SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT

Evolucijski algoritmi

Demonstracija rada evolucijskih algoritama: Genetsko programiranje

Luka Donđivić, 0036412961

Ivan Kokan, 0036420101

Voditelj: *doc. dr. sc. Marin Golub*

Zagreb, siječanj 2008.

Sadržaj

1. Uvod

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

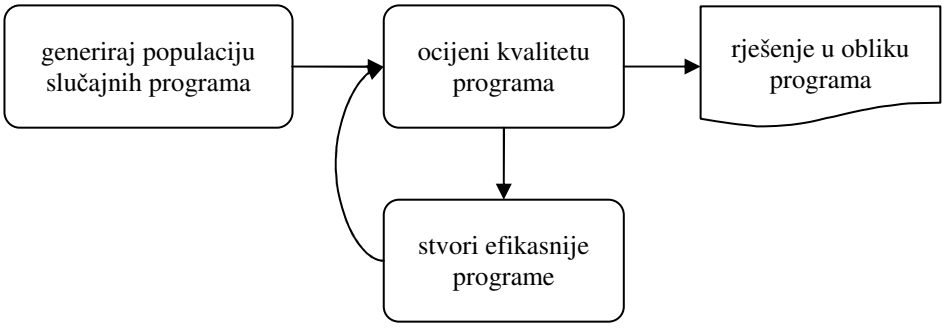
Temelj evolucijskih algoritama, kojima pripada i genetsko programiranje (eng. *genetic programming*), su pojave iz biološke evolucije – mutacija, rekombinacija, selekcija itd. Iako na prvi pogled ne postoji jasno vidljiva poveznica između spomenutih domena, nakon kraće analize ipak se dolazi do zaključka kako su principi darvinizma – *bolji opstaje, gori izumire* – idealni za pronalaženje *dovoljno dobrog* rješenja zadanog problema (ako ne i u potpunosti točnog).

Iako se teorijski razvijao paralelno sa ostalim evolucijskim metodama još od pedesetih godina 20. stoljeća, koncept genetskog programiranja je svoju 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, tako i kod samog Koze, koji neumorno nastavlja sa svojim životnim pozivom – pisanjem znanstvenih radova i prolongiranjem GP-a kao jedne od vodećih sila u razvoju umjetne inteligencije.

2. Osnovni pojmovi i definicije

Genetsko programiranje je automatiziran optimizacijski postupak razvoja računalnih programa, čija je namjena rješavanje većinom složenih problema iz područja računarstva, ali i problema na koje svakodnevno nailazimo. Koncept je zasnovan na općim idejama proizašlima iz teorije genetskih algoritama (eng. *genetic algorithms*), kao i drugih evolucijskih metoda. Najjednostavnije rečeno, konačni cilj GP-a (kao produkt) je univerzalni računalni program koji nalazi rješenja problema kao ulaznih podataka.

Pojednostavljeni shematski prikaz genetskog programiranja dan je na Slici 2.1.

