer penjanja. Glavni limit ovakvih metoda je njihovo lokalno ponašanje. Pretraživanje može lahko završiti u lokalnom maksimumu, a globalni maksimum se može promašiti (slika 1.)



Slika 1. Ograničenje metoda penjanjem uz brežuljak

Nekoliko metoda povečava vjerovatnoću nalaženja globalnog minimuma: početak penjanja od različiti tačaka u prostoru pretraživanja, primjenjivanje enumerativnog pretraživanja kao što je dinamičko programiranje, primjena slučajnog pretraživanja itd. Među ovim mogućnostima su genetski algoritmi.

2. GENETSKI ALGORITMI

Genetski algoritmi omogućavaju pristup učenju koje je bazirano na simuliranoj evoluciji. Hipoteze su obično opisane stringovima čija interpretacija zavisi od primjene, iako hipoteze mogu takođe biti opisane simboličkim izrazima ili čak kompjuterskim programom. Nalaženje odgovarajućih hipoteza počinje sa populacijom ili kolekcijom inicijalnih hipoteza. Članovi trenutne populacije rastu u slijedeću generaciju populacije, poput operacija kao što je slučajna mutacija ili ukrštanje, koje su uzor biološkoj evoluciji. U svakom koraku, hipoteze u trenutnoj populaciji su evaulirane relativno datom mjerom sposobnosti sa najpogodnijim hipotezama slučajno odabrane, kao sjeme za produkciju nove generacije. Genetski algoritam primjenjuje se uspješno za optimizaciju parametara učenja umjetnih neuronskih mreža. Oni omogučavaju metod učenja motivisan analogno biološkoj evoluciji.

Umjesto da koristi nalaženje hipoteza opšte u specifično, ili jednostavno u složeno, genetski algoritam generišu uspješne hipoteze uzastopno mutirajući i kombinujući dijelove od najboljih trenutno poznatih hipoteza. U svakom koraku kolekcija hipoteza nazvana trenutna populacija ažurira se zamjenom nekog malog dijela populacije sa potomstvom najpogodnijih trenutnih hipoteza. Proces formira generiši - i – testiraj

usmjereno – pretraživanje hipoteza, u kojem odstupanja od najpogodnije trenutne hipoteze su ta koja će se razmatrati kao sljedeća.

Problem povezan sa genetskim algoritmom je pretraživanje prostora od kandidiranih hipoteza da se identificira najbolja hipoteza.U genetskom algoritmu "najbolja hipoteza" definisana je ona koja optimizira predefinisanu numeričku mjeru za problem, nazvanu podobnost hipoteze. Naprimjer, ako je problem učenja aproksimacija nepoznate funkcije zadate primjerima učenja njenih ulaza i izlaza, onda podobnost se može definsati kao tačnost hipoteza na primjerima učenja. Ako je zadatak učenja strategija igranja šaha, podobnost može biti definisana kao broj individualno dobijenih igara odigranih protiv drugih u trenutnoj populaciji. Iako različite implementacije genetskih algoritama variraju u detaljima, oni tipično dijele sljedeću strukturu: algoritam radi iterativno ažuriranje grupe hipoteza nazvane populacija. U svakoj iteraciji svi članovi populacije se ocjenjuju u skladu sa funkcijom pogodnosti. Zatim se generiše nova populacija slučajnim odabiranjem podobnijih individua iz trenutne generacije. Neke od ovih odabranih individua se prosljeđuju u sljedeću generaciju netaknute. Druge se koriste kao baza za kreiranje potomstva primjenjujući genetske generacije kao što su ukrštanje i mutacija. Prototip genetskog algoritma je:

GA (F, F_t, p, r , m)

F: Funkcija koja pridružuje vrijednost vrednovanja datoj hipotezi
Ft: Prag koji specificira kriterij prekida
p: Broj hipoteza uključenih u populaciju
r: Dio populacije koji se zamjenjuje ukrštanjem
m: Odnos mutacije

- Inicijalizacija populacije: P ← generiše p hipoteza slučajni.
- Vrjednovanje : za svako h iz P, izračunava se F(h)
- While [max_h F(h)] < F_t, do kreiranja nove populacije P_s :

Selekcija: Odabir vjerovatnoćom (1-r)p članova populacije P koji se dodaju u P_s. Vjerovatnoća P_r (h_t) odabrane hipoteze h_t, iz P data je jednačinom (1.2.).

Ukrštanje: Odabire se rp/$_2$ parova iz hipoteze P u skladu sa P_r (h_t). Za svaki par (h_1,h_2), produkuju se dva potomka koristeći operatore ukrštanja. Svi se dodaju u populaciju P_s.

