SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Generiranje formule prostih brojeva koristeći genetsko programiranje

Marko Jančec

Voditelj: *prof. dr. sc. Domagoj Jakobović*

Zagreb, travanj, 201

Sadržaj

1. Uvod

1.1 Prosti brojevi

Prostim brojem naziva se prirodni broj veći od 1 takav da su njegovi jedini djelitelji broj 1 i on sam. Prosti brojevi igraju veliku ulogu u matematici, posebice u teoriji brojeva, gdje se njihov značaj može izraziti preko osnovnog teorema aritmetike koji govori da se svaki prirodni broj može rastaviti na proste faktore na način jedinstven do na poredak faktora. To proste brojeve čini osnovnim 'građevnim blokovima' svih prirodnih brojeva.

Iako predstavljaju jedan temeljan koncept, mnoga pitanja oko prostih brojeva još uvijek su neodgovorena, primjerice Golbachova hipoteza, ili hipoteza o prostim brojevima blizancima, te pitanje kojim se ovaj rad bavi: formula prostih brojeva.

Formula prostih brojeva označava formulu koja sljedno generira samo proste brojeve, bez preskakanja i ponavljanja, dakle formulu oblika $f: \aleph \rightarrow P$, gdje *P* predstavlja uređeni skup prostih brojeva. Do danas takva formula, a da je realno primjenjiva (izračunljiva), nije poznata. No, poznata su različita ograničenja u vidu oblika takve formule, ukoliko ona postoji.

Mnogi pokušaji (Euler, Fermat, Legendre) kroz povijest bili su prikazivanje takve formule kao polinoma s cijelobrojnim koeficijentima, no ispostavilo se da takav polinom ne može predstavljati formulu za proste brojeve.

Millsova formula [formula] dokazano generira samo proste brojeve (iako ne cijeli skup, i ne slijedno), ali ne postoji način izračunavanja konstante theta bez poznavanja samog skupa prostih brojeva.

Nadalje, pronađen je sustav 14 diofantskih jednadžbi od 26 varijabli koji za neki broj k govori je k+2 prost ukoliko taj sustav ima cjelobrojno rješenje.

Postoji još mnogo različitih formula za proste brojeve[1], no ova tri navedena primjera pokazuju obilježja sviju njih: takve formule su ili nemoguće za ostvariti u danom obliku, generiraju samo neke proste brojeve (uz to mogu ili ne generirati ostale ili generirati i složene), zahtjevaju poznavanje prostih brojeva prije nego što mogu biti primjenjene, ili su u krajnjem slučaju previše složene za primjenu, tj. složenije od jednostavnih, ali sporih, testova prostosti ili generiranja prostih brojeva pomoću npr. Erostratosovog sita.

Nepostojanje, odnosno nepoznavanje formule prostih brojeva uzrokovalo je korištenje prostih brojeva u kriptografskim algoritmima s javnim ključem (primjerice RSA kriptosustav, Diffie-Helman sustav razmjene ključeva), kod kojih se kao ključ koristi umnožak dvaju (relativno velikih) prostih brojeva te činjenica da faktorizacija tog umnoška nije linearno riješiv problem. Na taj način samo osoba koja zna faktore može izvršiti dekripciju.

[1] http://mathworld.wolfram.com/PrimeFormulas.html

1.2 Genetsko programiranje

Genetsko programiranje specijalizirana je vrsta genetskog algoritma, u kojem se evolucijskim pristupom, vođenim funkcijom dobrote i primjenjujući genetske operatore križanja i mutacije nad jedinkama kroz generacije dolazi do rješenja (ili dovoljno dobre aproksimacije). Razlika između genetskih algoritama i genetskog programiranja jest u tome što su kod genetskog programiranja jedinke - programi. Njihov prikaz u klasičnom GP-u ostvaren je u obliku strukture stabla. Prilagođeni su i genetski operatori, pa se tako kod križanja vrši zamjena čvorova zajedno s njihovim podstablima. Mutacija može izmjeniti samo vrijednosti pojedinog čvora, a može i čitavo podstablo, ovisno o implementaciji. Funkcija dobrote izravno evaluira dobiveni program. Ostale stvari, poput kriterija zaustavljanja, selekcije, elitizma istovjetne su onima kod genetskih algoritama.

