SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4331

SIMBOLIČKA REGRESIJA ZA PREDVIĐANJE VREMENSKIH NIZOVA UPORABOM GENETSKOG PROGRAMIRANJA

Domagoj Penić

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 9. ožujka 2016.

ZAVRŠNI ZADATAK br. 4331

Pristupnik: **Domagoj Penić (0036477119)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Simbolička regresija za predviđanje vremenskih nizova uporabom genetskog programiranja**

Opis zadatka:

Opisati postupke predviđanja vremenskih nizova i regresije uz pomoć evolucijskih algoritama. Posebnu pažnju posvetiti uporabi genetskog programiranja i srodnih tehnika u navedenom području. Ostvariti programski sustav koji omogućuje primjenu na proizvoljnom problemu predviđanja uz pomoć genetskog programiranja. Usporediti učinkovitost sustava s obzirom na genetske operatore, postupke selekcije i postojeće algoritme predviđanja. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:

Izv. prof. dr. sc. Domagoj Jakobović

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

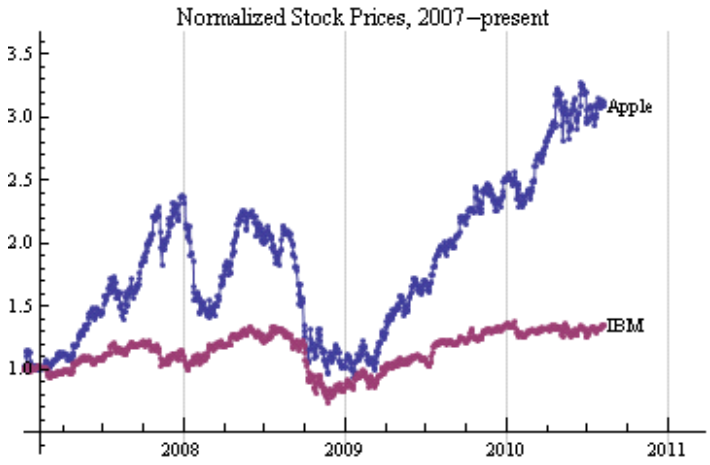
Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Sadržaj

1. Uvod

Vremenski su nizovi (eng. *time series*) nizovi kronološki uređenih vrijednosti neke pojave, najčešće dobivenih uzastopnim mjerenjima tijekom određenog vremenskog intervala. Veliki dio podataka iz stvarnog svijeta može se predstaviti vremenskim nizovima, kao što su kretanje cijena dionica, količina ugljikovog dioksida u zraku, broj putnika u zračnom prometu i slično. Primjer vremenskog niza cijena različitih dionica prikazan je na slici 1.1.



Slika 1.1 Vremenski niz cijene dionica

Kod mnogih vremenskih nizova stvara se potreba za analizom te predviđanjem njihovih vrijednosti. Analiza (eng. *time series analysis*) se odnosi na metode kojima se u vremenskim nizovima pokušavaju uočiti smislene statistike te određene karakteristike podataka, a prognoza (eng. *time series forecasting*) se bavi izradom modela kojim bi se predviđale buduće vrijednosti na temelju prijašnjih.

Jedna od ne tako čestih korištenih metoda u predviđanju vremenskih nizova jest genetsko programiranje (eng. *genetic programming*), metaheuristička metoda

optimizacije koja spada u skupinu evolucijskih algoritama (eng. *evolutionary algorithms*), algoritama koji traže rješenja optimizacijskih i sličnih problema koristeći tehnike inspirirane prirodnom evolucijom. Najčešće se ove metode koriste za rješavanje problema simboličke regresije, koji se može smatrati ekvivalentnim problemu predviđanja vremenskih nizova.

U ovom radu demonstrira se primjena genetskog programiranja za predviđanje različitih vremenskih nizova, prikazuju se rezultati dobiveni uporabom različitih parametara te njihova usporedba. Također, u svrhu izvođenja različitih mjerenja, ostvaren je i programski sustav koji nudi grafičko korisničko sučelje za pokretanje učenja s određenim parametrima.