Mutacija: Izabrani m procenata članova P_s sa uniformnom vjerovatnoćom.

Za svaki invertovani jedan slučajno odabrani bit.

Ažuriraj: P ← P_s

Vrednovanje: za svako h iz P , uzračunaj F(h).

- Vratiti hipoteze iz P sa najvećom pogodnosti.

Ulaz u ovaj algoritam uključuje funkciju ocjene podobnosti za rangiranje kandidiranih hipoteza, prag definiše prihvatljivi nivo podobnosti za završetak algoritma,veličina populacije koja će se održavati i parametri koji opisuju koliko će uspješnih populacija biti generisano; mali dio populacije koji će se zamjeniti u svakoj generaciji i nivo mutacije. U ovim algoritmima svaka iteracija kroz glavnu petlju proizvodi novu generaciju hipoteza bazirano na trenutnoj populaciji. Prvo, bitan broj hipoteza iz trenutne populacije odabire se za uključivanje u sledeću generaciju. Odabire se vjerovatnoćom odabiranja hipoteze hi data sa:

$$P(h_i) = \frac{F(h_i)}{\sum_{j=1}^{p} F(h_j)} \qquad (1.2)$$

Prema tome, vjerovatnoća da će hipoteza biti odabrana je proporcionalna njenoj pogodnosti i obrnuto proporcionalna pogodnosti drugih komponentnih hipoteza u trenutnoj populaciji. Jednom kada su članice populacije odabrane za uključivanje u populaciju sljedeće generacije dodatni članovi se generišu korištenjem operacije ukrštanja. Ukrštenje, definisano detaljno u sljedećem paragrafu, uzima dvije roditeljske hipoteze iz trenutne i kreira dvije hipoteze potomstva rekombinujući dijelove oba roditelja. Hipoteze roditelji odabrani su slučajno iz trenutne populacije, koristeći ponovo funkciju vjerovatnoće datu jednačinom (1.2.).

Poslije kreiranja novih članova ovom operacijom ukrštanja nova generacija populacije sada sadrži željni broj članova. U ovoj tački izvjestan dio m ovih članova izabran je slučajno, i slučajno mutacija je izvedena na ovim članovima.

Ovaj genetski algoritam prema tome izvodi se slučajno, paralelno usmjerenim pretraživanjem za hipotezama koje se ponašaju dobro u skladu sa funkcijom pogodnosti.

Hipoteze u genetskim algoritmima su često prestavljene sa stringovima, tako da se s njima može jednostavno manipulisati genetskim operatorima kao što je mutacija i ukrštanje. Hipoteze prestavljene ovakvim bit stringovima mogu biti vrlo kompleksne.

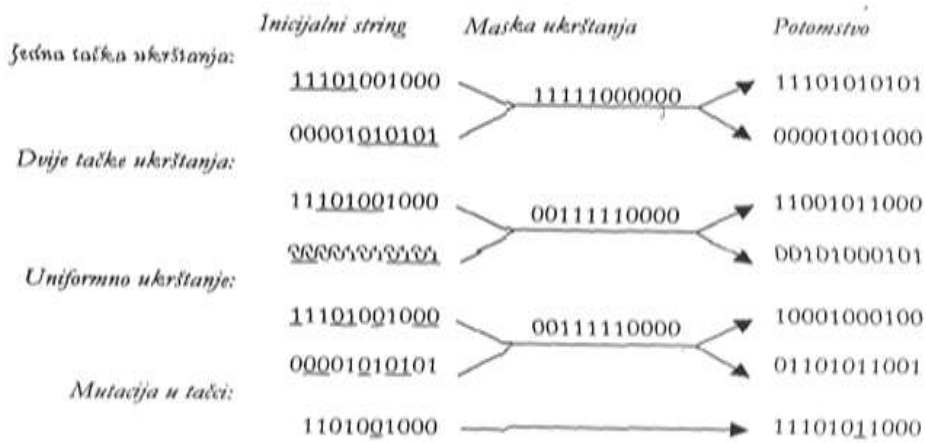
Naprimjer, skup ako-onda pravila mogu biti predstavljeni na ovaj način, izabiranjem pravila alociranjem određenog podstringa za svako pravilo pred-uslova i post-uslova.

String koji prestavlja pravilo sadrži podstring za svaki atribut u prostoru hipoteza, čak ako ovaj atribut nije ograničen pravilom pred-uslova. Ovo vodi fiksnoj dužini bit stringa za predstavljanje pravila u kojem

podstring na određenom mjestu opisuje ograničenja određenog atributa. Datim predstavljanjem za jedno pravilo može se predstaviti skup pravila jednostavno spajajući stringove koji predstavljaju pojedina pravila. U dizajniranju kodiranja bit stringova za neki prostor hipoteza korisno je urediti da svaki sintaksno legalan bit bude prestavljen dobro definisanim hipotezama.