1.3 Slični radovi

U [2] autori se bave generiranjem polinoma koji generiraju proste brojeve. Za ovaj rad relevantniji je drugi dio istraživanja u kojem, pomoću prikaza polinoma u obliku digitalnog sklopa i multi-kromosomskog GP-a, uspijevaju u generiranju (iznimno složenog) polinoma koji slijedno generira prvih 208 prostih brojeva. Autori vjeruju da bi se na sličan način mogao ostvariti i polinom koji generira primjerice prvih 10000 prostih brojeva.

Također, u prvom djelu [2] autori se bave generiranjem polinoma koji generiraju niz neuzastopnih prostih brojeva (do niza duljine 43), slično kao i u [3], gdje se pomoću MP (hibridni evolucijski algoritam) generiraju polinomi koji za uzastopni niz prirodnih brojeva proizvode niz različitih (ne nužno slijednih) prostih brojeva (do niza duljine 59).

2. Opis korištenih postupaka i proračuna

2.1 Temeljna ideja

Polazna ideja je ne bazirati rješenje na nekom obliku funkcije (primjerice polinom s cjelobrojnim koeficijentima, za koji je poznato da ne može biti rješenje), već dati algoritmu da sam, na temelju skupa prostih brojeva, pokuša čim bolje aproksimirati željenu formulu.

Budući da su jedinke nad kojima algoritam radi matematičke formule, korišten je klasični oblik genetskog programiranja u kojem je jedinka prikazana kao stablo.

Početne jedinke generiraju se *ramped half-and-half* tehnikom, gdje se polovica stabla generira potpuno balansirano, a ostatak nasumično.

Broj jedinki u populaciji testiran je iz skupa [100,200,500,1000,2000,5000] u preliminarnom testiranju. Najbolje rezultate polučio je broj jedinki od 2000, sa značajnim napretkom u usporedbi s 1000. Zbog toga su u algoritmu korištene vrijednosti od 2000 i 5000 jedinki.

2.2 Genetski operatori

Genetski operatori standardni su operatori križanja, odnosno mutacije, sa vjerojatnostima prema preporukama iz [1] od 0.8 za križanje, te 0.1 za mutaciju. Preliminarnim testovima nad poznatim funkcijama ove vrijednosti pokazale su se kao dobre.

2.3 Skup završnih znakova

Skup završnih znakova (*engl. Terminal set*) čine varijable i konstante koje predstavljaju listove stabla. To su:

- n - redni broj prostog broja koji tražimo
- randint(-10, 10) - nasumični cijeli brojevi između -10 i 10
- random() - nasumični realni broj između 0 i -1

2.4 Skup funkcija

Set funkcija uključuje:

- osnovne matematičke operacije (zbrajanje, oduzimanje, množenje, dijeljenje, modulo)
- kvadratni korijen
- prirodni logaritam

Ovdje nailazimo na problem kojim se treba izravno pozabaviti prilikom definiranja skupa funkcija. Naime, kako kod generiranja stabla te primjene genetskih operatora dolazi do nasumične raspodijele funkcija i završnih znakova, odnosno razmjene podstabala na bilo kojoj poziciji, potrebno je osigurati da argumenti funkcija budu iz njihovih domena, odnosno da funkcije vraćaju ispravne (definirane) vrijednosti.

To svojstvo nije lako osigurati prilikom samog generiranja stabla ili primjene genetskih operatora, već se osigurava prilikom evaluacije, najčešće na jedan od dva načina:

1. Uporabom zaštićenih operatora - izvodi se 'omatanjem' funkcije tako da, ukoliko argumenti funkcije nisu iz područja domene, funkcija vraća neku definiranu vrijednost. Primjerice, kod zaštićenog dijeljenja a/b, ukoliko je b jednak 0, funkcija vraća neutralni element 1.
2. Pridjeljivanjem vrlo velike (maksimalne), odnosno male (minimalne) vrijednosti (ovisno o tome je li cilj minimizirati ili maksimizirati funkciju dobrote) funkcije dobrote takvoj jedinki za koju evaluacija vrati nedefiniranu vrijednost. Na taj način izravno govorimo algoritmu da je takva jedinka vrlo loša, i primjenom selekcije vrlo je mala vjerojatnost da će preživjeti do sljedeće generacije.

U ovom radu koristi se uporaba zaštićenih operatora.