Nakon ovog poglavlja slijedi opis genetskog programiranja i njegove najvažnije karakteristike. Zatim slijedi opis ostvarenog programskog sustava te korišteni skupovi podataka. Iza toga slijede opisi mjerenja, dobiveni rezultati i njihova usporedba s već postojećim te zaključak i popis korištene literature.

2. Genetsko programiranje

Genetsko je programiranje (GP) tehnika inspirirana biološkom evolucijom koja pronalazi jedinke koje predstavljaju računalne programe za rješavanje različitih korisničkih problema. Imitira se prirodni proces evolucije tako da se od početnog skupa rješenja kroz određeni broj iteracija nastoji dobiti rješenje koje dovoljno dobro odgovara danom problemu. Evolucija u prirodi uzrokuje da sposobne jedinke preživljavaju i reproduciraju se prenoseći tako svoj genetski kod sljedećim generacijama. S vremenom sposobnije jedinke povećavaju udio svog genetskog materijala u populaciji što može dovesti do promjena na razini vrste. Kao i ostale metaheurističke metode, ni GP ne garantira da je nađeno rješenje optimalno, ali u prosjeku daje prihvatljiva rješenja dovoljno blizu globalnom optimumu, a izbjegava zapinjanje u lokalnim optimumima koje se događa algoritmima lokalne pretrage.

Cilj genetskog programiranja, kao i svih evolucijskih algoritama, jest napraviti početni skup jedinki, populaciju, te kroz iteracije birati bolje jedinke, a odbacivati lošije. Bitno je spomenuti da se ne smiju sve loše jedinke odbaciti jer, iako je neka jedinka loša, ne mora značiti da je sav njen genetski kod loš. Zbog toga i lošije jedinke u nekoj mjeri moraju preživljavati.

Posebnost genetskog programiranja je u činjenici da jedinke u populaciji genetskog programa predstavljaju računalne programe, odnosno strukture koje se mogu jednoznačno preslikati u oblik pogodan za izvođenje na računalu. Kako bi se olakšao proces stvaranja novih programa iz jednog ili više roditeljskih, programi su u većini ostvarenja genetskog programiranja napisani u obliku stabla. Novi se programi dobivaju uklanjanjem grana iz jednog stabla te dodavanjem tih grana na određeno mjesto u drugom stablu. Ovaj jednostavni proces kao rezultat također daje stablo i ujedno osigurava sintaktičku ispravnost dobivenih rješenja.

2.1. Elementi GP-a

Problem koji želimo riješiti genetskim programiranjem mora sadržavati određene elemente sustava. Prema (Koza 1992.), elementi su sljedeći:

- podatkovni i funkcijski elementi rješenja
- funkcija dobrote
- parametri genetskog programa
- uvjet zaustavljanja
- građa rješenja

Prikazujemo li rješenje pomoću stabla, razlikujemo dvije vrste čvorova stabla: funkcijski i podatkovni čvorovi. Funkcijski čvorovi sadrže funkcijske elemente, a podatkovni podatkovne elemente koji su završni čvorovi stabla, odnosno listovi. Funkcijski čvor također može biti završni čvor stabla, ali samo ako ima nula argumenata, jer tada mora biti završni čvor.

Parametri algoritma su veličina populacije, vjerojatnost primjene operatora i slično, dok uvjet zaustavljanja postupka određuje trenutak završetka evolucijskog procesa. Određivanje građe rješenja podrazumijeva definiranje eventualnih struktura u obliku potprograma koje glavni program može pozvati, koje Koza naziva automatski definiranim funkcijama (eng. *automatically defined functions*, ADF).

