Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Završni rad br. 317

Primjena genetskog programiranja u postupku simboličkog integriranja

Ivan Kokan

Zagreb, lipanj 2008.

Primjena genetskog programiranja u postupku simboličkog integriranja

Sažetak

Genetsko programiranje je član obitelji evolucijskih algoritama temeljenih na biološkim principima preslikanima u svijet računarstva. Kao rezultat genetskog procesa javlja se računalni program, što genetsko programiranje uvelike razlikuje od ostalih srodnih područja. Početak razvoja bilježi se u pedesetim godinama 20. stoljeća, ali procvat doživljava od devedesetih, prvenstveno zbog računalne revolucije.

Iznesene su osnovne teorijske postavke genetskog programiranja, kao i temelji numeričke analize – interpolacijski polinom i trapezne formule. Razvijena je i programska aplikacija koja rješava aproksimacijski problem te problem simboličke integracije, a omogućava uvid u opisane genetske procese kroz analizu njihovih rezultata.

Ključne riječi:

genetsko programiranje, algoritam, evolucija, numerička integracija, simbolička integracija, aproksimacija

Application of Genetic Programming in Symbolic Integration

Abstract

Genetic programming is member of the Family of evolutionary algorithms, based on biologic principles mapped into the world of modern computing. The result of genetic process is computer program, which distinguishes genetic programming from similar areas of study. Genetic programming traces its roots back to the 50's, and has thrived since the 90's, when the computer revolution has taken place.

Here are introduced basic theoretic statements of genetic programming and bases of numerical analysis – interpolating polynomial and trapezoidal formulas. Also, a Windows application, which solves approximation problem and problem of symbolic integration, has been developed. It gives practical view on described genetic processes through the analysis of their results.

Keywords:

genetic programming, algorithm, evolution, numerical integration, symbolic integration, approximation

Sadržaj

1 Uvod **1**

2 Uvod u genetsko programiranje **2**

2.1 Formalna predodžba računalnog programa 2

2.2 GP kao poseban slučaj evolucijskog algoritma 3

2.3 Jedinke i populacija 4

2.4 Funkcija dobrote i dobrota jedinke 4

2.5 Genetski operatori 4

2.5.1 Reprodukcija 4

2.5.2 Križanje 5

2.5.3 Mutacija 5

2.6 Selekcija 6

2.7 Zaustavljanje generativnog procesa i utvrđivanje rješenja 6

2.8 Blok-dijagram genetskog programiranja 7

3 Primjer problema **8**

3.1 Reprezentacija jedinki 8

3.2 Evaluacija dobrote 8

3.3 Analiza rezultata 9

4 Od numeričke analize do genetskog programiranja **10**

4.1 Interpolacijski polinom 10

4.2 Lagrangeov oblik interpolacijskog polinoma 11

4.3 Aproksimacijski problem u genetskom programiranju 11

4.4 Osnovna ideja numeričkog integriranja 12

4.5 Newton-Cotesova formula 12

4.6 Trapezne formule i simbolička integracija 12

5 Programska rješenja **14**

5.1 Problem umjetnog mrava 14

5.2 Aproksimacijski problem 14

5.3 Problem simboličke integracije 14

5.3.1 Korisničko sučelje 14

5.3.2 Osnovni parametri 15

5.3.3 Generiranje jedinki 16
5.3.4 Funkcija dobrote i selekcija 16
5.3.5 Uzorkovanje i odabiranje problema 17
5.3.6 Praćenje rada i prikaz rezultata 17
5.3.7 Ispitivanje i analiza dobivenih rezultata 18

6 Zaključak **21**

1 Uvod

Znanstveni problemi današnjice, posebno u računarstvu, poprimaju sve veće razmjere. Paralelno s time, smanjuje se vjerojatnost da su rješenja istih jednostavna, a kamoli egzaktna. U skladu s tom nepobitnom činjenicom, a na temeljima stohastike, svijet modernog računarstva sve se više okreće empirizmu, čak i u slučaju eksplicitno riješenih problema (najčešće zbog složenosti samih rješenja).

Temelj evolucijskih algoritama (eng. *evolutionary algorithms*), kojima pripada i genetsko programiranje (eng. *genetic programming*), su pojave iz biološke evolucije – reprodukcija, mutacija, rekombinacija, selekcija itd. [10]. Iako na prvi pogled ne postoji jasno vidljiva poveznica između spomenutih domena, nakon kraćeg razmatranja ipak se dolazi do zaključka kako su principi **darvinizma** – *bolji opstaje, lošiji izumire* – idealni za pronalaženje dovoljno dobra rješenja zadanog problema (ako ne i u potpunosti točna).

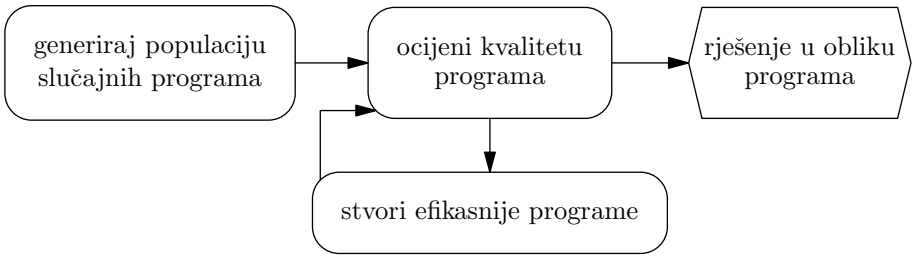
Iako se teorijski razvijao paralelno s ostalim evolucijskim metodama još od pedesetih godina 20. stoljeća, koncept genetskog programiranja svoju je pravu vrijednost dosegao tek u devedesetima, kada ga je utvrdio (i u rješavanju raznih problema primijenio) John R. Koza, doajen računarske znanosti sa Sveučilišta Stanford. Otada se uočava još veća progresija u razvoju, kako u istraživačkim timovima diljem svijeta, tako i kod samoga Koze koji neumorno nastavlja sa svojim životnim pozivom – objavljivanjem znanstvenih radova i prolongiranjem genetskog programiranja kao jedne od vodećih sila u razvoju **umjetne inteligencije** (eng. *artificial intelligence*).

Ovaj se uradak dotiče konkretne primjene genetskog programiranja u rješavanju aproksimacijskog problema s fokusom na **simboličkoj integraciji** [5]. Svojevrsni je nastavak seminara s prijašnjih kolegija Seminar [4] i Projekt [2], čiji su pročišćeni dijelovi ovdje i upotrijebljeni, a u kojima su redom izneseni temelji numeričke integracije i genetskog programiranja. U okviru Završnog rada razvijena je i programska aplikacija *IntegratorGP*, čiji su rezultati prikazani i analizirani.

2 Uvod u genetsko programiranje

Genetsko programiranje (GP) je automatiziran optimizacijski postupak razvoja računalnih programa, čija je namjena rješavanje složenih problema iz područja računarstva, ali i problema na koje se svakodnevno nailazi. Koncept je zasnovan na općim idejama proizašlima iz teorije genetskih algoritama (eng. *genetic algorithms*), kao i drugih evolucijskih metoda. Najjednostavnije rečeno, konačni cilj genetskog programiranja (kao produkt) je univerzalni **računalni program** koji nalazi rješenja problema kao ulaznih podataka. Naravno, ti se problemi moraju moći i opisati i prikazati na prikladan način.

Pojednostavljeni shematski prikaz genetskog programiranja dan je na Slici 2.1.