2.5 Funkcija dobrote

Implementirane i korištene su dvije različite funkcije dobrote:

1. **Srednja kvadratična pogreška** (*engl. RMSE - Root-mean-square error*) - kao mjera dobrote uzima se suma korijena kvadrirane razlike između stvarne i predviđene vrijednosti svih članova promatranog niza. Pokazalo se da takav pristup funkcionira za manji broj članova niza (~10) za koje se gradi funkcija, dok kod većeg broja (~20) dolazi do interpolacije tako da se izgrađena funkcija prilagođava skupu.
2. **Jedinična funkcija dobrote** - namjena funkcije je eliminirati problem prevelikog prilagođavanja (generalizacije) generirane funkcije. To je ostvareno tako da se svako odstupanje od stvarne vrijednosti kažnjava konstantom kaznom. Tako se izjednačavaju sve vrijednosti funkcije za određeni ulaz koje odstupaju od stvarne vrijednosti, bez obzira na iznos odstupanja. Isti princip koristi se i u [1], te sličan u [2] kao broj uzastopnih prostih brojeva.

Očito, koristeći obje funkcije dobrote, cilj je minimizirati njen iznos, jer u oba slučaja ona predstavlja pogrešku koju proizvodi generirana formula.

2.6 Kriteriji zaustavljanja

Dva su kriterija zaustavljanja: dosegnut iznos pogreške od 0 (funkcija daje ispravne rezultate na cijelom zadanom skupu), ili dosegnut zadani maksimalan broj generacija. Poželjno je očito da algoritam završi na prvi način, no u općenitom slučaju GP nakon nakon određenog broja generacija prestaje konvergirati ka rješenju, te se tada prekida. Granica je odabrana na 50 generacija (prema [1]) što se u praksi također pokazalo kao dobar odabir, budući da u većini pokretanja algoritam završava na konačnoj vrijednosti između 10 i 30 generacije.

2.7 Implementacija

GP je implementiran koristeći pyEvolve modul u programskom jeziku Python. Također, koristi se matplotlib za vizualizaciju rješenja i crtanje grafova, te sqlite3 adapter i baza podataka za spremanje rezultata.

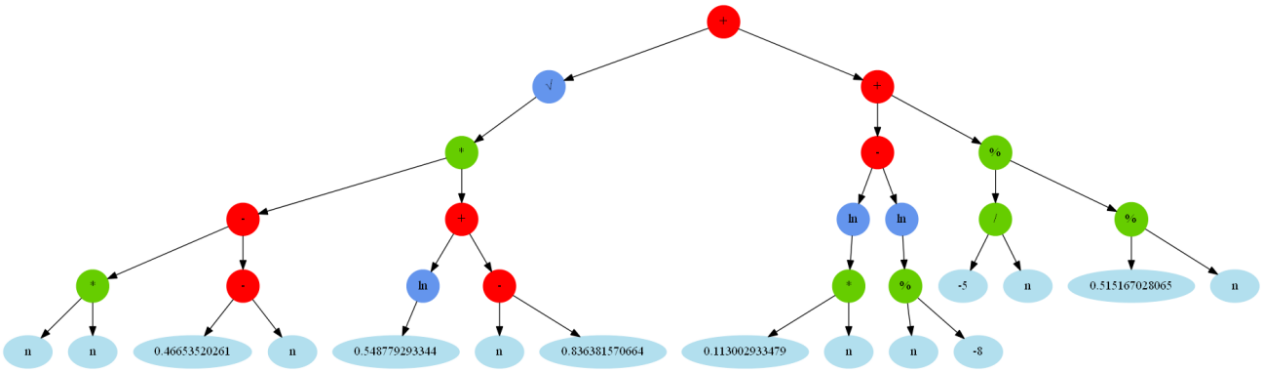
3. Rezultati

3.1 p(n) za n iz skupa [1, 10]