U nekim genetskim algoritmima hipoteze su predstavljene simboličkim opisivanjem umjesto bit stringova.

Generacije uspješnih genetskih algoritama određuju se skupom operatora koji rekonbinuju i mutiraju odabrane članove trenutne populacije. Tipično genetski algoritamski operatori za manipulisanje bit string hipotezama su prestavljeni na slici (2.):



Slika 2. Genetski operatori

Ovi operatori odgovaraju idealiziranim verzijama genetski operatora koji se mogu pronaći u prirodi. Dva najčešća operatora su ukrštanje i mutacija. Operator ukrštanja daje dva nova potomka od roditeljskih stringova kopirajući odabrane bitove roditelja. Bit na poziciji **i** u svakom potomku kopirani je bit na poziciji i nekog od dvoje roditelja.

Izbor koji roditelj doprinosi bit na poziciji **i** određuje sa drugim stringom koji se naziva maska ukrštanja. Za ilustraciju razmotrimo operator ukrštanja sa jednom tačkom. Na vrhu slike 2 razmotrimo najviši od dva potomka u ovom slučaju. Ovaj potomak uzima prvih pet bita od prvog roditelja, a ostalih šest bitova od drugog, zbog toga što je maska ukrštanja 11111000000 koja određuje ovaj izbor za svaku poziciju bita. Drugi potomak koristi istu masku ukrštanja,ali mjenja pravilo za oba roditelja. Zbog toga sadrži bite koje ne koristi prvi potomak. U ukrštanju sa jednom tačkom, maska ukrštanja uvjek je konstruisana tako da poćinje sa stringom koji sadrži kontinualno n i iza kojeg slijedi potreban broj 0 tako da se kompletira string.

Ovo rezultira u potomke u kojima je prvi n bita doprinjeti od jednog roditelja a ostatak bita od drugog. Svaki puta kada se maska sa jednom tačkom primjenjuje tačka ukkrštanja n se izabire slučajno.

U ukrštanju sa dvije tačke, potomak se kreira zamjenjujući središnji dio jednog roditelja sa sredinom drugog. Maska ukrštanja je string sa n_o nula na početku, slijedi kontinualno n_1 jedinica i ostatak je popunjen potrebnim brojem nula. Uvijek kada se primjenjuje ovaj operator n_o i n_1 se odabiru slučajno. U primjeru pokazanom na slici 2 potomci su kreirani koristeći masku $n_0=2$ i $n_1=5$. Ponovno su dva potomka kreirana zamjenjujući pravila za oba roditelja.

Uniformno ukrštanje kombinira bitove uzete uniformno od dva roditelja, kao što je prikazano na slici 2. U ovom slučaju maska ukrštanja je generisana kao slučajni bit string sa svakim izabranim bitom slučajno i neovisno o drugim. Za razliku od operatora koji proizvede potomke kombinujući dijelove roditelja, drugi tip operatora proizvodi potomke od jednog roditelja. Operator mutiranja proizvodi male slučajne promjene bit stringova birajući jedan bit slučajno i mijenja njegovu vrijednost.

Mutacija se obično izvodi poslije ukrštanja. Funkcija pogodnosti definiše kriterijum rangiranja potencijalnih hipoteza i za vjerovatnoću odabiranja za uključivanje u sljedeću generaciju populacije. Ako je zadatak naučiti klasifikacijsko pravilo, onda funkcija pogodnosti tipično ima komponentu da vrednjuje tačnost klasifikacije pravila nad skupom datih primjera učenja. Česti drugi kriteriji takođe mogu biti uključeni,kao što je kompleksnost i poštenje pravila. Uopštenije, kada se bit string hipoteza interpretira kao kompleksna procedura (npr. kada bit string prestavlja kolekciju ako-onda pravila koja če biti povezana zajedno za upravljanje robotom), funkcija pogodnosti može mjeriti sveukupno ponašanje rezultirajuče procedure umjesto ponašanja individualnih pravila.

U našem prototipskom genetskom algoritmu, vjerovatnoća da će hipoteza biti odabrana je data sa odnosom njene pogodnosti sa pogodnostima ostalih članica trenutne populacije kao što je viđeno u jednačini (1.2.). Ovaj metod se ćesto naziva proporcionalno odabiranje pogodnosti , ili kružna rulet sekcija. Postoje i drugačiji metodi biranja hipoteza. Npr., takmičarsko odabiranje, prvo se odaberu dvije hipoteze slučajno uz trenutne populacije. Sa nekom predefinisanom vjerovatnoćom p onda se odabire pogodniji od ove dvije hipoteze i hipoteza sa vjerovatnoćom (1-p) se odabire.