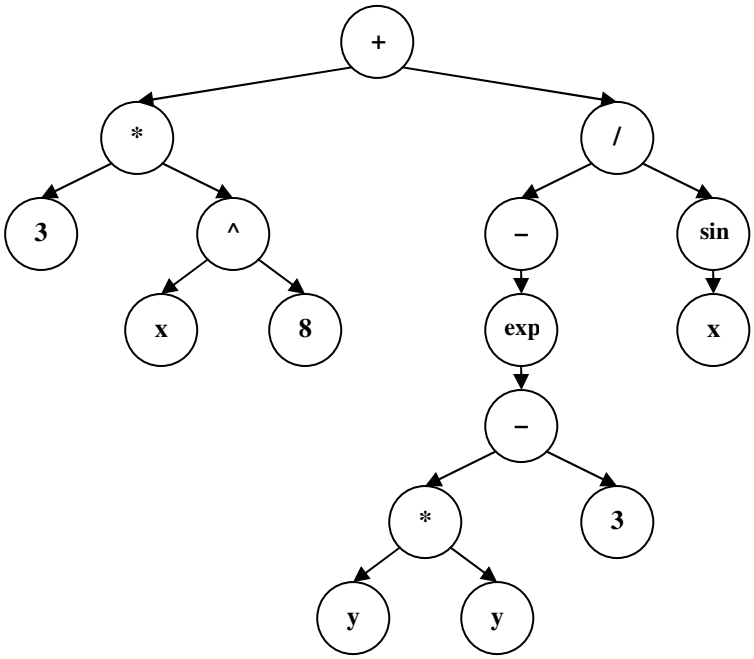


Slika 2.1. *Pojednostavljeni shematski prikaz genetskog programiranja*

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. iz linearne – uglavnom se radi o univerzalnom pristupu računalnom programu kao formalnom stablu u kontekstu teorije grafova. Naime, svaki računalni program se može prikazati kao stablo (eng. *tree*) ili šuma stabala (u širem smislu), gdje unutarnji čvorovi stabla (eng. *nodes*) imaju ulogu operatora (ili funkcije određenog broja varijabli), a listovi ulogu operanada. Pritom su skup operatora (eng. *function set* F) i operanada (eng. *terminal set* T) unaprijed definirani konačni skupovi.

U skladu s općenitom teorijom evolucijskih algoritama, možemo reći da kod GP-a ulogu kromosoma imaju nelinearni objekti – grafovi, odnosno stabla. Upravo svojstva stabala kao strogih matematičkih objekata, poput jednostavnog rekurzivnog obilaska, zapečatila su razvojni put genetskog programiranja u naglašenom smjeru.



Slika 2.2. *Stablasta predodžba izraza* $3x^8 + \left(-e^{y^2-3}\right) / \sin x$

2.2 GP kao poseban slučaj evolucijskog algoritma

Iterativni postupak, na kojem se bazira genetsko programiranje, u sebi sadrži jasno vidljive konture općenitog pseudokoda evolucijskih algoritama, a to su:

