Evolucija programa upotrebnom genetičkog programiranja

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Kristina Petrović, Nikola Milovanović

20. jun 2020.

Sažetak

Genetičko programiranje predstavlja tehniku koja omogućava rešavanje određenog problema bez potrebe za stvaranjem programa koji rešava zadati problem, funkcioniše oslanjajući se direktno na teoriju Čarlsa Darvina i proces prirodne evolucije. U ovom radu biće predstavljen pojam genetičkog programiranja, prednosti i mane njegovog korišćenja kao i primena na praktičan problem simboličke regresije.

Sadržaj

1 Uvod **3**

2 Istorijat **3**

3 Elementi genetičkog programiranja **4**
3.1 Odabir podatkovnih i funkcijskih elemenata 4
3.2 Funkcija prilagođenosti 4
3.3 Parametri genetskog programa 5
3.4 Uslov zaustavljanja 5
3.5 Građa rešenja 5

4 Prednosti i nedostaci genetskog programiranja **6**

5 Reprezentacija jedinki **6**

6 Genetički operatori **7**
6.1 Selekcija 8
6.2 Ukrštanje 8
6.3 Mutacija 9

7 Simbolička regresija **9**
7.1 Implementacija problema u programskom jeziku Python . . 9
7.2 Pregled rezultata 9
7.2.1 Ciljna funkcija $x^4 + x^3 + x^2 + x$ 10
7.2.2 Ciljna funkcija $sin(\pi/4 + 2 * x)$ 11

8 Zaključak **12**

1 Uvod

Genetičko programiranje je tehnika koja omogućava rešavanje nekog problema bez potrebe za stvaranjem programa koji rešava zadati problem. Način rada genetičkog programiranja je uporaba evolutivnog procesa u stvaranju računarskinh programa. Evolucija se u prirodi javlja radi preživljavanja i reprodukcije sposobnih jedinki koje predaju svoj genetski kod sledećim generacijama. Kao posledica ovog procesa, genetski kod koji čini sposobnije jedinke ima mogućnosti da poveća svoj udeo u celokupnoj populaciji, što uzrokuje promene na nivou vrste.

Ono što čini genetičko programiranje posebnim u odnosu na ostale tehinike evolutivnih izračunavanja, je u činjenici da jedinke u populaciji genetičkog programa predstavljaju računarske programe, odnosno strukture koje se mogu jednoznačno preslikati u oblik pogodan za izvođenje na računaru. Kako bi se olakšao proces stvaranja novih programa iz jednoga ili više roditeljskih, programi su u većini primera genetskog programiranja napisani u obliku stabla. Novi programi se dobijaju jednostavnim manipulacijama nad već postojećim stablima, uklanjanjem grana iz jednoga stabla, dodavanjem tih grana na određeno mjesto u drugom stablu i slično. Ovaj proces kao rezultat takođe daje stablo i osigurava sintaksičku ispravnost dobijenih rešenja.

2 Istorijat

Sredinom 60-tih godina profesor Holland je prvi razvio metodu rešavanja problema optimizacije genetskim algoritmom. Njegov student Goldber razvio je i usavršio metodu za rešavanje kompleksnih problema optimizacije. 1989 godine Goldber objavljuje knjigu, "Genetski algoritmi u pretrazi, optimizaciji i mašinskom učenju", koja se poslije javlja kao inspiracija i startna tačka skoro svih metoda razvijenih na bazi genetskog programiranja.

Koza je 1996. godine započeo godišnju konferenciju o genetičkom programiranju koju je 1998. pratila pratila godišnja konferencija EuroGP i priva knjiga o genetičkom programiranju koju je uredio Koza. 1998 predstavljen je i prvi užbenik o genetskom programiranju. Rik Raolo nastavio je rad na genetskom programiranju što je dovelo do prvog specijalizovanog časopisa opšte prakse a 2003. nastala je godišnja radionica „Teorije i prakse genetičkog programiranja". Radovi o genetičkom programiranju se akrivno kreiraju i objavljuju na različitim konferencijama i u mnogim časopisima.