Takmičarski odabir često vodi više različitosti u populaciji nego odabir funkcijom pogodnosti. U drugom metodu nazvanom odabir rangiranjem, hipoteze u trenutnoj populaciji prvo se sortiraju po pogodnosti. Vjerovatnoća da će hipoteza biti odabrana je proporcionalna rangu u ovoj sortiranoj listi,umjesto njene pogodnosti.

Kako je predhodno ilustrirano, genetski algoriram koristi slučajni usmjereni metod pretraživanja za nalaženje maksimalno pogodne hipoteze. Ovo pretraživanje je potpuno različito od drugih metoda učenja, i ono je više iznenadno, zamjenjujući roditeljsku hipotezu sa potomcima koji mogu biti značajno drugačiji.

Jedna praktična poteškoča u nekim primjenama genetskog algoritma je problem nagomilavanja. Nagomilavanje je fenomen u kojem se neke individue koje više pogoduju nego ostale u populaciji brzo reprodukuju, tako da kopije ovih individua i veoma sličnih induvidua zauzimaju veoma veliki dio populacije. Negativna posljedica nagomilavanja je reduciranje raznovrsnosti populacije, zbog čega se usporava budući progres u Genetskim algoritmima. Različite su strategije pronađene za reduciranje nagomilavanja. Jedan je pristup mjenjanje funkcije odabira, koristeći kriterij kao što je takmičarsko odabiranje ili odabir rangiranjem umjesto proporcionalnog kružnog rulet odabiranja. Odgovarajuća strategija je **“**djeljenje pogodnosti”, u kojoj mjerenje pogodnosti individua reducira prisutnost drugih, sličnih individua u populaciji. Treći pristup je restrikcija vrsta individua dozvoljenih za rekonbiniranje u formiranju potomaka. Npr., omogućavanjem samo najsličnijim individuama rekonbinovanje, možemo ohrabriti formiranje klastera sličnih individua, ili više **“**podprostora” unutar populacije. Sličan pristup je prostorna distribucija individua i omogučavanje samo bliskim individuama rekombinovanje. Mnoge od ovih tehnika su inspirisane analogijom sa biološkom evolucijom.

Interasantno je pitati da li neko može matematički okarakterisati evoluciju tokom vremena populacije unutar genetskih algoritama.

”Šema teorem“ (Holland; 1975) daje jadnu takvu karakterizaciju. Bazirano je na konceptu šema ili uzoraka koji opisuju skup stringova. Da bi bili precizniji, šema je bilo koji string sastavljen od 0, 1, i *. Svaka šema predstavlja skup bit stringova koji sadrže indikatore 0 i 1 sa svakom ***** interpretiranom kao “nebitno”. Npr., šema 0*10 predstavlja skup bit stringova koji uključuju egzaktno 0010 i 0110.

Pojedinačni bit string može se posmatrati predstavljen različitim odgovarajućim šemama. Npr., bit string 0010 može se zamisliti kao predstavnik 2^4 odvojenih šema uključujući 00**, 0*10, **** itd. Na sličan način, populacija bit stringova može se posmatrati u zavisnosti skupa šema koje ga predstavljaju i broj individua svakoj od ovih šema.

Šema teorem karakteriše evoluciju populacije unutar genetskog algoritma u zavisnosti broja instanci predstavljanja svake šeme. Neka m (s,t) označava broj instanci šeme **s** u populaciji u trenutku t (tj. za vrijeme t-te generacije). Šema teorem opisuje vrijednost očekivanja od m(s, t+1) u zavisnosti od m(s,t) i drugih osobina šeme, populacije i algoritamski parametara genetskog algoritma.

Evolucija populacije u genetskom algoritmu zavisi od koraka selekcije, koraka rekombinovanja i koraka mutacije. Počnimo sa posmatranjem samo efekta koraka selekcije. Neka f(h) označava funkciju pogodnosti pojedinog bit stringa h, a f(t) označava srednju vrijednost pogodnosti svih individua u populaciji u trenutku t. Neka hЄs∩pt pokazuje da individua h je ujedno prestavljena šemom s i članom populacije u trenutku t.