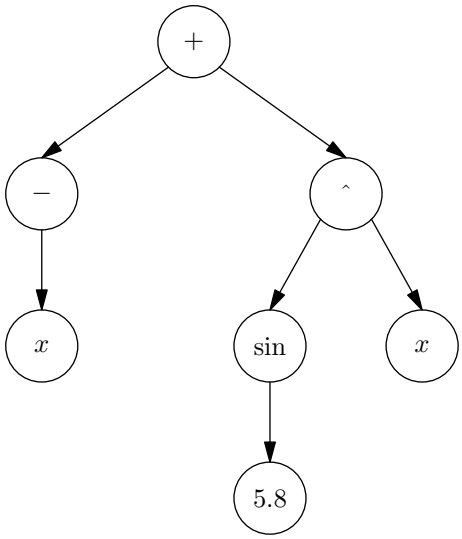


Slika 2.1: *Pojednostavljeni shematski prikaz genetskog programiranja.*

2.1 Formalna predodžba računalnog programa

Bez obzira na činjenicu da se genetskom programiranju može pristupiti na više različitih načina i iz više različitih perspektiva, kao npr. linearne – uglavnom se radi o univerzalnom pristupu računalnom programu kao formalnom stablu u kontekstu teorije grafova [11]. Naime, svaki računalni program može se prikazati kao stablo (eng. *tree*) ili šuma stabala (u širem smislu), gdje unutarnji čvorovi stabala (eng. *nodes*) imaju ulogu operatora (ili funkcije određenog broja varijabli), a listovi ulogu operanada. Pritom su skup operatora F (eng. *function set*) i skup operanada T (eng. *terminal set*) unaprijed definirani konačni skupovi. Na Slici 2.2 prikazano je stablo koje je ekvivalentno jednostavnu matematičkom izrazu, koji također može predstavljati računalni program.

U skladu s općenitom teorijom evolucijskih algoritama, može se reći da kod genetskog programiranja ulogu **kromosoma** u genetskim procesima imaju **nelinearni** objekti – grafovi, odnosno stabla. Upravo svojstva stabala kao strogih matematičkih objekata, poput jednostavna rekurzivnog obilaska, zapečatila su razvojni put genetskog programiranja u naglašenom smjeru.



Slika 2.2: *Stablasta predodžba izraza* $-x + \sin^x 5.8$.

2.2 GP kao poseban slučaj evolucijskog algoritma

Iterativni postupak na kojemu se temelji genetsko programiranje u sebi sadrži jasno vidljive konture općenitog pseudokoda evolucijskih algoritama (Slika 2.3), a to su:

1) generiranje inicijalne **populacije** rješenja (eng. *population*),
2) analiza svake **jedinke** (eng. *unit*) populacije i populacije općenito te
3) odabir **genetskog operatora** uz provedbu odgovarajućih akcija nad jedinkom ili jedinkama.

```
Inicijaliziraj populaciju
Evaluiraj inicijalnu populaciju
Radi
   1.  izvrši selekciju
   2.  primijeni operacije odgovarajuće genetskim operatorima
       i izgeneriraj nova rješenja
   3.  evaluiraj rješenja u populaciji
Dok nije zadovoljen kriterij konvergencije
```

Slika 2.3: *Pseudokod evolucijskih algoritama.*

2.3 Jedinke i populacija

Biološka definicija opisuje populaciju kao skup jedinki iste vrste koje egzistiraju u istom prostoru [13]. Razmnožavanjem unutar populacije (stvaranjem potomaka), ali i umiranjem, veličina populacije je uravnotežena (konstantna) kroz generativni proces u uvjetima prirodne ravnoteže.

U genetskom programiranju jedinku predstavlja računalni program, a prostor egzistencije jedinki iste populacije je jedna iteracija optimizacijskog algoritma. Prema tome, govori se o populaciji računalnih programa jedne iteracije algoritma.

2.4 Funkcija dobrote i dobrota jedinke

Poznato je da se među biološkom vrstom na određenom staništu neke jedinke više ističu, imaju dulji životni vijek, veće mogućnosti za stvaranje potomstva i sl. Kao i prije, povlači se **analogija** s genetskim programiranjem. Naime, u cilju rješavanja problema evolucijskim algoritmom, vrlo je bitno da bolja rješenja ostaju u daljnjem razmatranju, a ona lošija otpadaju. Upravo dobro definirana funkcija dobrote obavlja tu ulogu. Na temelju raznih parametara, ona određuje **dobrotu** (eng. *fitness*) pojedine jedinke, o kojoj kasnije uvelike ovisi sudbina iste.

Definiranje funkcije dobrote jedan je od ključnih problema genetskog programiranja. Potrebno je da bude što "bolja", a što "jednostavnija", jer je njeno evaluiranje prisutno u svakom koraku generativnog procesa. Zapravo se odabir i definiranje funkcije dobrote **prilagođava** samu problemu i njegovim karakteristikama, a zadovoljenost navedenih oprečnih zahtjeva (kvaliteta i jednostavnost) pokušava se uravnotežiti.

2.5 Genetski operatori

Evolucijski aspekt genetskog programiranja očituje se u načinu optimiranja promatranog programa, gdje su u ulogama genetskih operatora prisutne metode **reprodukcije** (eng. *reproduction*) i **križanja** (eng. *crossover*).

U nekim slučajevima pojavljuju se i operatori **mutacije** (eng. *mutation*), permutacije (eng. *permutation*), inverzije (eng. *inversion*) itd.

2.5.1 Reprodukcija

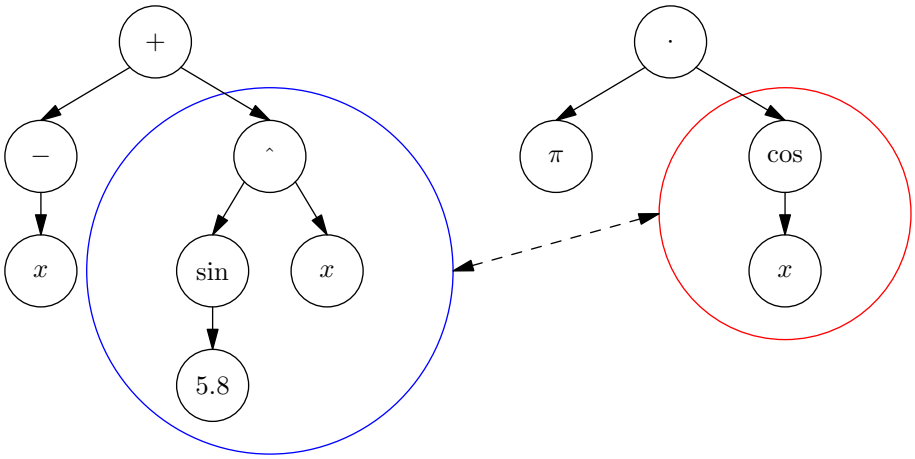
Reprodukcija je jednostavno kopiranje odabrane jedinke i umetanje iste u novu populaciju. Semantika ovog genetskog operatora je **preživljavanje** odabrane jedinke, a očekivana posljedica je povećanje prosječne dobrote populacije kroz iteracije algoritma. Važni parametri su:

- p_r – vjerojatnost odabira reprodukcije kao genetskog operatora,
- p_u – vjerojatnost odabira pojedine jedinke.

2.5.2 Križanje

Križanje je analogno biološkoj **spolnoj rekombinaciji** (Slika 2.4). Naime, u tom postupku dolazi do zamjene nekih podstabala dviju odabranih jedinki. Najučestaliji oblik je obavljanje opisane radnje na dva slučajno odabrana podstabla. Važni parametri su:

- p_c – vjerojatnost odabira križanja kao genetskog operatora,
- p_u – vjerojatnost odabira pojedine jedinke,
- p_n – vjerojatnost odabira pojedinih čvorova jedinki kao korijena podstabala.

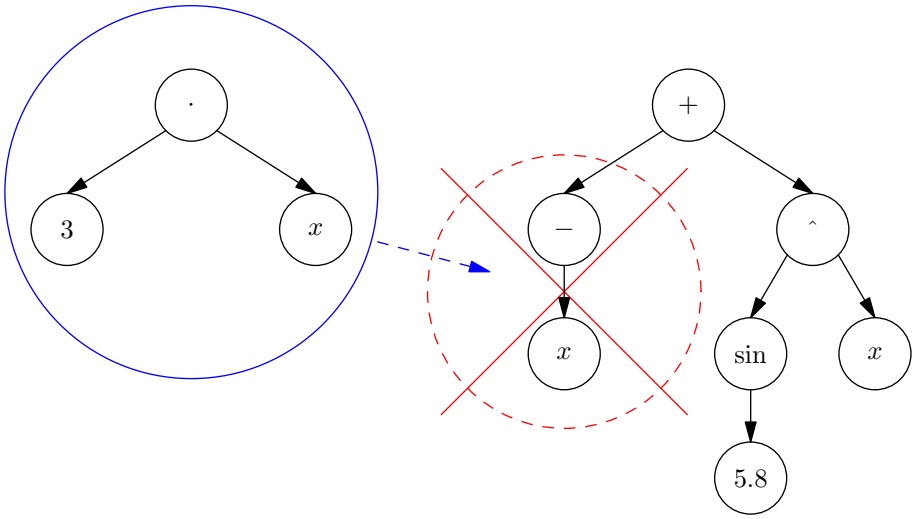


Slika 2.4: *Križanje.*

2.5.3 Mutacija

Analogno biološkoj mutaciji, dolazi do **poremećaja** "genskog materijala" najčešće slučajno odabrane ili drugim operatorima stvorene jedinke. Radi se o zamjeni nekog podstabla jedinke novim (slučajno generiranim) stablom (Slika 2.5). Važni parametri su:

- p_m – vjerojatnost odabira mutacije kao genetskog operatora,
- p_u – vjerojatnost odabira pojedine jedinke,
- p_n – vjerojatnost odabira pojedinog čvora jedinke kao korijena podstabla.