1) generacija inicijalne populacije rješenja,
2) analiza svake jedinke populacije i populacije općenito, te
3) odabir genetskog operatora i provedba odgovarajućih akcija nad pojedinom jedinkom (jedinkama).

```
Inicijaliziraj populaciju
Evaluiraj inicijalnu populaciju
Radi
  1. selektiraj
  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 Populacija

Biološka definicija populacije opisuje ju skupom jedinki iste vrste koje egzistiraju u istom prostoru. Razmnožavanjem unutar populacije (stvaranjem potomaka), ali i umiranjem, veličina

populacije je konstantna kroz generativni proces. Što se tiče genetskog programiranja, jedinku predstavlja računalni program, a prostor egzistencije jedinki iz iste populacije je jedna iteracija u optimizacijskom algoritmu. Prema tome, govori se o populaciji računalnih programa jedne iteracije algoritma.

2.4 Funkcija dobrote

Poznato je da se među biološkom vrstom na određenom staništu neke jedinke više ističu, imaju duži životni vijek, veće mogućnosti za stvaranje potomstva i sl. Kao i prije, povlači se analogija sa GP-om. Naime, u cilju rješavanja problema evolucijskim algoritmom, jako je bitno da bolja rješenja ostaju u daljnjem razmatranju, a ona lošija ne ostaju. 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 je jedan od ključnih problema genetskog programiranja, jer je potrebno da bude što "bolja", a što jednostavnija; pošto je njeno evaluiranje prisutno u svakom trenutku generativnog procesa. U stvarnosti, odabir i definiranje funkcije dobrote se prilagođava samom problemu i njegovim karakteristikama, a zadovoljenost navedenih oprečnih zahtjeva se pokušava uravnotežiti.

2.5 Genetski operatori

Evolucijski aspekt genetskog programiranja se očituje u načinu optimiranja promatranog programa, gdje su, u ulogama genetskih operatora, prisutne metode reprodukcije i križanja (eng. *crossover*). U nekim slučajevima se javljaju i mutacija, permutiranje i sl.

2.5.1 Reprodukcija

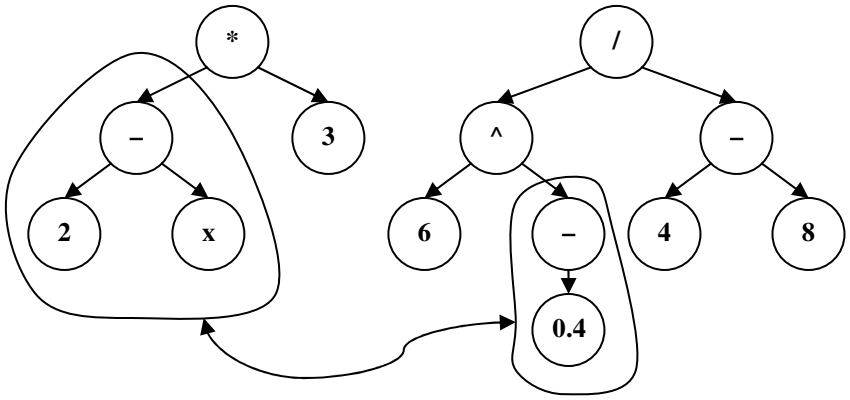
Reprodukcija je jednostavno kopiranje odabrane jedinke i njezino umetanje u novu populaciju.

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

2.5.2 Križanje

Križanje je analogno biološkoj spolnoj reprodukciji. Naime, u tom postupku dolazi do zamjene nekih podstabala odabranih dviju jedinki. Najučestaliji oblik je obavljanje opisane radnje na dva slučajno odabrana podstabla.

Parametri: p_c – vjerojatnost odabira križanja kao genetskog operatora,
p_u – vjerojatnost odabira pojedine jedinke,
p_n – vjerojatnost odabira pojedinog čvora jedinke kao korijena podstabla.



Slika 2.4. *Križanje (crossover)*

2.6 Selekcija

Konačno, tijekom generiranja populacija odvija se selekcija, i to na temelju vjerojatnosti odabira genetskog operatora, te dobrote jedinki. Uobičajeno je da selekcija direktno (proporcionalno) ovisi o dobroti.

Kao jedan od boljih načina selekcije spominje se sljedeći:

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

2.7 Kriterij prekidanja generativnog postupka

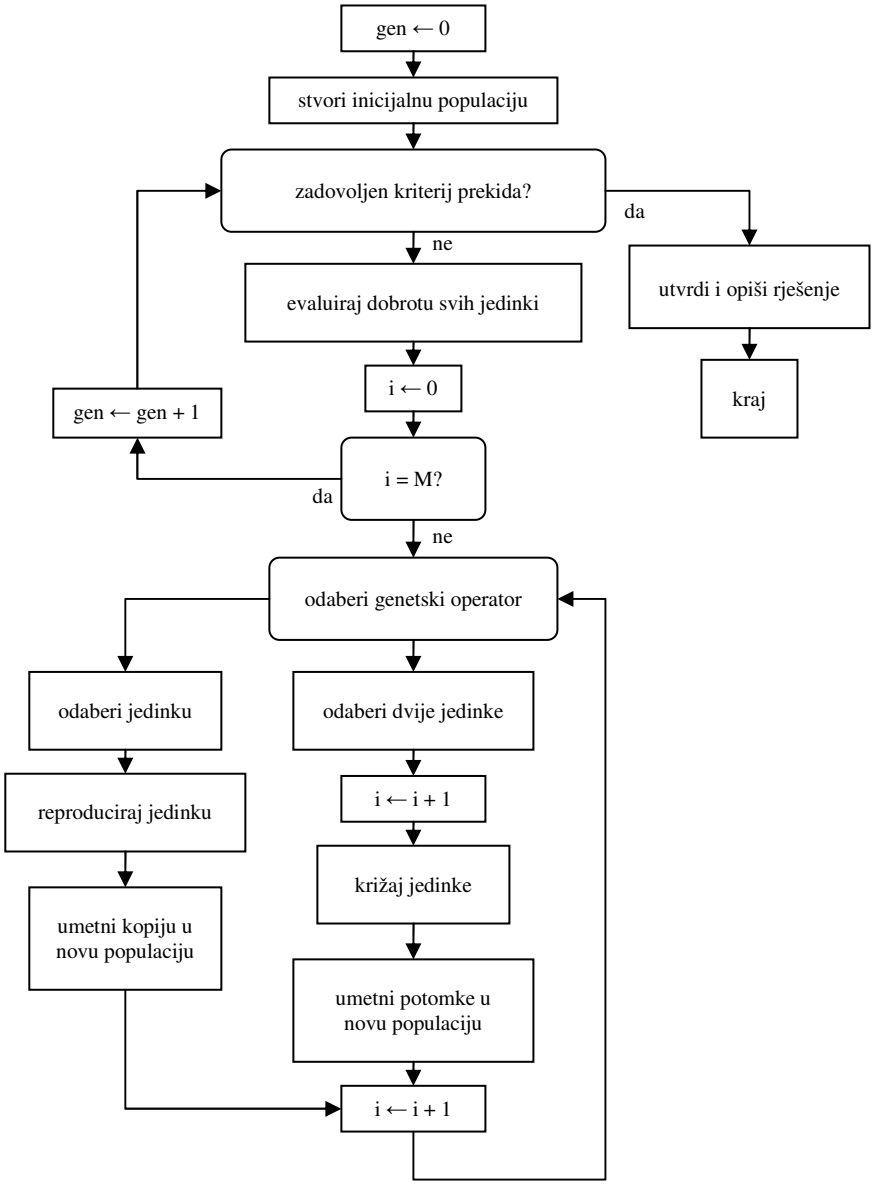
S obzirom na činjenicu da koncept genetskog programiranja s dobro definiranom funkcijom dobrote nad određenim problemom, i adekvatnim skupovima F i T, zadovoljavajuće brzo dolazi do rješenja problema, u velikom broju slučajeva generativni postupak se prekida pri ispunjenju određenog logičkog predikata. 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.