3 Elementi genetičkog programiranja

Prilikom rešavanja problema upotrebom genetičkog programiranja neophodno je definisati određene elemente sistema. Elementi koje je potrebno definisati su:

- Podacijski i funkcijski elementi rešenja
- Funkcija prilagođenosti
- Parametri genetskog programiranja
- Uslov zaustavljanja
- Građa rešenja

U slučaju rešenja predstavljenog stablom, podatkovni elementi su završni čvorovi stabla tj. listovi, koje nazivamo podatkovnim čvorovima, dok funkcijske elemente nazivamo funkcijskim čvorovima. Funkcijski čvorovi su u većini slučajeva unutrašnji čvorovi stabla, ali u nekim primerima funkcijski element može imati nula argumenata, pa samim tim mora biti izavršni čvor stabla. Odabir podatkovnih i funkcijskih elemenata određuje mogući prostor pretraživanja, kao i mogućnosti rešavanja problema.

Parametri algoritma uključuju veličinu populacije, maksimalnu dubina i širina stabla (opciono), verovatnoće primene operatora i slično. Određivanje građe rešenja podrazumeva definisanje struktura u obliku potprograma koje glavni program može pozivati, one se nazivaju automatski definisanim funkcijama (engl. automatically defined functions, ADF).

3.1 Odabir podatkovnih i funkcijskih elemenata

Najvažniji zahtev koji treba zadovoljiti prilikom odabira podatkovnih i funkcijskih elemenata je taj da je uz pomoć njih moguće izraziti rešenje problema kojeg pokušavamo da rešimo. Ovo svojstvo se naziva potpunost (engl. sufficiency), ono zavisi od samog problema i nije ga moguće definisati u opštem slučaju. U najvećem broju slučajeva u skup podatkovnih i funkcijskih elemenata stavljaju se svi elementi za koje se smatra da bi mogli biti od koristi za trenutni problem.

Drugo svojstvo koje skup elemenata rešenja treba zadovoljiti je zatvorenost (engl. closure). Zatvorenost zadatog skupa elemenata definiše se, za bilo koji funkcijski element, kao sposobnost prihvatanja rezultata od bilo kojeg drugog funkcijskog ili podatkovnog elemenata. Zadovoljavanje ovog svojstva zavisi o vrsti podataka svojstvenoj za određeni problem. U slučajevima mešanja različitih vrsta podataka, potrebno je za svaku moguću kombinaciju definisati transformaciju ili način inerpretacije ulaznih vrednosti. Čak i u slučaju jedinstvene vrste podataka, zatvorenost može biti narušena ponašanjem nekih funkcija, npr. rezultat deljenja nulom. Neka od rešenja ovakvih problema jesu menjanje definicija funkcija za kritične vrednosti, ili definisanje neadekvatnog tipa podataka.

3.2 Funkcija prilagođenosti

Funkcija prilagođenosti ili fitnes funkcija je najzaslužniji element u rukovođenju evolucijom i upravljanjem kretanja celokupne populacije. Ona treba da nagrađuje ne samo bolja rešenja, već i sva poboljšanja pronađena tokom evolucije. U većini primera genetskog programiranja određivanje fitnes funkcije oduzima najviše procesorksog vremena. Određivanje fitnes funkcije zavisi od samog primera, često se ispituje ponašanje rešenja na nizu ispitanih primera.

3.3 Parametri genetskog programa

Izbor odgovarajućih vrednosti zavisi od konkretne primene i ne postoji skup vrednosti parametara koji su najbolji za svaki problem. Postavljanje parametara postaje složenije sa povećanjem zavisnosti između njih, pa se u većini slučajeva koriste one vrednosti koje ispitivač smatra najprikladnijima za konkretan problem. Jedan od najvažnijih parametara u genetskom programiranju je veličina populacije, za netrivijalne probleme preporučuje se da ona bude veća od 1000 jedinki, a u nekim primerima korišćene su populacije i sa više od 100000 jedinki.

3.4 Uslov zaustavljanja

Najčešći oblik uslova zaustavljanja je dostizanje predefinisanog broja generacija. J. Koza u svom radu za sve eksperimente koristi broj od 50 generacija kao uslov zaustavljanja ako prethodno nije pronadjeno tačno rešenje. Argument iza ove vrednosti je zapažanje da nakon tog broja generacija genetski program gubi sposobnost pronalaženja bitno različitih rešenja, tj. da su eventualna naknadna poboljšanja vrlo mala. U najvećem broju slučajeva radi se dvostruka provera, algoritam se zaustavlja ili nakon zadatog maksimalnog broja genracija ili nakon što u zadatom broju uzastopnih generacija nije postignuto poboljšanje fitnes funkcije najbolje jedinke.