Slika 2.5: *Mutacija.*

2.6 Selekcija

Konačno, tijekom generiranja populacija odvija se **selekcija**, i to na temelju vjerojatnosti odabira genetskih operatora te dobrota jedinki. Uobičajeno je da selekcija izravno (proporcionalno) ovisi o dobroti (npr. jednostavna selekcija uz tzv. *roulette wheel* metodu odabira jedinki).

Vrlo su popularne i tzv. **turnirske** selekcije (eng. *tournament selections*), kod kojih se slučajno odabire k jedinki te se najgora, odnosno najgore jedinke zamjenjuju novim jedinkama nastalima primjenom genetskih operatora nad najboljima. Parametar k je unaprijed zadan u generativnom procesu (obično je $k = 3$ ili $k = 4$), pa se često govori o k-turnirskoj selekciji. Turnirske selekcije imaju i zanimljivo svojstvo **elitizma**, što znači da najbolja jedinka sigurno preživljava selekciju u promatranoj iteraciji.

Kao jedna od boljih selekcija, u literaturi [5] se navodi i sljedeća:

1) populacija se podijeli na dvije podpopulacije,
2) većina operacija (npr. 80%) određenog genetskog operatora se provodi nad jednom podpopulacijom.

2.7 Zaustavljanje generativnog procesa i utvrđivanje rješenja

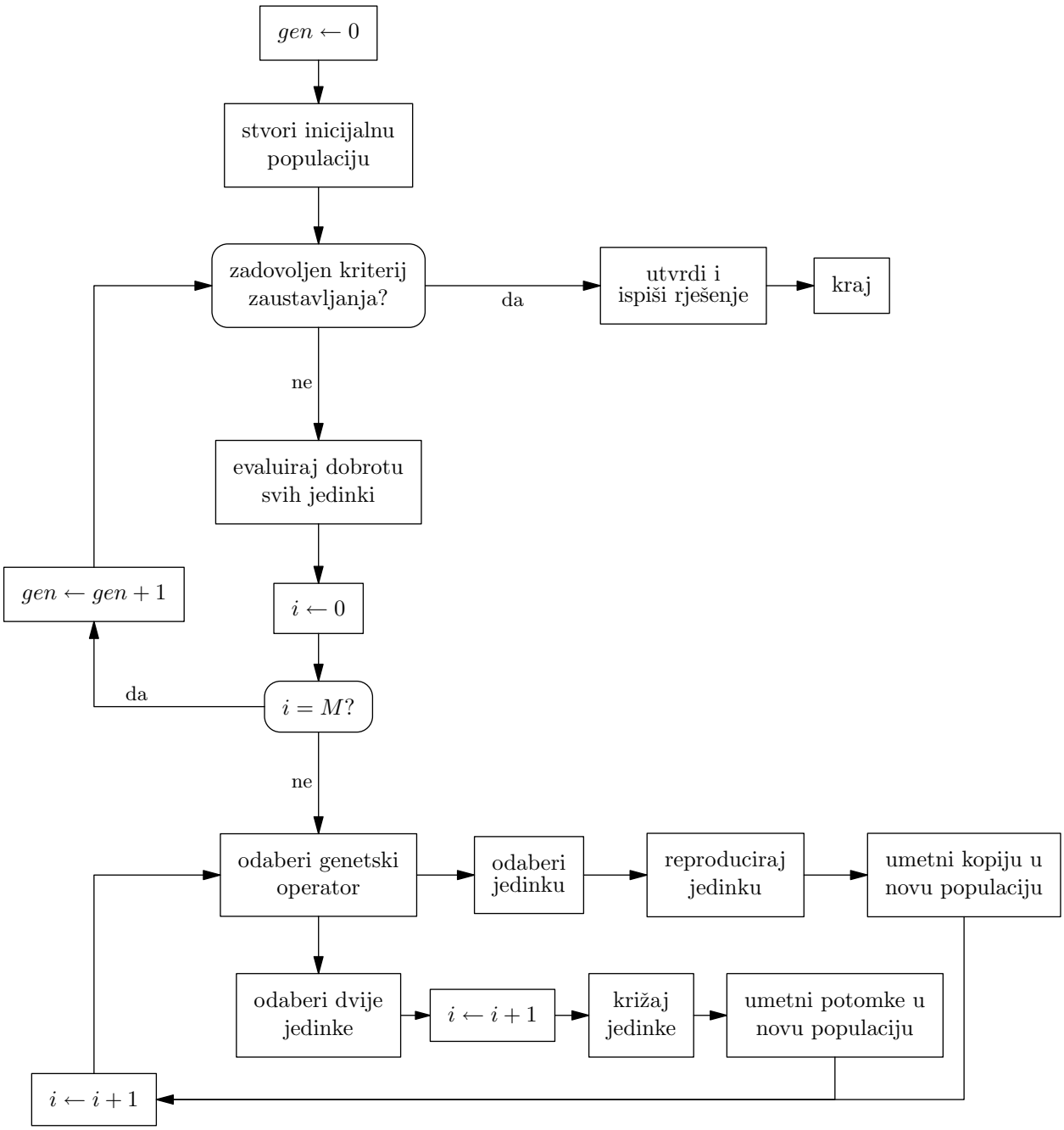
S obzirom na činjenicu da koncept genetskog programiranja s dobro definiranom funkcijom dobrote nad određenim problemom i adekvatnim skupovima F i T, relativno brzo dolazi do zadovoljavajućeg rješenja problema, u velikom broju slučajeva generativni proces se zaustavlja pri ispunjenju određenog logičkog predikata (u ovisnosti o dobroti). Dodatno se uzima i konstanta G, kao supremum broja iteracija, odnosno broja generacija populacije. Također, zadaje se i supremum M broja jedinki u populaciji, te supremum D **dubine** generiranih stabala.

Jednom kada se ispuni kriterij zaustavljanja, iz populacije rješenja odabire se najk-

valitetnije te se **semantički** opisuje, ako je ikako moguće. Poželjne su i daljnje analize istog, kao i ponavljanje cijelog postupka zbog nezadovoljstva dobivenim rješenjem. Bolje rješenje se uglavnom pokušava pronaći redefiniranjem funkcije dobrote, redefiniranjem skupova F i T, ili povećanjem supremuma broja iteracija.

2.8 Blok-dijagram genetskog programiranja

Genetsko programiranje konačno se može predočiti blok-dijagramom na Slici 2.6.



Slika 2.6: *Blok-dijagram genetskog programiranja.*

3 Primjer problema

Koncept genetskog programiranja može se prikazati na dobro istraženom primjeru pozicioniranja kolica s raketom u ishodište promatranog koordinatnog sustava [5] u što kraćem vremenu, koja u nekom trenutku t na beskonačno dugoj podlozi bez trenja imaju položaj $x(t)$, brzinu $v(t)$, a sve djelovanjem sile proizvoljne orijentacije iznosa F. Očito se radi o optimizacijskom problemu po varijabli vremena.

Formalno gledajući, problem se može opisati s dvije varijable stanja – $x(t)$ i $v(t)$. Ako se sa m označi masa kolica, iznos akceleracije $a(t)$ implicitno je dan sa $F = ma(t)$. Slijede jednadžbe stanja:

$$\begin{aligned} v(t+1) &= v(t) + \tau a(t), \\ x(t+1) &= x(t) + \tau v(t). \end{aligned} \tag{3.1}$$

Problem je u odlučivanju o **orijentaciji** sile stalnog iznosa F koja djeluje na kolica. Definira se pozicioniranje u ishodište u trenutku t_0 ako i samo ako vrijedi:

$$\lim_{t \to t_0} |x(t)| = \lim_{t \to t_0} |v(t)| = 0, \tag{3.2}$$

ne ulazeći dublje u svojstva limesa i prirodu funkcija $x(t)$ i $v(t)$.