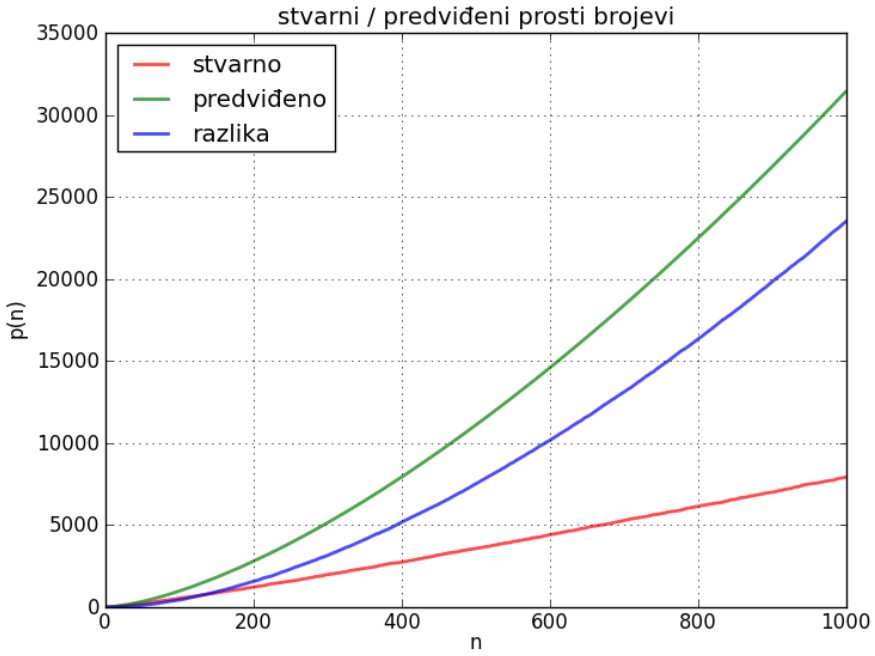
Redni broj	Broj pokretanja	Maks. dubina stabla	Veličina populacije	Funkcija dobrote	Dobrota najbolje jedinke
1	25	3	2000	1 (RMSE)	5
2	25	3	5000	1 (RMSE)	4
3	25	4	2000	1 (RMSE)	2
4	25	4	5000	1 (RMSE)	3
5	**25**	**5**	**2000**	**1 (RMSE)**	**0**

Za prvih 10 prostih brojeva algoritam pronalazi rješenje. Korištena je samo RMSE funkcija dobrote. Veća populacija ne doprinosi algoritmu.

Formula rješenja: gp_add(gp_add(gp_mod(gp_mod(n, 0.515167028065), gp_div(n, -5)), gp_sub(gp_ln(gp_mod(-8, n)), gp_ln(gp_mul(n, 0.113002933479)))), gp_sqrt(gp_mul(gp_add(gp_sub(0.836381570664, n), gp_ln(0.548779293344)), gp_sub(gp_sub(n, 0.46653520261), gp_mul(n, n)))))



slika 1 - AST rješenja



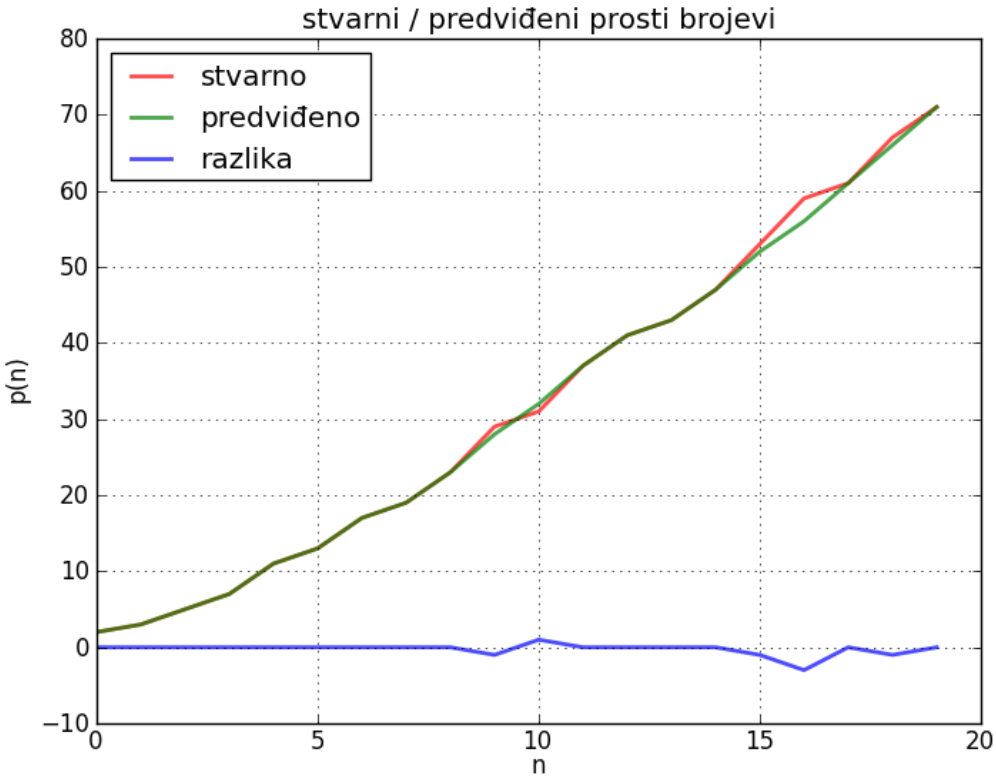
slika 2 - evaluacija formule na skupu od 1000 prostih brojeva