Na kraju, neka u (s,t) označava srednju vrijednost pogodnosti instanci šeme s populacije u trenutku t. Nas zanima računanje očekivanja m(s,t +1), koja označavamo sa E[m (s,t+1)], koristeći distribuciju vjerovatnoće za selekciju datu sa jednačinom (1.2.), koja se može prestaviti koristeći ovu terminologiju kako slijedi:

$$P_r(h) = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} = \frac{f(h)}{n\bar{f}(t)}$$

Sada ako odaberemo jednog člana nove populacije u skladu sa distribucijom vjerovatnoće, onda je vjerovatnoća koju ćemo odabrati da predstavlja šemu s:

$$P_r(h \in s) = \sum_{h \in snpt} \frac{f(h)}{n\bar{f}(t)} = \frac{\hat{u}(s,t)}{n\bar{f}(t)} m(s,t) \qquad (1.3)$$

Drugi korak slijedi iz činjenice date definicijom,

$$\hat{u}(s,t) = \frac{\sum_{h \in snpt} f(h)}{m(m,t)}$$

Jednačina (1.3.) daje vjerovatnoću da će jedna hipoteza odbrana za genetski algoritam biti instanca šeme **s**. Zbog toga, očekivanje instanci iz s koja je rezultat **n** nezavisni koraka koji kreiraju cijelu novu generaciju je tačno **n** puta ova vjerovatnoća.

$$E\left[m(s,t+1)\right] = \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t) \qquad (1.4)$$

Jednačina (1.4.) izjavljuje da je očekivanje instanci šeme **s** kod generacije t+1 proporcionalno srednjoj pogodnosti u(s,t) instanci ove šeme u trenutku t, i obrnuto proporcionalno srednjoj pogodnosti f (t) svih članova populacije u trenutku t.

Prema tome možemo očekivati šeme sa gore spomenutom srednjom vrijednosti pogodnosti da će biti predstavljene sa povećanom učestalošću na uspješnim generacijama. Ako posmatramo da se genetski algoritam, kao izvođenje virtualnog paralelnog pretraživanja u prostoru mogućih šema, u istom trenutku izvodi eksplicitno paralelnom pretraživanju prostora individua, onda jednačina (1.3.) pokazuje da će više pogodni šema rasti u uticaju tokom vremena.

Dok gornja analiza razmatra samo korak selekcije genetskog algoritma, moraju koraci ukrštanja i mutacije biti razmotreni. Šema teorem razmatra samo mogući negativni uticaj ovih genetskih operatora (npr., slučajna mutacija može smanjiti predstavljanje **s,** nezavisno od u(s,t) i razmatrati samo slučaj ukrštanja sa jednom tačkom. Potpuna šema teorem prema tome daje donji propusni opseg očekivane frekvencije šeme **s,** kako slijedi:

$$E\left[m(s,t+1)\right] = \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t)\left(1 - p_c \frac{d(s)}{l-1}\right)(1-p_m)^{O(s)} \qquad (1.5)$$

Ovdje je p_c očekivanje da će operator ukrštanja sa jednom tačkom bitit primjenjen na odgovarajući bit string i p_m je očekivanje da će odgovarajući bit odgovarajuće individue biti mutirane sa operatorom mutiranja.

O(s) je broj definisanih bita u šemi s, gdje su definisani biti, ali ***** nije

d(s) je udaljenost između krajnjeg lijevog i krajnjeg desnog definisanog bita u s

l je dužina pojedinog stringa u populaciji.

Termin krajnji lijevi u jednačini (1.5.) identičan je terminu jednačine (1.4.) i opisuje efekat koraka odabiranja. Termin srednji opisuje efekat ukrštanja u jednoj tački – pojedinačno, to opisuje vjerovatnoću da će odgovarajuće individualno predstavljanje s još uvjek predstavljati **s** slijedeći primjenu operatora mutacije. Termin krajnji desni opisuje vjerovatnoću da će određeno individualno predstavljanje šeme s još uvjek predstavljati s slijedeći primjenu operatora mutacije. Primjećujemo da se efekat operatora ukrštanja u jednoj tački mutacije povećava s brojem definisanih bita o(s) u šemi sa distancom d(s) između definisanih bita. Prema tome šema teorem se može grubo interpletirati kao izjava da će više pogodnih šema težiti rastu u uticaju, posebno šeme koje sadrže male brojeve definisanih bita (tj., koje sadrže veliki broj *) i posebno kada su ove definicije bita bliske jedna drugoj unutar bit stringa.

Šema teorem je možda najraširenija citirana karakterizacija evolucije unutar gentskog algoritma. Jedini put u kojem nije kompletna je da pada u razmatranju pozitivnih efekata ukrštanja i mutacije.