Iz jednadžbi stanja analitički se jednostavno dolazi do optimalne strategije – ako je

$$-x(t) > \frac{mv(t)^2 \operatorname{sgn} v(t)}{2F}, \tag{3.3}$$

na kolica se djeluje pozitivno orijentiranom, a inače negativno orijentiranom silom F.

Ako se želi napisati računalni program koji traži rješenje opisanog problema, bez imalo razmišljanja nameće se rješenje ostvareno funkcijom koja vraća orijentaciju u kojoj je potrebno djelovati silom F za ulazne parametre $x(t)$ i $v(t)$. Genetsko programiranje se čini jako prikladnim rješenjem u ovakvoj situaciji. Slijedi opis takva rješenja.

3.1 Reprezentacija jedinki

Svaka jedinka, tj. računalni program koji predstavlja strategiju rješavanja problema, prikazuje se na standardni način – kao stablo kojemu su unutarnji čvorovi operatori, a vanjski čvorovi operandi.

3.2 Evaluacija dobrote

Vrijeme potrebno za pozicioniranje kolica očito ovisi o početnim uvjetima $x(0)$ i $v(0)$. Na pitanje koliko je neki program (strategija) koji rješava ovaj problem dobar, može se odgovoriti mjerenjem **prosječnog vremena** potrebnog za pozicioniranje kolica na nekom uzorku početnih stanja. Da bi ta generalizacija bila opravdana, uzorak mora biti reprezentativan, tj. mora predstavljati problem u cjelini. Jedan od načina na koji se to vjerojatno može postići je odabiranje relativno velikog uzorka.

Nadalje, potrebna je metoda koja evaluira dobrotu onih programa koji uspijevaju pozicionirati kolica u ishodište, kao i onih koji to ne uspijevaju. To se može ostvariti

računanjem stanja u diskretnim vremenskim koracima, pa ako se kolica u nekom koraku nalaze **dovoljno blizu** ciljnom stanju (ishodištu), smatra se da je program ispunio cilj te je njegova dobrota jednaka utrošenu vremenu. U protivnom, dobroti programa dodjeljuje se neka **kaznena vrijednost** (eng. *penalty value*).

U opisanom slučaju odabrano je 20 slučajnih točaka iz podravnine $[-0.75, 0.75]^2$, koje predstavljaju uređene parove $(x(0), v(0))$, a kaznena vrijednost u slučaju da program ne uspije pozicionirati kolica je 10 s. Koristi se standardni operator križanja među jedinkama. Za odabrani uzorak optimalno rješenje postiže prosječno vrijeme od 2.02 s.

3.3 Analiza rezultata

Već u trećoj generaciji jedinki došlo se do rješenja vrlo blizu optimalnom, a u 33. generaciji pronašlo se upravo 100% točno rješenje. Primjetno je da na početku **nije pretpostavljena** ni veličina, ni oblik, ni strukturna složenost rješenja, nego se ono samo pojavilo kao posljedica dobra odabira funkcije dobrote. Za razliku od ovog primjera, kao rezultat genetskog programiranja često se ne dobivaju 100% točna rješenja, već neka blizu optimalnih.

4 Od numeričke analize do genetskog programiranja

Postupci numeričkog integriranja kao dio numeričke analize – jedne od starijih matematičkih grana, danas su vrlo prisutni i popularni, kao i genetsko programiranje. Naime, matematičari su odavna tražili postupke i načine da svoje (u krajnju ruku točne) zaključke i formule primijene u interesnim područjima, ali su nerijetko bili dužni određene probleme rješavati i za druge znanstvene discipline. Kako često **kompleksna rješenja** takvih problema i nisu bila "korisna" u tom obliku, morali su se okrenuti aproksimativnom pristupu kao "nužnom zlu", ali i dalje baziranom na čvrstim dokazima i **proračunima pogreške**, čije su ocjene vrlo zadovoljavajuće.

Među brojnim metodama aproksimiranja i numeričkog integriranja, ovdje se navode najjednostavnije, ali dovoljno dobre metode – **interpolacijski polinom** i **trapezne formule**, koje su poseban slučaj **Newton-Cotesovih formula**.

4.1 Interpolacijski polinom

Promatra se funkcija $x \mapsto f(x)$ i različite točke njenog grafa Γ_f:

$$(x_0, f(x_0)),\ (x_1, f(x_1)),\ \ldots,\ (x_n, f(x_n)), \tag{4.1}$$

bez obzira jesu li te točke dobivene uzorkovanjem grafa funkcije (zadane formulom preslikavanja) ili eksperimentalno. Odabrani x_i nazivaju se **čvorovima**, odnosno baznim ili interpolacijskim točkama [3]. Cilj je na temelju tih točaka funkciju $f(x)$ aproksimirati **polinomom** za $x \neq x_i$, gdje je $i = 0,\ 1,\ \ldots,\ n$.

Pretpostavivši interpolacijski polinom s realnim koeficijentima

$$P(x) = a_0 + a_1 x + \ldots + a_n x^n, \tag{4.2}$$

gdje je $\deg P(x) \leqslant n$ i $P(x_i) = f(x_i)$, $i = 0,\ 1,\ \ldots,\ n$, problem se svodi na rješavanje sljedećeg sustava linearnih jednadžbi:

$$\begin{aligned} a_0 + a_1 x_0 + \ldots + a_n x_0^n &= f(x_0), \\ a_0 + a_1 x_1 + \ldots + a_n x_1^n &= f(x_1), \\ &\vdots \\ a_0 + a_1 x_n + \ldots + a_n x_n^n &= f(x_n). \end{aligned} \tag{4.3}$$

Sustav (4.3) ima jedinstveno rješenje po nepoznanicama a_i (čime je i polinom $P(x)$ jedinstven), jer je determinanta tog sustava **Vandermondeova determinanta** [7], za koju vrijedi:

$$\begin{vmatrix} 1 & x_0 & \ldots & x_0^n \\ 1 & x_1 & \ldots & x_1^n \\ & & \ddots & \\ 1 & x_n & \ldots & x_n^n \end{vmatrix} \triangleq \prod_{i>j} (x_i - x_j) \neq 0. \tag{4.4}$$

(Prijelaz Δ u (4.4) lako se dokazuje matematičkom indukcijom.)

Radi lakše daljnje notacije, uvodi se prirodna oznaka $y_i := f(x_i)$.

4.2 Lagrangeov oblik interpolacijskog polinoma

Interpolacijski polinom $P(x)$ se može prikazati u pogodnijem **Lagrangeovom obliku** interpolacijskog polinoma $L_n(x)$ [12]. Može se primijetiti da za svaki odabrani čvor x_i vrijedi

$$f(x_i) = \sum_{j=0}^{n} y_j \delta_{ji}, \tag{4.5}$$

gdje je δ_{ij} **Kroneckerov delta simbol**, za koji vrijedi $\delta_{ii} = 1$, a inače $\delta_{ij} = 0$.

Pretpostavi li se Lagrangeov oblik interpolacijskog polinoma na sličan način:

$$L_n(x) = \sum_{j=0}^{n} y_j p_j(x), \tag{4.6}$$

gdje je $p_j(x)$ polinom s realnim koeficijentima, zaključuje se da $p_j(x)$ za nultočke ima sve čvorove osim x_j. Slijedi:

$$p_j(x) = C_j \left(x - x_0\right)\left(x - x_1\right)\cdots\left(x - x_{j-1}\right)\left(x - x_{j+1}\right)\cdots\left(x - x_n\right). \tag{4.7}$$

Uz uvjet $p_j(x_j) = \delta_{jj} = 1$ dobiva se

$$C_j = \frac{1}{\left(x_j - x_0\right)\left(x_j - x_1\right)\cdots\left(x_j - x_{j-1}\right)\left(x_j - x_{j+1}\right)\cdots\left(x_j - x_n\right)}. \tag{4.8}$$

Konačan oblik Lagrangeova interpolacijskog polinoma:

$$L_n(x) = \sum_{i=0}^{n} y_i \, \frac{\prod_{j=0}^{n} \left(x - x_j\right)}{\left(x - x_i\right) \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(x_i - x_j\right)}. \tag{4.9}$$

Intuitivno se povećanjem broja čvorova smanjuje pogreška aproksimacije (dokaz se ovdje ne navodi). U daljnjem tekstu podrazumijeva se **ekvidistantnost** čvorova, jer je u tom slučaju prosječna pogreška najmanja.