3.5 Građa rešenja

Ako posmatramo uobičajeni prikaz rešenja genetskog programiranja u obliku stabla, građa rešenja može podrazumevati definisanje potprograma odnosno, automatski definisanih funkcija (ADF). ADF je potprograma koji može biti pozvan od strane glavnog programa, a svi eventualni potprogrami, kao i glavni program, se izgradjuju dinamički tokom evolucije. U svom radu Koza daje niz primera u kojima genetski program sa automatskim funkcijama brže pronalazi rešenje ili uspeva rešiti teže probleme nego klasičan genetski program. Osim upotrebe automatskih funkcija, građa rešenja može uključivati i višestabalnu strukturu, u kojoj je svako pojedinačno stablo zaduženo za različit deo problema.

4 Prednosti i nedostaci genetskog programiranja

Prednost genetskog programiranja je što se njime može rešavati proizvoljni optimizacioni priblem, pri čemu postoji veliki broj mogućih nadogradnji i povećanja učinkovitosti. Jednom kada se pronađe rešenje, ono ne predstavlja rešenje odabranog problema samo za konkretne vrednosti, već predstavlja algoritam rešavanja koji se može primeniti nad proizvoljnim podacima. Genetskim programiranjem dobijamo čitavu populaciju rešenja, čime se omogućuje odabir više najboljih rešenja ukoliko je to potrebno. Još jedna prednost genetskog programiranja je jednostavnost izvođenja, jednom kada se definišu onsovni elementi izgradnja rešenja se u potpunosti prepušta genetskom programu.

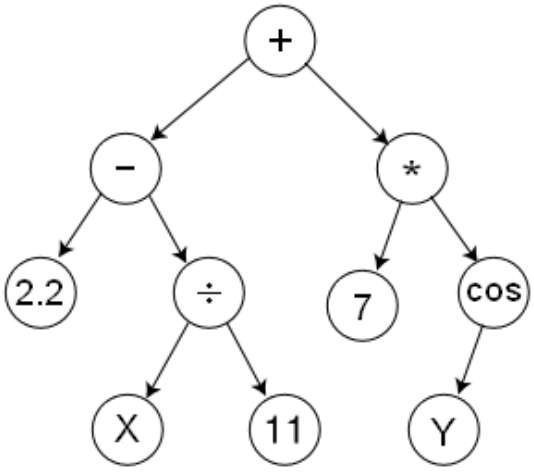
Jedan od najčešćih problema genetskog programiranja je preuranjena konvergencija. Ova pojava se dešava ukoliko jedna ili nekoliko relativno dobrih jedinki, ali ne i optimalnih, postepeno preovlada u populaciji i proces konvergira u lokalnom ekstremu. U tom slučaju mogućnosti za poboljšanje trenutnog rešenja su male, iz razloga što selekcija i ukrštanje u populaciji sa istim jedinkama nema efekta. Jedino mutacija može da doprinese izlasku iz date situacije, međutim u praksi je često bez efekta jer su nivoi mutacije, u većini slučajeva, veoma mali i prave neznatne razlike na genetskim materijalima, i dominantne jedinke ponovo eliminišu ostale jedinke iz populacije. Preuranjena konvergencija najčešće nastaje kao posledica primene proste ruletske selekcije. Ukoliko u populaciji postoji jedinka sa relativno velikom funkcijom prilagođenosti, ona će najverovatnije istisnuti sve ostale jedinke iz populacije.

Drugi čest problem jeste spora konvergencija, koja se dešava u kasnijoj fazi izvršavanja. Dešava se kada populacija postane dovoljno slična, ali nije postigla optimalno rešenje. Razlike izmedju najbolje jedinke i ostalih jedinki u populaciji su male, zobg čega postoji nedovoljni gradijent u fitnes funkciji koji bi pomogao algoritmu da dostigne optimalnu vrednost

5 Reprezentacija jedinki

U genetičkom programiranju jedinke se u memoriji najčešće predstavljaju upotrebom strukture stabla tako da se u listovima drveta nalaze terminali dok se u ostalim čvorovima nalaze funkcije. Terminali mogu sadržati vrednosti konstanti ili promeljivih, dok funkcije mogu biti aritmetičke, trigonometrijske, logičke ali takođe i različite programske konstrukcije poput uslovnih izraza ili petlji. Svako stablo mora imati definisan skup terminala i funkcija koji se naziva primitivni skup.

Drvo se lako može obilazitii na rekurzivan način, što takođe predstavlja jedan od razloga upotrebe stabla kao strukture.



Slika 2. Reprezentacija izraza (2.2-(x/11))+(7*cos(y)) upotrebom stabla.