3.2 p(n) za n iz skupa [1, 20]

Redni broj	Broj pokretanja	Maks. dubina stabla	Veličina populacije	Funkcija dobrote	Dobrota najbolje jedinke
1	25	4	2000	1 (RMSE)	26
2	25	5	2000	1 (RMSE)	22
3	25	5	2000	2 (JED)	5
4	25	5	5000	1 (RMSE)	16
5	25	6	2000	1 (RMSE)	18
6	25	6	2000	2 (JED)	6
7	25	6	5000	2 (JED)	7
8	25	7	2000	2 (JED)	8
9	25	9	2000	2 (JED)	6
10	25	11	2000	1 (RMSE)	20
11	25	12	2000	1 (RMSE)	24
12	25	13	2000	2 (JED)	7

Već za prvih 20 prostih brojeva algoritam ne uspijeva pronaći rješenje. U tablici zelenom bojom označena su pokretanja algoritma kod kojih je korištena jedinična funkcija dobrote. Ponovno je vidljivo da veća populacija ne doprinosi pronalasku rješenja. Također, porastom dubine stabla dobrota se smanjuje. To je očekivano jer veća dubina stabla znači da algoritam može detaljnije prilagoditi funkciju ulaznom skupu podataka. No, nakon neke dubine povećanje maksimalne dubine stabla više ne doprinosi poboljšanju kvalitete.

Jedinična funkcija dobrote ponaša se bolje od RMSE.



slika 3 - najbolja jedinka za prvih 20 prostih brojeva

4. Zaključak

Formula prostih brojeva pokazala se zaista kao složen problem. Genetsko programiranje, barem u osnovnom obliku koji je korišten u ovom radu, nije se pokazalo kao dovoljno dobar pristup rješavanju ovog problema, a čak i korištenjem naprednijeg pristupa (kao primjerice u [2]) dobivena rješenja ne djeluju dovoljno skalabilno i kose se sa zahtjevom da formula prostih brojeva bude jednostavna i jednostavno izračunljiva.

5. Literatura

[1] Riccardo Poli, William B. Langdon, Nicholas F. McPhee, A Filed Guide to Genetic Programming, 2008. , ISBN 978-1-4092-0073-4

[2] James Alfred Walker, Julian Francis Miller, Predicting Prime Numbers Using Cartesian Genetic Programming (EuroGP, 10th European Conference on GP), 2007., ISBN 978-3-540-71602-0

[3] Emad Mabrouk, Julio César Hernández-Castro, Masao Fukushima, Prime number generation using memetic programming, 2011., SpringerLink

[4] Andrej Dujella, Uvod u teoriju brojeva, skripta

6. Sažetak

Pitanje (ne)postojanja formule prostih brojeva, tj. formule lika $f: ℵ \rightarrow P$ (gdje je P skup prostih brojeva) neodgovoreno je još od samih početaka matematike. I dok dio matematičara vjeruje da nešto toliko fundamentalno u znanosti mora imati uzorak i pokoravati se nekoj pravilnoj raspodijeli, drugi smatraju kako je upravo taj slučajni obrazac ključ njihovih specifičnih svojstava. Cilj ovog rada jest, koristeći genetsko programiranje, pokušati dobiti uvid u složenost i izgled takve formule, te generirati formulu koja što više odgovara takvom opisu, uzimajući pritom u obzir poznata matematičke ograničenja.