4.3 Aproksimacijski problem u genetskom programiranju

Za razliku od egzaktnih matematičkih metoda, aproksimacijskom problemu može se pristupiti i iz domene genetskog programiranja. Naime, ukoliko se za jedinku odabere realna funkcija realne varijable i vizualizira se njen graf, za očekivati je da genetsko programiranje "ispravlja" grafove potpuno "promašenih" funkcija do grafova vrlo dobrih aproksimacija, ako ne i do grafa izvorne funkcije.

4.4 Osnovna ideja numeričkog integriranja

Ukoliko se promatrana funkcija $f(x)$ aproksimira nekom "jednostavnijom" funkcijom $g(x)$ (pogodnijom za integriranje), iz $f(x) \approx g(x)$ dobiva se

$$\int_a^b f(x)\,\mathrm{d}x \;\approx \int_a^b g(x)\,\mathrm{d}x + R(f), \tag{4.10}$$

gdje je $R(f)$ pogreška uzrokovana aproksimativnim pristupom u integraciji, te $a, b \in \mathbb{R}$.

4.5 Newton-Cotesova formula

Neka je $g(x)$ interpolacijski polinom u Lagrangeovom obliku koji aproksimira promatranu funkciju $f(x)$. Iz (4.10) izravno se dobiva

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \;\approx\; \sum_{i=0}^{n} y_i A_i, \tag{4.11}$$

gdje je

$$A_i = \int_{x_0}^{x_n} \frac{\prod_{j=0}^{n} (x - x_j)}{(x - x_i) \prod_{\substack{j=0 \\ j\neq i}}^{n} (x_i - x_j)}\,\mathrm{d}x. \tag{4.12}$$

Uz već pretpostavljenu ekvidistantnost čvorova x_i dobiva se **Newton-Cotesova formula** [6]:

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \;\approx\; (x_n - x_0) \sum_{i=0}^{n} y_i H_i, \tag{4.13}$$

gdje su $H_i = \dfrac{A_i}{x_n - x_0}$ **Cotesovi koeficijenti**, za koje vrijedi $\displaystyle\sum_{i=0}^{n} H_i = 1$ i $H_i = H_{n-i}$ (simetričnost). Dokaz ovih svojstava ovdje se ne navodi.

4.6 Trapezne formule i simbolička integracija

Za $n = 1$ Newton-Cotesova formula prelazi u **osnovnu trapeznu formulu**:

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x \;\approx\; \frac{x_1 - x_0}{2}\,(y_0 + y_1)\,. \tag{4.14}$$

Veće vrijednosti broja n dovode do nepraktičnih formula, koje se ovdje ne navode.

U slučaju $n+1$ ekvidistantnih čvorova, uzastopnom primjenom osnovne trapezne formule po dijelovima dobiva se **ulančana** trapezna formula:

$$\int_{x_0}^{x_n} f(x)\,\mathrm{d}x \;\approx\; \frac{x_1 - x_0}{2}\left(y_0 + 2y_1 + \ldots + 2y_{n-1} + y_n\right). \tag{4.15}$$

Kako su trapezne formule vrlo jednostavne i pogodne za evaluaciju, a daju zadovoljavajuću točnost za dovoljno velik broj čvorova, nameće se ideja o njihovoj primjeni u rješavanju problema simboličke integracije primjenom genetskog programiranja.

Naime, taj problem može se poistovjetiti s aproksimacijskim problemom jer uzevši početni uzorak točaka (x_i, y_i), numeričkom integracijom može se izgenerirati novi uzorak $(x_i, I(x_i))$, gdje je $I(x_i)$ numerička vrijednost integrala na intervalu $[x_0, x_i]$ (Tablica 4.1).

Tablica 4.1: *Generiranje integriranog uzorka primjenom ulančane trapezne formule.*

x_i	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
y_i	-0.158	-0.443	-0.688	-0.872	-0.978	-0.996	-0.926	-0.773	-0.551	-0.279
$I(x_i)$	0.000	-0.030	-0.087	-0.165	-0.257	-0.356	-0.452	-0.537	-0.603	-0.644

Kako je $I(x_0) = 0$, primjetno je da ovakva metoda za rješenje daje jednu od beskonačno mnogo **primitivnih funkcija** aproksimacijske ili izvorne funkcije (ovisno o odabiru čvora x_0).

5 Programska rješenja

5.1 Problem umjetnog mrava

Među brojnim dostupnim programskim rješenjima s Interneta na bazi genetskog programiranja, svojom jednostavnošću i učinkovitošću se ističe rješenje problema **umjetnog mrava** (eng. *artificial ant*), kojemu je cilj u zadanom dvodimenzionalnom prostoru pojesti što više hrane.

Razvijena aplikacija pruža uvid u pronađena rješenja problema, kako preko mreže (eng. *grid*) na kojoj se prikazuje kretanje mrava, tako i u tekstualnoj formi gdje se ispisuje generirani program. Detaljna objašnjenja vezana za problem i za samu programsku implementaciju mogu se naći na [1], odakle se aplikacija može i preuzeti.

5.2 Aproksimacijski problem

U okviru kolegija Projekt razvijena je aplikacija *ApproxGP* [2], koja rješava problem aproksimacije zadanog uzorka točaka s realnom funkcijom realne varijable uz pretpostavljene operatore i operande. Ulazni parametri aplikacije su uzorak točaka, skupovi operatora i operanada, te standardni parametri poput veličine populacije, najveće dubine stabla jedinke, broja generacija i sl. Selekcija je **jednostavna** (formalno stvaranje novih generacija potencijalnih rješenja), a koriste se tri genetska operatora – reprodukcija, križanje i mutacija. Izvođenje aplikacije završava kada se izgenerira zadani broj generacija.

Rješenja problema se ispisuju u odgovarajućim izlaznim datotekama, a preporučuje se **grafička** predodžba i analiza pomoću programskog paketa *gnuplot* [14], koji se može preuzeti besplatno. U razvoju pomoćne aplikacije *Sampler* (za generiranje uzoraka) korištena je besplatna *Function parser* biblioteka, koja se uz adekvatnu dokumentaciju može preuzeti s [8]. Aplikacija *ApproxGP* je konzolna, a implementirana u programskom jeziku C++ uz nezaobilazno korištenje *Standard Template Libraryja* (STL) [9].

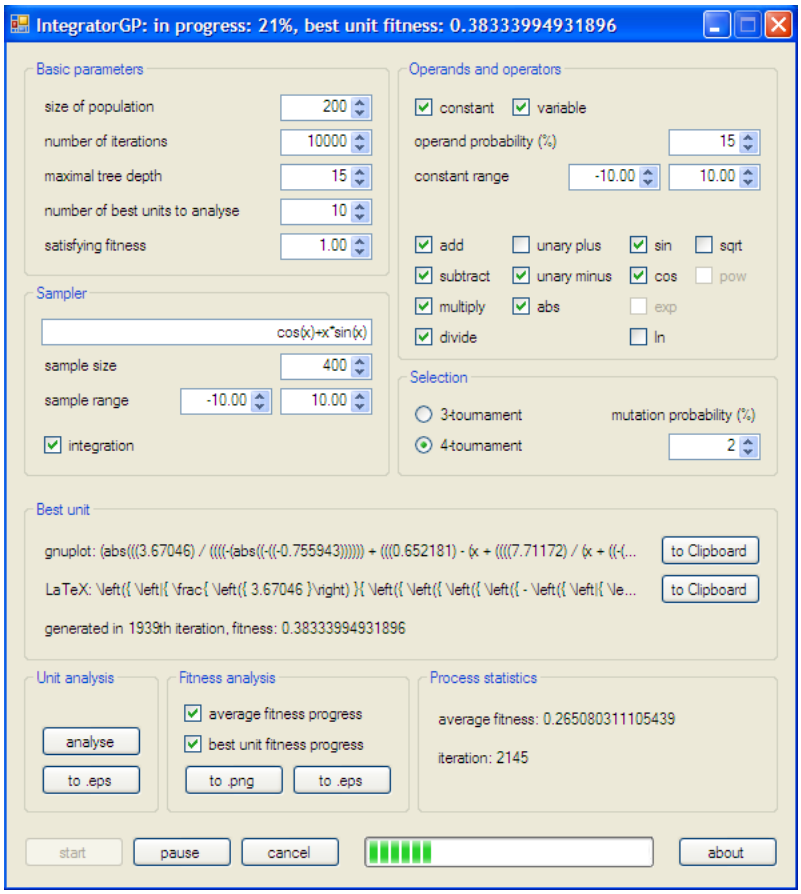
5.3 Problem simboličke integracije

Aplikacija *IntegratorGP* nastala je nadogradnjom navedene aplikacije *ApproxGP*. Preciznije, dodatno je implementirano **grafičko korisničko sučelje** (eng. *Graphical user interface* – GUI) i povećana funkcionalnost. Razvojna okolina je Microsoftov *Visual Studio 2008* s *.NET 3.5* radnim okruženjem (eng. *framework*).