2.8 Utvrđivanje i opisivanje rješenja

Jednom kada se ispuni kriterij prekida generativnog procesa, iz populacije rješenja se odabire najkvalitetnije, te se opisuje, ako je ikako moguće. Moguće su i poželjne daljnje analize istog, kao i ponavljanje cijelog postupka zbog nezadovoljstva dobivenim rješenjem. Bolje rješenje se uglavnom pokušava naći redefiniranjem funkcije dobrote ili povećanjem supremuma broja iteracija.

2.9 Blok-dijagram GP-a

Nakon svega, genetsko programiranje se može predočiti sljedećim blok-dijagramom:



Slika 2.5. *Blok-dijagram genetskog programiranja*

3. Primjer GP-a

Pokušajmo koncept genetskog programiranja prikazati na dobro istraženom primjeru pozicioniranja kolica s raketom u ishodište promatranog koordinatnog sustava u što manjem vremenu, koja u danom trenutku t na beskonačno dugoj podlozi bez trenja imaju položaj $x(t) < 0$, brzinu $v(t)$ pozitivne orijentacije, a sve djelovanjem sile iznosa F proizvoljne orijentacije. Očito se radi o optimizacijskom problemu po varijabli vremena.

Formalno gledajući, problem možemo opisati sa dvije varijable stanja – $x(t)$ i $v(t)$. Ako sa m označimo masu kolica, tada je iznos akceleracije implicitno dan sa $F = ma(t)$. S druge strane, imamo jednadžbe stanja:

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

Problem je u odlučivanju o orijentaciji sile stalnog iznosa F koja djeluje na kolica. Definirajmo pozicioniranje u ishodište kao situaciju kada su $v(t)$ i $x(t)$ istovremeno jako mali, približno jednaki nuli.

Iz jednadžbi stanja jednostavno se dođe do optimalne strategije: ako je $-x(t) > \dfrac{v(t)^2 \operatorname{sgn} v(t)}{2F/m}$, na kolica djelujemo pozitivno orijentiranom silom F. Inače, radimo suprotno. Kao što vidimo, do algoritma nije bilo teško doći analitičkim putem.

Zamislimo sada da želimo napisati računalni program koji traži rješenje opisanog problema. Bez imalo razmišljanja nameće se rješenje ostvareno funkcijom koja nam vraća smjer u kojem trebamo djelovati silom F za ulazne parametre $x(t)$ i $v(t)$.

3.1 Reprezentacija jedinki

Svaku jedinku, tj. računalni program koji predstavlja strategiju rješavanja problema, prikazat ćemo na standardan način – pomoću stabla kojem su unutarnji čvorovi operatori, a vanjski čvorovi operandi.

3.2 Računanje dobrote

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

Nadalje, potrebna nam je metoda koja će moći izračunati dobrotu onih programa koji uspijevaju pozicionirati kolica u ishodište, kao i onih koji to ne uspijevaju. To možemo ostvariti računanjem stanja u diskretnim vremenskim koracima, pa ako se u nekom koraku nađemo u stanju dovoljno blizu ciljnog stanja smatramo da je program ispunio cilj, te je njegova dobrota jednaka utrošenom vremenu. U protivnom dobroti programa dodijelimo neku kaznenu vrijednost.

Konkretno, u ovom slučaju odabrat ćemo 20 slučajnih točaka iz podravnine $[-0.75, 0.75]^2$, a kaznena vrijednost u slučaju da program ne uspije pozicionirati kolica je 10 sekundi. Za odabrani uzorak optimalno rješenje postiže prosječno vrijeme od 2.02 sekundi.

3.3 Genetski operatori

Koristi se standardna operacija križanja među jedinkama.

3.4 Analiza rezultata

Već u trećoj generaciji jedinki dolazimo do rješenja vrlo blizu optimalnog, a u 33. generaciji nalazimo upravo 100% točno rješenje. Primijetimo da na početku nismo pretpostavili veličinu, oblik, ni strukturnu složenost rješenja, nego se ono samo pojavilo kao posljedica odabira funkcije dobrote. Za razliku od ovog primjera, kao rezultat genetskog programiranja rijetko kad ćemo dobiti 100% ispravno rješenje, već neko rješenje blizu optimalnog.