6 Genetički operatori

Genetičko programiranje prirodno opisuje proces evolucije i varijacija gena je neophodne da bi se taj proces uspešno simulirao. Genetički operatori koji se koriste u genetičkim algoritmima su analogni onima u prirodnom svetu:

- Selekcija
- Ukrštanje
- Mutacija

Koristeći mehanizme evolucije u prirodi se stvaraju bolje i prilagodljivije jedinke koje će daljnjim ukrštanjem proizvesti nove jedinke sa kvalitetnijim svojstvima. Kao i u prirodi, tako i pri genetičkom programiranju samo najsnažniji preživljavaju. Upotrebom operatora selekcije omogućava se opstanak naprikladnijih gena dok operator ukrštanja predstavlja reprodukciju, rekombinaciju gena. Mutacija predstavlja izmene na jednom genu. Džon Koza nije koristio operator mutiacije i kao posledica njegovih preporuka mnoga ostvarenja genetičkog programiranja ne uključuju operator mutacije.

6.1 Selekcija

Selekcija predstavlja proves kojim se određeni pojedinci biraju iz trenutne generacije za roditelje naredne generacije. Pojedinci se biraju tako da bolje jedinke imaju veće šanse da budu odabrane. Najčešća metoda selekcije je selekcija turnira mada je dokaano da i druge metode kao što su ruletska selekcija, proporcionalna selekcija imaju dobar učinak.

Turnirska selekcija funkcioniše tako što se iz populacije na slučajan način izvuče određen broj jedinki. Izvučene jedinke se porede i biraju se najbolje jedinke na osnovu funkcije prilagođenosti. Najbolje izabrene jedinke će učestvovati u reprodukciji.

Ruletska selekcija funkcioniše tako što se izračunavaju verovatnoće sa kojima će jedinke biti izvučene. Verovatnoća da će jedinka biti izabrana je:

$$p_i = \frac{f(i)}{\sum_j^N f(i)}$$

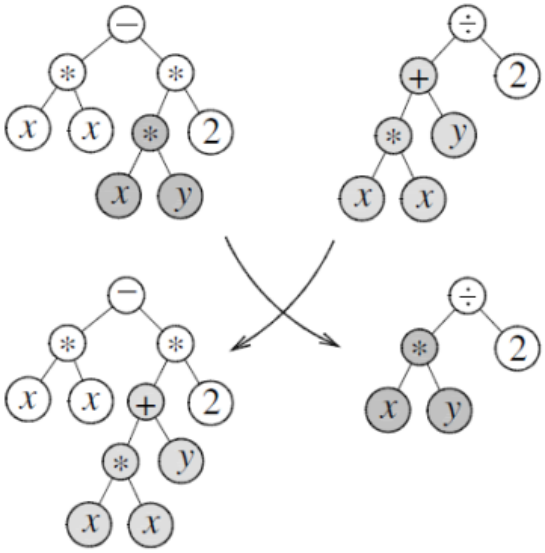
U formuli f(i) predstavlja funkciju prilagođenosti za i-tu jedinku a N broj jedinki.

Cilj selekcije je čuvanje i prenošenje dobrih osobina na sledeću generaciju jedinki. Selekcijom se biraju dobre jedinke koje će učestvovati u sledećem koraku, u ukrštanju. Na taj način se dobri geni čuvaju i prenose na sledeće populacije, a loši odumiru.

6.2 Ukrštanje

Ukrštanje je osnovni operator kojii omogućava stvaranje novih jedinki. Ukrštanje predstavlja razmenu genetičkog materijala izmeñu dve jedinke analogno istoimenom procesu nad živim jedinkama.

Kod genetičkog programiranja se najčešće na slučajan način odaberu tačke ukrštanja kod oba roditelja nakon čega se zamene podstabla roditelja sa korenom u tački ukrštanja.

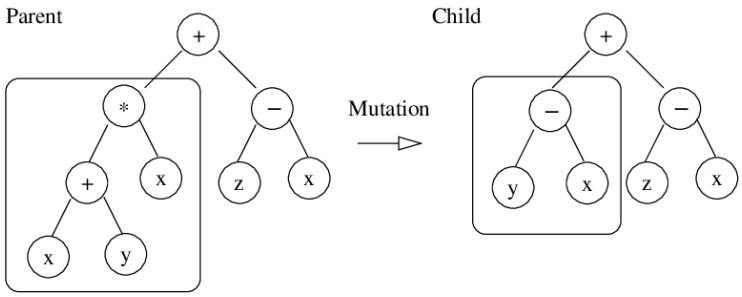


Slika 3. Ukrštanje dve jedinke zamenom podstabala.