5.3.1 Korisničko sučelje

Sučelje čini osnovna Windows-forma (eng. *form*) na čijoj su površini istaknute sve korisničke opcije (Slika 5.1), a podijeljene su u nekoliko logičkih grupnih područja (eng. *groupbox*). U gornjoj polovici forme smještene su sve one opcije koje određuju generativni proces, dok su u donjoj polovici forme smještena područja za praćenje rezultata tijekom rada, kao i njihovu analizu, te kontrolni gumbi (eng. *command button*), čije

je značenje i ponašanje intuitivno. Sučelje je, kao i programski kod, na **engleskom jeziku**, što je stvar programerske prakse.



Slika 5.1: *Prikaz korisničkog sučelja.*

5.3.2 Osnovni parametri

U području *Basic parameters* unose se osnovni parametri generativnog procesa. To su redom:

- veličina populacije (*size of population*),
- broj iteracija (*number of iterations*),
- najveća moguća dubina stabla jedinki (*maximal tree depth*),
- broj najboljih jedinki dostupnih u analizi rezultata (*number of best units to analyse*) te
- zadovoljavajuća dobrota (*satisfying fitness*), pri čijem se ostvarenju generativni proces zaustavlja.

Vrijednosti parametara unaprijed su **ograničene** na razumne vrijednosti.

5.3.3 Generiranje jedinki

Jedinke se generiraju **slučajnim procesima** na temelju definiranih skupova operanada i operatora u području *Operands and operators*. Implementirani operandi su:

- varijabla x (*variable*) te
- konstanta (*constant*), kojoj se podešava raspon (*constant range*).

Vrijednost konstante se generira **uniformno**, kao i jedan od operanada u slučaju kad generator jedinki zatraži generiranje operanda, što ovisi o vjerojatnosti odabira operanda (*operand probability*).

Uniformno se generiraju i operatori, a implementirani su:

- operatori zbrajanja, oduzimanja, množenja i dijeljenja (*add*, *subtract*, *multiply* i *divide*),
- unarnog plusa i unarnog minusa (*unary plus* i *unary minus*),
- apsolutne vrijednosti (*abs*),
- funkcija sinus i kosinus (*sin* i *cos*),
- logaritamska funkcija po prirodnoj bazi e (*ln*) te
- funkcija korjenovanja (*sqrt*).

Operator eksponencijalne funkcije e^x (*exp*) i funkcije x^y (*pow*) nisu implementirani do kraja pa u konačnici **nisu ni omogućeni**.

Prilikom evaluacije izraza generiranih ovim operandima i operatorima, posebno se obrađuju **iznimke**, poput dijeljenja s nulom, korjenovanja negativnog ili logaritmiranja nepozitivnog realnog broja, čime dotični operatori imaju ulogu **sigurnosnih operatora** (eng. *safety operators*) [5].

5.3.4 Funkcija dobrote i selekcija

Funkcija dobrote izražava **relativno odstupanje** od uzorka koji se pokušava aproksimirati. Definirana je formulom:

$$fitness(unit) := \frac{1}{1 + \frac{\sqrt{\sum_{i=0}^{n-1} (y_i - y_i')^2}}{y_{PP}}}, \tag{5.1}$$

gdje je n veličina uzorka, y_i y-vrijednost točke uzorka, a y_i' y-vrijednost koju evaluira promatrana jedinka za $x = x_i$, te y_{PP} *peak-to-peak* vrijednost.

Iz (5.1) vidljivo je da se radi o funkciji **normiranoj** na interval $[0, 1]$, gdje veća dobrota odgovara "boljoj" jedinki.

Područje *Selection* služi za podešavanje selekcije. Implementirane su dvije turnirske selekcije – 3-turnirska (*3-tournament*) i 4-turnirska (*4-tournament*). Također, uključen

je i genetski operator mutacije nad novim jedinkama (eng. *offsprings*), čija aktivnost ovisi o zadanoj vjerojatnosti izraženoj u postotcima (*mutation probability*).

5.3.5 Uzorkovanje i odabiranje problema

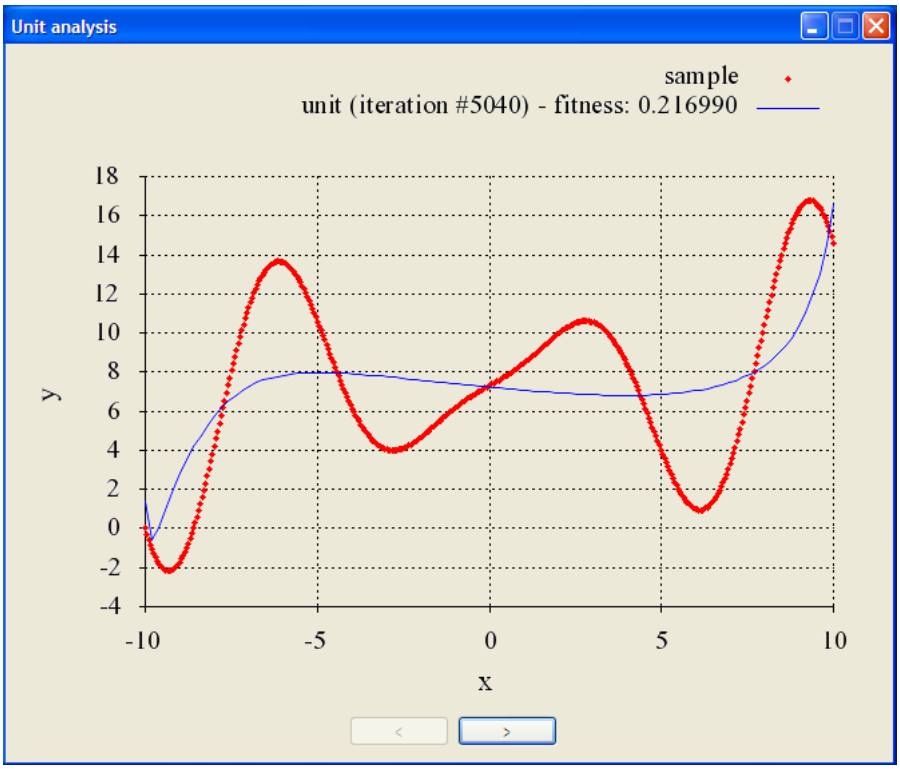
Područje *Sampler* služi za podešavanje i generiranje željenog uzorka za zadanu funkciju u standardnome programskom formatu (*C-style*). Određuje se veličina uzorka (*sample size*) i interval uzorkovanja (*sample range*). Također, odabire se i problem koji se rješava.