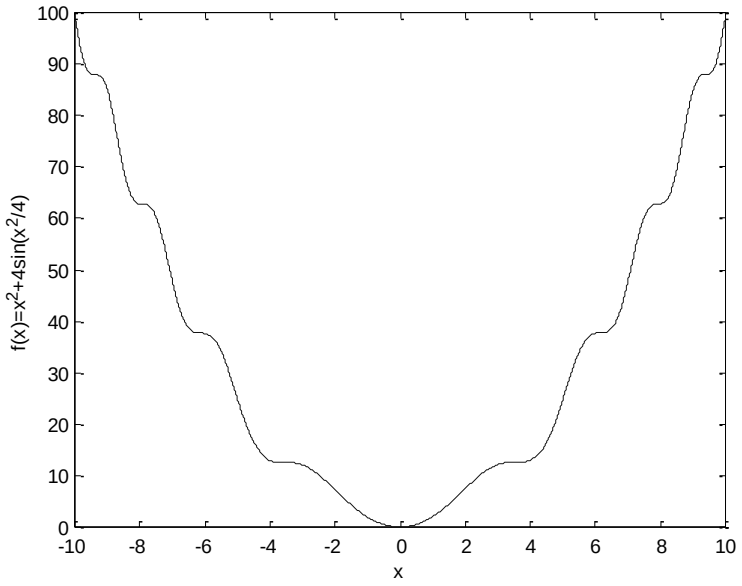
2.1. Produženi kompaktni genetski algoritam (ECGA)

Ovaj primjer nam govori na koji način instalirati, kombinovati i pokrenuti produženi, kompaktni genetički algoritam (ECGA), opisan u Harikovim papirima (Harik, 1999). Objašnjeno je kako modificirati ciljne funkcije koje se pojavljuju pri distribuciji koda. Izvor je pisan u C++, ali dovoljno je poznavanje programskog jezika C za modificiranje ciljne funkcije, tako da možete koristiti ECGA na vašu vlastitu odgovornost.

2.1.1. Kako pokrenuti kod

Za izvršenje funkcije ECGA, potrebna su 2 argumenta : input file i output file. ECGA čita svoje parametre preko input file i čuva rezultate u output file. Uzorak input file se čuva kao primjer sa raspodjelom koda. Ime datoteke je inputfile i njen sadržaj (zajedno sa brojem redova) je prikazan ispod.

Primjer 1D funkcije:



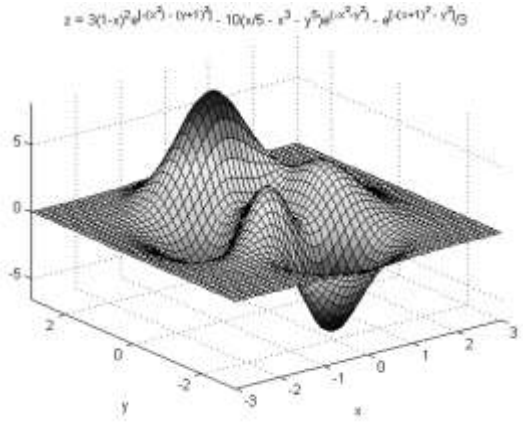
Dijagram 1. Primjer 1D funkcije

Ulazni podaci:

```
1  #
2   # funkcija
     f(x)=x^2+4*sin(x^2/4);
3  #
4  # maksimum: z=-f(x)
   #
5  POCETAK
6  duzina_hromozoma         16
7   inicijalno_sjeme

0.254534
8           velicina_populacije
1000
9  vjerovatnoca_ukrstanja   1
10 tournament_velicina      16
11 ucenje_MPM               da
12 stop_kriterij

dostignuta_vrijednost
13  stop_kriterij_argument   1e-
10
14 #
15 # parametri za stampanje
16 # rezultata
17 stampaj_populaciju       ne
18 stampaj_hromozom         da
19 stampaj_podobnost        da
20 stampaj_MPM              ne
21 KRAJ
```