4. Programska rješenja

4.1 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*) kojem je cilj u danom dvodimenzionalnom prostoru pojesti što više hrane. Razvijena aplikacija pruža uvid u pronađena rješenja problema, kako preko *grida* na kojem se prikazuje kretanje mrava, tako i u tekstualnoj formi gdje se ispisuje generirani program.

Detaljna objašnjenja vezana uz problem i uz samu programsku implementaciju mogu se naći na http://msdn.microsoft.com/msdnmag/issues/04/08/GeneticAlgorithms/default.aspx, odakle se spomenuta aplikacija može i preuzeti.

4.2 Aproksimacijski problem

U sklopu Projekta razvijene su dvije aplikacije (*ApproxGP* i *Aproksimator_funkcija*) koje rješavaju problem aproksimacije zadanog uzorka točaka sa realnom funkcijom jedne realne varijable uz pretpostavljene operatore i operande. Ulazni parametri obiju aplikacija su već navedeni uzorak, skupovi operatora i operanada, te standardni parametri poput veličine populacije, najveće dubine stabla jedinke i broja generacija.

Bitna razlika između aplikacija je selekcija koja se koristi u generativnim postupcima. U slučaju aplikacije *ApproxGP* radi se o formalnom stvaranju novih generacija, dok je u slučaju aplikacije *Aproksimator_funkcija* implementirana turnirska selekcija. Također, izvođenje *ApproxGP*-a se završava kada se izgenerira zadani broj generacija, dok se kod druge aplikacije izvršavanje prekida u trenutku kad se rješenje nije popravilo za više od 1% u zadnjih NSEL selekcija, gdje je NSEL parametar zadan u programu.

Rješenja problema se ispisuju u odgovarajućim izlaznim datotekama, a preporuča se grafička predodžba i analiza pomoću programskog paketa *gnuplot*, kojeg se besplatno može preuzeti sa http://www.gnuplot.info/. U razvoju pomoćnih aplikacija (za generiranje uzorka) korištena je besplatna Function parser biblioteka, koja se uz adekvatnu dokumentaciju može preuzeti sa http://iki.fi/warp/FunctionParser/.

5. Zaključak

Iako je vrlo povezano sa 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 i bolje i brže razvijati. Zapravo, na prvi pogled, dolazimo do paradoksa – programa koji sam sebe mijenja, no, kad malo bolje razmislimo, vidimo da je to zapravo generalizacija samog genetskog programiranja u teoriji.

Genetsko programiranje je najučinkovitija metoda za nekoliko tipova problema, od onih sa egzaktnim rješenjima, do onih bez njih; te probleme koji su vremenski ovisni, jer je u njihovom rješavanju nužna karakteristika dobre i brze prilagodbe.

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

6. Literatura

[1] Genetic programming: On the Programming of Computers by Means of Natural Selection; John R. Koza; The MIT Press; 1992.

[2] Introduction to Genetic programming; Matthew Walker; http://www.massey.ac.nz/~mgwalker/gp/intro/introduction_to_gp.pdf; 2001.

[3] Genetic programming, Chapter 6; A. E. Eiben, J. E. Smith; http://www.cs.vu.nl/~eblaauw/ec0708/slides/Lecture06-Chapter6-GeneticProgramming.ppt

[4] Genetic programming – Wikipedia, the free encyclopedia; http://en.wikipedia.org/wiki/Genetic_programming; 2007.

[5] The Genetic programming Notebook; http://www.geneticprogramming.com/; 2003.

[6] Genetic-programming.org-Home-Page; http://www.genetic-programming.org/; 2007.