5.3.6 Praćenje rada i prikaz rezultata

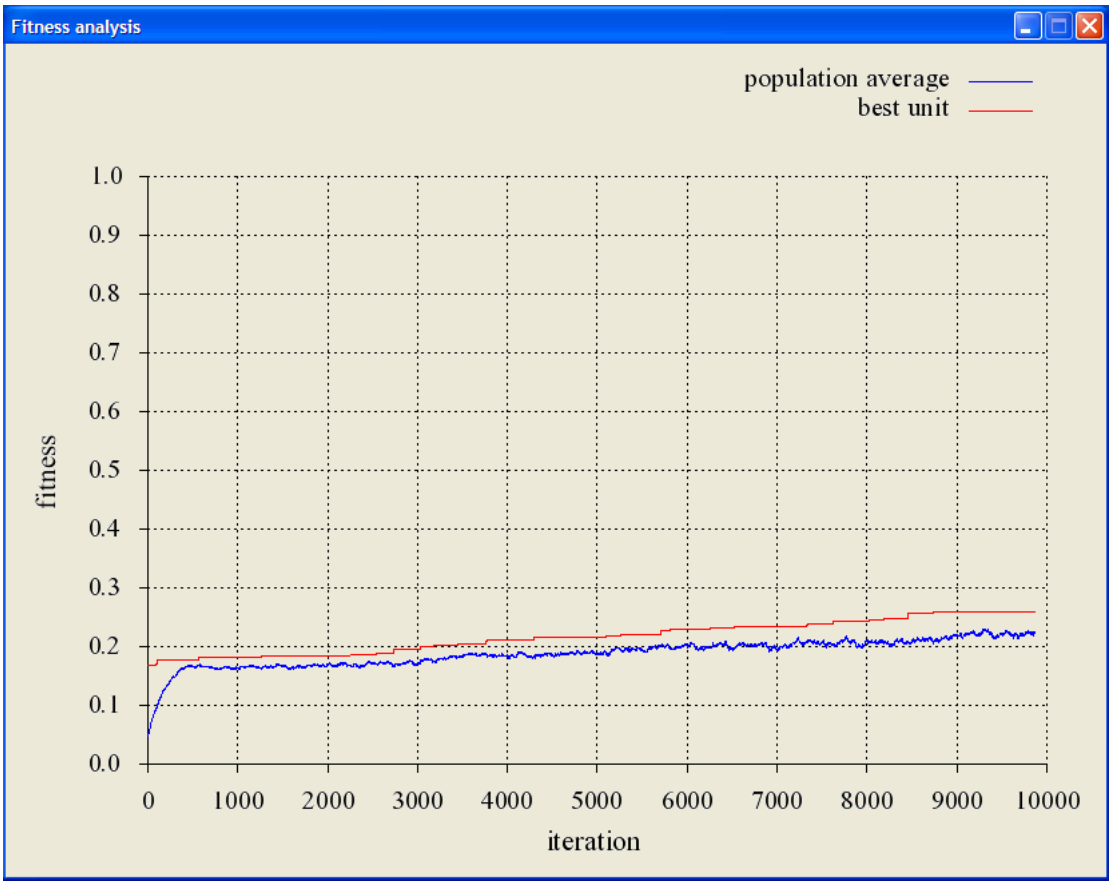
Područje *Best unit* tijekom generiranja jedinki daje informacije o najboljoj jedinki čiji se zapisi prilagođeni paketima *gnuplot* i LaTeX mogu kopirati na Windows *Clipboard*.

Druga dva područja, *Unit analysis* i *Fitness analysis*, omogućuju **vizualizaciju rezultata** – *Unit analysis* uspoređuje grafove nastalih jedinki s uzorkom koji se aproksimira (Slika 5.2), dok *Fitness analysis* daje uvid u ponašanje najbolje i prosječne dobrote (*best unit* i *population average*) tijekom generativnog procesa (Slika 5.3).

Područje *Process statistics* objedinjuje "u malom" postignute rezultate.



Slika 5.2: *Vizualizacija jedinki.*

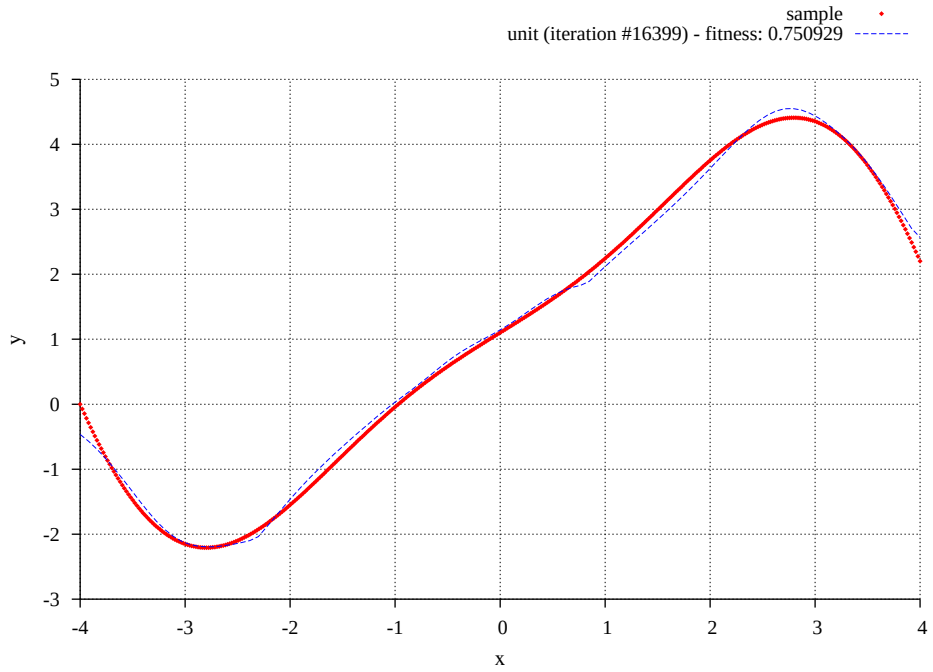


Slika 5.3: *Analiza dobrote.*

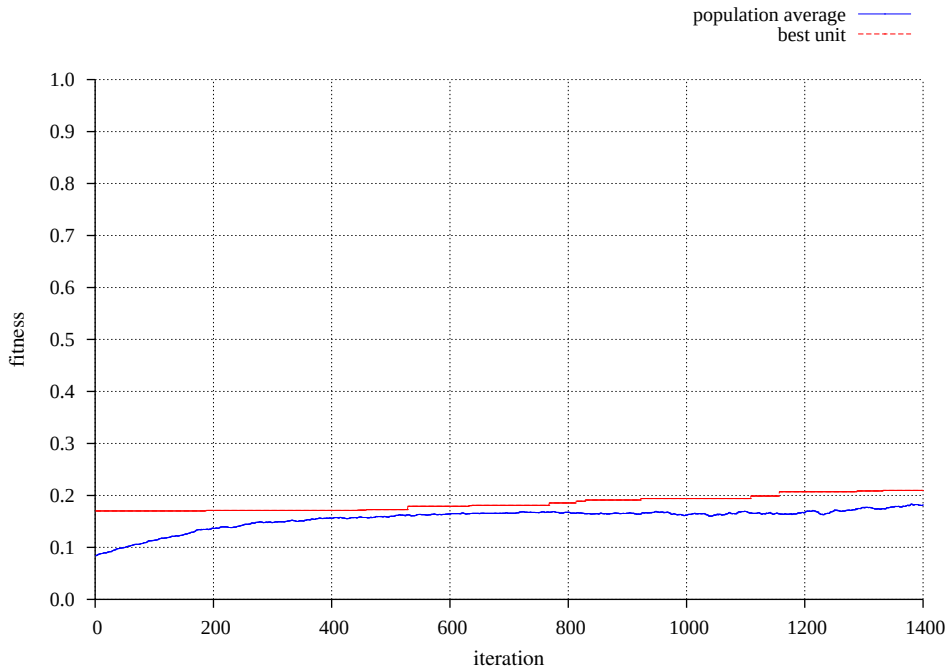
5.3.7 Ispitivanje i analiza dobivenih rezultata

Tijekom ispitivanja aplikacije došlo se do očekivanih zaključaka. Veći broj iteracija i bolji izbor operatora (prema prirodi uzorka) uzrokuju bolju dobrotu generiranih jedinki. Dubina stabala je **kritičan parametar** jer njeno povećanje povlači i povećanje prostora pretraživanja, što može narušiti rezultate evolucije. Također, bitno je napomenuti da 3-turnirska selekcija konvergira rješenjima **dvostruko sporije** od 4-turnirske, jer 4-turnirska selekcija ipak pridonosi većoj raznolikosti unutar populacije, a i u svakom koraku rezultira s dvije nove jedinke. Dobre vjerojatnosti mutiranja su **vrlo male** (svega par postotaka), jer takve pridonose izmjeni genskog materijala s vremena na vrijeme, a opet ga previše ne narušavaju.