Rezultat:

```
Generacija       : 0
Max  podobnost            :  -
0.0002217081
Srd. podobnost    : -32.92348
Najbolji       hromozom       :
0111111111011101
 f= -0.0002217081
------------------------------
Generacija       : 1
Max  podobnost            :  -
4.191079e-007
Srd. podobnost    : -0.8858986
Najbolji       hromozom       :
0111111111111110
 f= -4.191079e-007
------------------------------
→
Generacija       : 2
Max podobnost     : -4.656755e-008
Srd. podobnost    : -1.343858
Najbolji         hromozom         :
0111111111111111
 f= -4.656755e-008
------------------------------

Generacija       : 3
Max podobnost     : -4.656755e-008
Srd. podobnost    : -5.337492
Najbolji         hromozom         :
0111111111111111
 f= -4.656755e-008
------------------------------
Generacija       : 4
Max podobnost     : -4.656755e-008
Srd. podobnost    : -1.832825
Najbolji         hromozom         :
0111111111111111
 f= -4.656755e-008
------------------------------
Generacija       : 5
Max podobnost     : -4.656755e-008
Srd. podobnost    : -9.239002e-008
Najbolji         hromozom         :
0111111111111111
 f= -4.656755e-008
------------------------------
Generacija       : 6
Max podobnost     : -4.656755e-008
Srd. podobnost    : -4.656755e-008
Najbolji         hromozom         :
0111111111111111
 f= -4.656755e-008
------------------------------

x=-1,525902189669642175936522468
     90e-4
```



Dijagram 2. Primjer 2D funkcije:

Ulazna datoteka:

```
1  #
2  # maksimum funkcije: (peaks)
   #
3   # z = 3*(1-x).^2.*exp(-
          (x.^2) - (y+1).^2) ...
   #      - 10*(x/5 - x.^3 -
          y.^5).*exp(-x.^2-y.^2)
   #      - 1/3*exp(-(x+1).^2 -
          y.^2);
4  #
5  POCETAK
6  duzina_hromozoma         40
7   inicijalno_sjeme
                        0.254534
8  velicina_populacije      1000
9  vjerovatnoca_ukrstanja   1
10 tournament_velicina      16
11 ucenje_MPM               da
12 stop_kriterij
           dostignuta_vrijednost
13 stop_kriterij_argument   0
14 #
15 # parametri za stampanje
16 # rezultata
17 stampaj_populaciju       ne
18 stampaj_hromozom         da
19 stampaj_podobnost        da
20 stampaj_MPM              ne
21 KRAJ
```

Rezultat:

```
Generacija       : 0
Max podobnost     : 6.31652
Srd. podobnost    : 0.02062806
Najbolji          hromozom          :
1000010110000101100110010010000000010
100
   f= 6.31652
-----------------------------------
Generacija       : 1
Max podobnost     : 7.910785
Srd. podobnost    : 8.11649
Najbolji          hromozom          :
0111111111011101010010010100001110100
100   f= 8.11649
-----------------------------------
Generacija       : 13
Max podobnost     : 8.11649
Srd. podobnost    : 8.11649
Najbolji          hromozom          :
0111111111011101001110010100001110100
111   f= 8.11649
-----------------------------------
Generacija       : 14
Max podobnost     : 8.11649
Srd. podobnost    : 8.11649
Najbolji          hromozom          :
0111111111011101001110010100001110100
111   f= 8.11649
-----------------------------------
Generacija       : 15
Max podobnost     : 8.11649
Srd. podobnost    : 8.11649
Najbolji          hromozom          :
0111111111011101001110010100001110100
111   f= 8.11649
-----------------------------------
Generacija       : 16
Max podobnost     : 8.11649
Srd. podobnost    : 8.11649
Najbolji          hromozom          :
0111111111011101001110010100001110100
111   f= 8.11649

x=-1,061440526428724697804
     16279236e-2

y= 1,580344753594163507617
     4808668908
```

ECGA preskače svaki red sve dok ne dođe do riječi POČETAK (5-ti red u gornjim primjerima funkcije 1D i funkcije 2D). Tada počinje čitati parametre u prethodno definisanom redu. Program ne čini ni jednu suvišnu radnju na input datoteci. To znači da poslije riječi POČETAK, NE SMIJETE MIJENJATI REDOSLIJED REDOVA U INPUT DATOTECI, jer bi u suprotnom ECGA postao totalno konfuzan. Input datoteka je jako razumljiva.

U narednom tesktu slijedi objašnjenje svakog reda:

RED 6 govori da je problematična dužina 16, ili 40 u drugom primjeru.

RED 7 govori da 0.254534 predstavlja sjeme za inicijaciju brojnog generatora. To mora biti broj izmedu 0 i 1.

RED 8 govori da je veličina populacije 1000

RED 9 govori da je vjerovatnoća preokreta 1, što znači da se cijela populacija obnavlja poslije svakog generacijskog kruga.

RED 10 govori da je veličina prvenstva 16. Operator sprovodi samo jednu selekciju: prvenstvenu selekciju bez mogućnosti zamjene.

RED 11 govori da ECGA uči svaku generaciju MPM. Možete postaviti ovu opciju ali i ne morate (on ili off). Ako postavite na on, dobivate normalni ECGA. Ako postavite na off, dobivate orginalni kompaktni GA.