6.3 Mutacija

Mutacija se primenjuje da jedinke vremenom ne bi bile previše slične. Takođe zbog izbegavanja lokalnog ekstremuma do kog bi moglo doći bez upotrebe mutacije. Njegovim delovanjem se vrši izmena slučajno izabranih gena jedinke. Kako deluje nad samo jednom jedinkom, mutacija je unarni operator koji kao rezultat daje izmenjenu jedinku.

U genetičkom programiranju mutacija se implementira tako što se podstablo zameni slučajno generisanim podstablom ili se čvor stabla zameni čvorom iz primitivnog skupa.



Slika 4. Mutacija jedne jedinke.

7 Simbolička regresija

Simbolička regresija predstavlja postupak pronalaženja matematičkog izraza iz empirijskih podataka. Za razliku od linearne regresije osim traženja parametara traži se i sam matematički model.

7.1 Implementacija problema u programskom jeziku Python

Pri rešavanju problema simboličke regresije neophodno je da se pronađe izraz koji se sastoji od članova primitivnog skupa, čija evaluacija predstavlja vrednost ciljne funkcije. Problem je rešen korišćenjem programskog jezika Python, pomoću metoda DEAP biblioteke.

Ciljne funkcije simboličke regresije su: $x^2 - 5x, x^4 + x^3 + x^2 + x, x^6 - 2 * x^4 + x^2, sin(\pi/4 + 2 * x), sin(3 * x^3 - x^2/7), N(0, 1)$

Generiše 180 tačaka iz intervala $[-\pi, \pi]$ koje će koristiti u izračunavanju i predstavljanju ciljnih funkcija i rešenja umesto promenljive x.

Svakoj jedinki se dodeljuje vrednost koja predstavlja njegovu prilagođenost, funkcija prilagođenosti se može računati na dva načina, jedan je izračunavanje kvadrata razlike evaluacije jedinke i ciljne funkcije a drugi način je računjanje asolutne vrednosti distance između evaluacije jedinke i ciljne funkcije.

Skup korišćenih terminala je: {x, -1, 0, 1}

Skup korišćenih funkcija je: {+, -, *, /, sin, cos}

7.2 Pregled rezultata

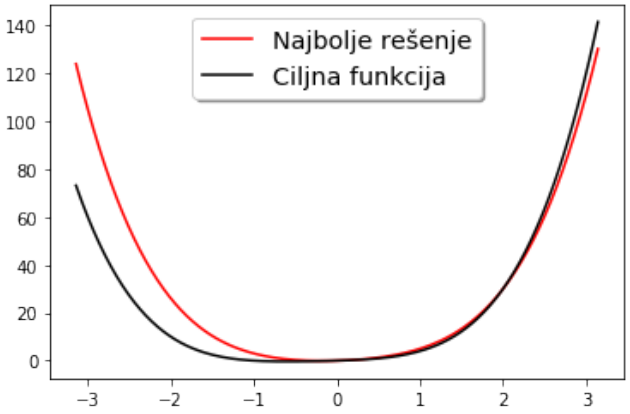
U narednom delu će biti prikazani rezultati upotrebe genetičkog programiranja korišćenjem različitih parametara i ciljnih funkcija.

7.2.1 Ciljna funkcija $x^4 + x^3 + x^2 + x$

Veličina populacije	Broj generacija	p mutacije	Turnirska selekcija	Metrika greške
50	30	0,3	2 jedinke	Kvadratna

Reprezentacija nabolje jedinke: $x^2 + (x + (x^2 * (1 + (1 + x^2))))$

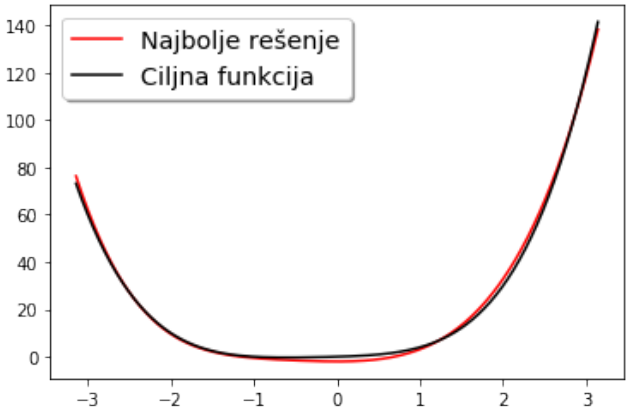
Iscrtavanje najbolje jedinke i ciljne funkcije:



Veličina populacije	Broj generacija	p mutacije	Turnirska selekcija	Metrika greške
300	30	0,1	2 jedinke	Kvadratna

Reprezentacija nabolje jedinke: $((1 - (1 - x^2)) - (cos(-x) + 1)) + ((x * (x + 1)) * (x^2 + sin(x)))$

Iscrtavanje najbolje jedinke i ciljne funkcije:

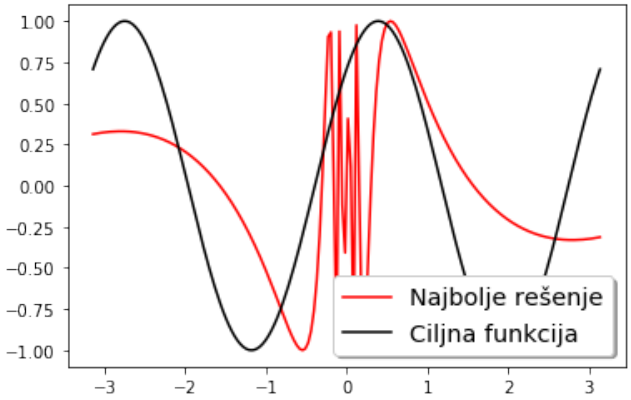


7.2.2 Ciljna funkcija $sin(\pi/4 + 2 * x)$

Veličina populacije	Broj generacija	p mutacije	Turnirska selekcija	Metrika greške
300	5	0	5 jedinki	Apsolutna

Reprezentacija nabolje jedinke: $sin(cos(x)/x)$

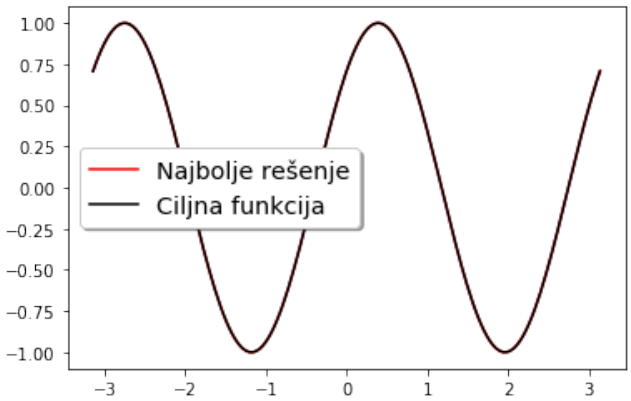
Iscrtavanje najbolje jedinke i ciljne funkcije:



Veličina populacije	Broj generacija	p mutacije	Turnirska selekcija	Metrika greške
300	30	0,3	5 jedinki	Apsolutna

Reprezentacija nabolje jedinke: $cos(sin(((-x + x) + 1)) + (-x - x))$

Iscrtavanje najbolje jedinke i ciljne funkcije:



8 Zaključak

Genetičko programiranje napredna je grana računalne znanosti. Od trenutka otkrića, koji zahvaljujemo Johnu R. Kozi genetičkom programiranju se posvećuje dosta pažnje. Njegova odlična primenjivost na raznim područjima svakim danom pridonosi sve više poklonika diljem svijeta.

Činjenica je da u svakodnevnom životu možemo naći živopisne primere na kojima se temelji genetičko programiranje je očaravajuća. Nastanak živoga svijeta možemo poistovetiti sa generiranjem nulte generacije. Nastanak prvih jednostaničnih organizama i njihova evolucija, preko koje se ukrštanjem i mutiranje gena došlo do složenijih organizama savršeno odgovara ukrštanju i mutaciji unutar genetičkoga programa. Još jedan dokaz sličnosti je temeljna nit vodilja genetičkog programa koja je preživljavanje najboljih rešenja, oslanjajući se na identičan zakon prirode koji nalazimo unutar istraživanja Čarlsa Darvina.

Literatura

[1] John Koza, Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems, 1990

[2] A Aho, M.S. Lam, R. Sethi, and J. Ullman. Compilers: principles, techniques and tools. Addison-Wesley Longman Publishing Co., 2006.

[3] https://en.wikipedia.org/wiki/Genetic_programming

[4] http://poincare.matf.bg.ac.rs/kartelj/nastava/RI2019/07.Genetsko.programiranje.pdf

[5] https://deap.readthedocs.io/en/master/

[6] https://www.mi.sanu.ac.rs/jkratica/