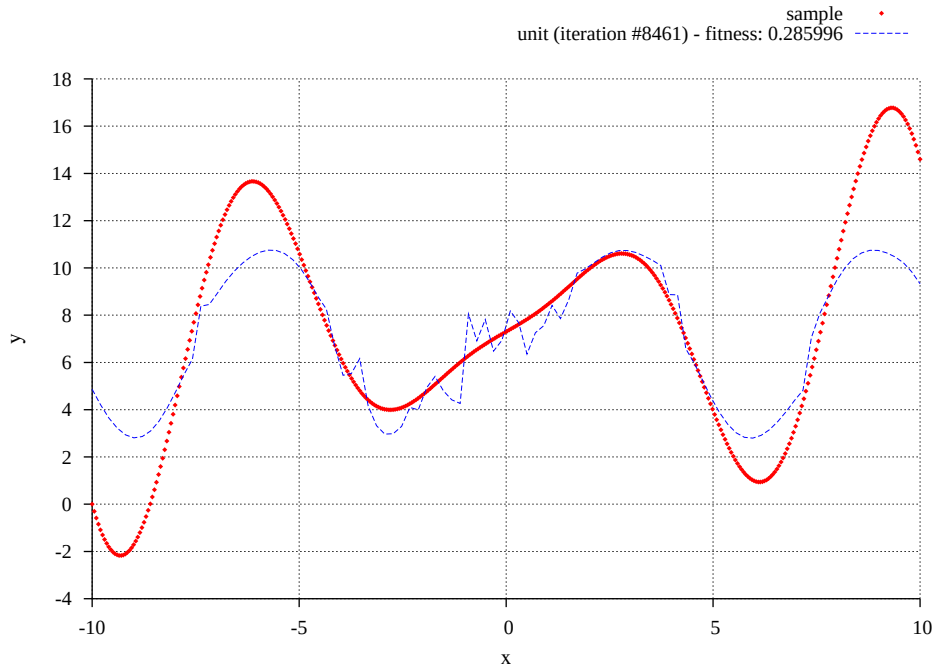
Slike 5.4 – 5.7 potvrđuju navedene zaključke pružajući uvid u različite rezultate aproksimacije i simboličkog integriranja ispitne funkcije $f(x) = \cos x + x \sin x$ na različitim intervalima. Rezultati su ovisni o zadanim parametrima.



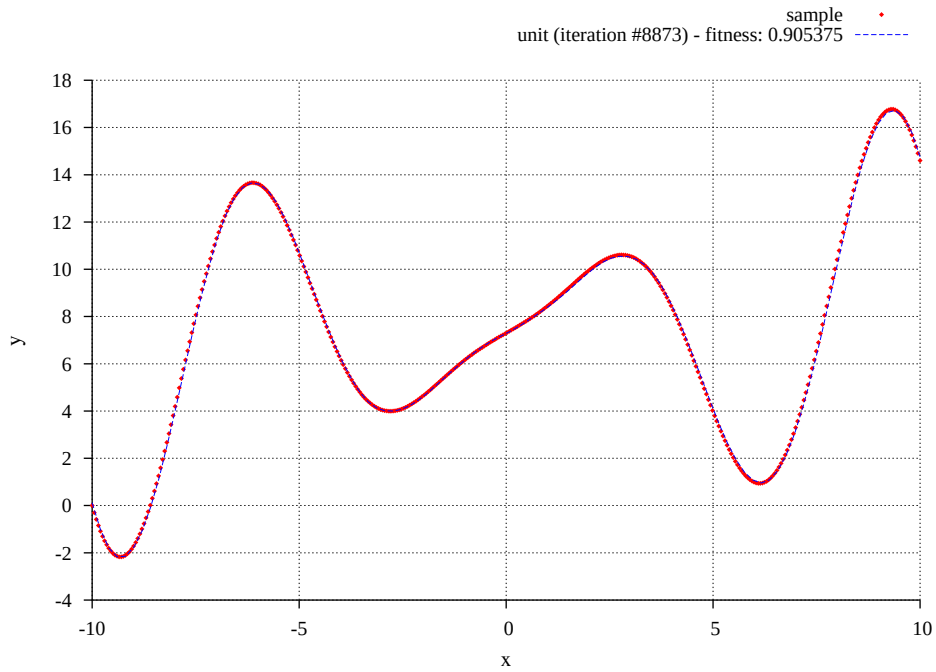
Slika 5.4: *Relativno prirodno ponašanje uzorka dovodi do brže i bolje konvergencije.*



Slika 5.5: *Izostanak trigonometrijskih operatora ugrožava dobrotu jedinki.*



Slika 5.6: *Sigurnosni operatori uzrokuju neprirodno osciliranje u okolini osi ordinata.*



Slika 5.7: *Gotovo savršena aproksimacija (operator dijeljenja nije uključen u proces).*

6 Zaključak

Iako je vrlo povezano s genetskim algoritmima, genetsko programiranje je u većini slučajeva učinkovitija metoda, i ima svoje čvrsto mjesto u obitelji evolucijskih algoritama. Radi se o činjenici da je produkt genetskog algoritma ipak algoritam, a genetskog programiranja postojani računalni program, koji se daljnjim optimizacijama može kako bolje, tako i brže razvijati. Zapravo, na prvi pogled dolazi se do paradoksa – programa koji sam sebe mijenja, no kad se malo bolje razmisli, vidi se da je riječ o **generalizaciji** genetskog programiranja u teoriji.

Postavlja se pitanje kako je moguće da jedan koncept vođen slučajnim procesima postiže iznenađujuće dobre rezultate, ali sama ideja darvinizma koja je ukorijenjena u genetskom programiranju očito djeluje i na neživoj materiji.

Genetsko programiranje je najučinkovitija metoda za nekoliko vrsta problema, od onih s egzaktnim rješenjima, do onih bez njih; te probleme koji su vremenski ovisni, jer je u njihovu rješavanju nužna karakteristika **dobre** i **brze prilagodbe**. Aproksimacijski problem i problem simboličke integracije samo su izdvojeni primjeri gdje genetsko programiranje postiže dobre rezultate, a vrlo slikoviti i pogodni za implementaciju i ispitivanje u kratkom vremenskom razdoblju.

Možda je najbitnije to što je genetsko programiranje između svih ostalih evolucijskih algoritama najkasnije doživjelo uspjeh, a daljnjim razvojem i primjenom teško je i zamisliti gdje mu je vrhunac.

Literatura

[1] Brian Connolly. *Survival of the Fittest: Natural Selection with Windows Forms.* http://msdn.microsoft.com/en-us/magazine/cc163934.aspx, kolovoz 2004.

[2] Luka Donđivić i Ivan Kokan. *Evolucijski algoritmi – Demonstracija rada evolucijskih algoritama: Genetsko programiranje.* http://www.autobrodogradnja.110mb.com/gp.html, siječanj 2008., Projekt.

[3] Ivan Ivanšić. *Numerička matematika.* Element, Zagreb, 1998.

[4] Ivan Kokan. *Numerička integracija*, svibanj 2007., Seminar.

[5] John R. Koza. *On the Programming of Computers by Means of Natural Selection.* A Bradford Book – The MIT Press, Cambridge, Massachusetts, 1992.

[6] Wolfram MathWorld. *Newton-Cotes Formulas.* http://mathworld.wolfram.com/Newton-CotesFormulas.html.

[7] Wolfram MathWorld. *Vandermonde Determinant.* http://mathworld.wolfram.com/VandermondeDdeterminant.html.

[8] Juha Nieminen i Joel Yliluoma. *Function parser for C++.* http://warp.povusers.org/FunctionParser/.

[9] SGI. *Standard Template Library Programmer's Guide.* http://www.sgi.com/tech/stl/.

[10] Wikipedia. *Evolutionary algorithm.* http://en.wikipedia.org/wiki/Evolutionary_algorithms.

[11] Wikipedia. *Genetic programming.* http://en.wikipedia.org/wiki/Genetic_programming.

[12] Wikipedia. *Lagrange polynomial.* http://en.wikipedia.org/wiki/Lagrange_interpolation.

[13] Wikipedia. *Population.* http://en.wikipedia.org/wiki/Population.

[14] Thomas Williams i Colin Kelley. *gnuplot homepage.* http://www.gnuplot.info/.