RED 12 govori da se ECGA zaustavlja kada je populacija potpuno konvergirala, odnosno, kada se populacija sastoji od N kopija iste individue. Osim allele_konvergentne opcije, također možete odabrati opciju max_ generacija.

RED 13 ovdje možete specificirati maksimalni broj generacija u slučaju da ste odabrali opciju max_generacija u prethodnom redu. Ako ste odabrali opciju allele_konvergenciju, ovaj parametar je irelevantan.

RED 17 govori da ne želimo poslati populaciju output datoteci na kraju svake generacije. Ovu opciju možete postaviti na on ili off. Ako odaberete on, trebate biti oprezni da ne biste otvorili veliki problem, jer bi u suprotnom izvještajna datoteka postala ogromna.

RED 18, 19, 20 su on i off zastavice korištene za svrhu izvještavanja.

Sada ste spremni da krenete naprijed i pokrenete ECGA. Unesite ecga inputfile outputfile. Statistika populacije se pojavljuje na ekranu na kraju svake generacije. Ista informacija se također šalje output datoteci. U dodatku, output datoteka također pokazuje različite MPM strukture koje ECGA pronalazi za vrijeme svog MPM pretraživanja.

2.1.2. Kako priključiti vašu vlastitu ciljnu funkciju

Šifra za ciljnu funkciju je u funkciji objective_func( ) u datoteci objfunc.cpp. Ovo je jedina funkcija koju morate prepisati ako želite isprobati vaš problem. Početak funkcije je sljedeći:

Double objective_func (char # chrom, int lchrom);

Kao argument uzima karaktere niza od 1 i 0, kao i dužinu niza. Funkcija vraća stvarne brojeve vrijednost niza ciljne funkcije. U sadašnjoj implementaciji ECGA da je problem binarno kodiran.

2.1.3. Nešto o c++ kodu

Sprovođenje ecga ne koristi prednosti C ++ jezika kao što je nasljeđe i forma. To znači da ne morate biti stručnjak za C ++ da bi modificirali kod. Možete modificirati kod i priključiti vašu ciljnu funkciju koristeći samo programski jezik. Stoga, jasan opis izvora datoteka je dat u nastavku. Svaka .cpp datoteka ima odgovarajuću .hpp datoteku. *hpp je glavna datoteka i sadrži definicije različitih razreda. *cpp datoteka sadrži aktualnu implementaciju.

Gen.cpp sadrži sprovođenje gena razreda. Gen ima locus I allele.

Hromosom.cpp sadrži implemetaciju hromosoma razreda.

Populacija.cpp sadrži implementaciju poppulacije razreda. Populacija je hromosoma. Odabir operatora, statistika populacije, i stop kriteriji se sprovode ovdje.

Objfunc.cpp sadrži šifru za ciljnu funkciju. Ako želite isprobati ECGA na vašem vlastitom problemu, trebate modificirati funkciju objective_func ( ) sadržanu u ovoj datoteci.

Korisnost.cpp sadrži korisne funkcije i procedure.

Intlist.cpp sprovodi listu integera.

Mpm.cpp sadrži operacije koje ne mogu biti izvršene u MPM.

Cache.cpp sprovodi cache koji se koristi za brže pretraživanje MPM-a.

Slučajni izbor.cpp je brojni generator.

Ecga.cpp sadrži glavni loop ECGA.

Glavni.cpp sadrži glavnu ( ) funkciju i inicijalne procedure.

3. ZAKLJUČAK

Popularnost Genetskog algoritma motivirana je brojnim faktorima koji uključuju:

- Evolucija je poznato uspješna,robusan metod za adaptaciju unutar bioloških sistema.
- Genetski algoritmi mogu pretraživati prostor koji sadrži kompleksno uzajamne dijelove gdje uticaj svakog dijela nad pogodnost hipoteze može biti težak za modelovanje.

Genetski algoritmi su jednostavni za paralelizaciju i mogu iskoristiti prednost u opadanju cijene računarske snage.

LITERATURA

1. Baluja, S., “Genetic Algorithms and Exsplicit Search Statistics”
2. Harik, G. R., 1999. “Linkage Learning via Probabilistic Modeling in the ECGA (IlliGAL Report No.99010), University of Illinois at Urbana-Champaing.
3. Lin, C.H., Ling Wu, J., 1999.“Automatic Facial Feature Exstraction by Genetic Algorithms”,Transactions on Image Processing, Vol.8.No.6.
4. Lobo, F.G., Harik, G. R., “Extended Compact Genetic Algorithm in C + +”, University of Illinois at Urbana-Champaing,Genetic Algorithms Laboratory.