2.1.1. Podatkovni i funkcijski elementi

Odabir podatkovnih i funkcijskih elemenata mora biti takav da se njima može izraziti rješenje problema kojega rješavamo. Ovo svojstvo se naziva potpunost (eng.

sufficiency), a općenito ga nije moguće definirati jer ovisi o konkretnom problemu. Obično u skup podatkovnih i funkcijskih elemenata stavljamo sve one elemente za koje se smatra da su od koristi pri rješavanju zadanog problema. Još jedno svojstvo koje trebamo zadovoljiti jest svojstvo zatvorenosti (eng. *closure*). Svojstvo zatvorenosti Koza definira kao mogućnost svake funkcije iz skupa funkcijskih elemenata da, kao svoje argumente, prihvati bilo koju vrijednost i tip podatka koja može biti izračunata, odnosno evaluirana, od bilo koje funkcije u funkcijskom skupu i bilo koju vrijednost i tip podatka iz skupa podatkovnih elemenata. Dakle, skup podatkovnih i funkcijskih elemenata trebaju zadovoljavati svojstvo zatvorenosti tako da bilo koja kompozicija funkcija i podatkovnih elemenata (terminala) daje izvršiv računalni program. Uzmimo za primjer dijeljenje s nulom. Ako to dopustimo nećemo dobiti izvršiv računalni program, stoga se uzima zaštićeno dijeljenje s nulom koje vraća 1 ako je djelitelj nula.

2.1.2. Funkcija dobrote

Cijeli proces evolucije najviše ovisi o dobroti jedinki, odnosno funkciji koja je evaluira. Svaka jedinka populacije, odnosno svaki individualni računalni program izvršen je i evaluiran koristeći funkciju dobrote. Dobrota je obično određena nedosljednošću između rezultata dobivenog evaluacijom jedinke i željenog rezultata. Što je pogreška manja, program je bolji. Funkcija dobrote ne bi trebala nagrađivati samo najbolje jedinke (rješenja) u populaciji, već i sva poboljšanja pronađena tijekom evolucijskog procesa. Određivanje dobrote nekog rješenja sastoji se od ispitivanja ponašanja toga rješenja na nizu ispitnih primjera. Postoji jako puno različitih procjenitelja pogreške, a u ovom radu su implementirana tri različita načina računanja greške:

- MAPE – *Mean Absolute Percentage Error*, često se koristi kao funkcija pogreške za probleme vremenskih nizova ili predviđanja, zadana je formulom:

$$MAPE = 100\% \cdot \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

- MSE – *Mean Square Error*, srednja kvadratna pogreška, često se koristi kao funkcija gubitka, zadana je formulom:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- MAE – *Mean Absolute Error*, srednja apsolutna pogreška, zadana je formulom:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

U formulama je *N* broj vrijednosti, y_i stvarna vrijednost, a $\hat{y}_i$ izračunata, predviđena vrijednost.

2.1.3. Parametri genetskog programa

Osnovni parametri genetskog programiranja su veličina populacije i maksimalni broj generacija. Osim ovih parametara proces genetskog programiranja uključuje i postavljanje vrijednosti mnogih drugih parametara. Odgovarajuće vrijednosti su ovisne o samoj primjeni i ne postoji neki općeniti skup vrijednosti parametara koji je dobar za svaki problem. U genetskom programiranju, za probleme koji nisu trivijalni, koriste se populacije od nekoliko tisuća ili čak milijuna jedinki koje evoluiraju desecima, stotinama ili tisućama generacija.

2.1.4. Uvjet zaustavljanja

Svako pokretanje genetskog programa zahtjeva specificiranje uvjeta zaustavljanja koji određuje kada ćemo prekinuti izvođenje i odrediti rezultat. Jedna od metoda za određivanje rezultata jest određivanje najbolje jedinke iz bilo koje generacije populacije u tijeku izvođenja (*best-so-far individual*) kao rezultata izvođenja. Najčešći oblik uvjeta zaustavljanja jest dostizanje prethodno definiranog broja generacija evolucijskog procesa. U svom radu (Koza 1992) John R. Koza za sve pokuse koristi broj od 50 generacija kao uvjet zaustavljanja izvođenja postupka, ukoliko prethodno nije pronađeno rješenje problema. Nakon tog broja generacija ne pronalaze se bitno različita rješenja, odnosno poboljšanja su vrlo mala. Dobro je imati dvojaku provjeru: algoritam se zaustavlja nakon zadanog broja generacija ili se zaustavlja nakon što u određenom broju generacija nije nađeno bolje rješenje, a to znači kraj učinkovite pretrage.

2.1.5. Građa rješenja

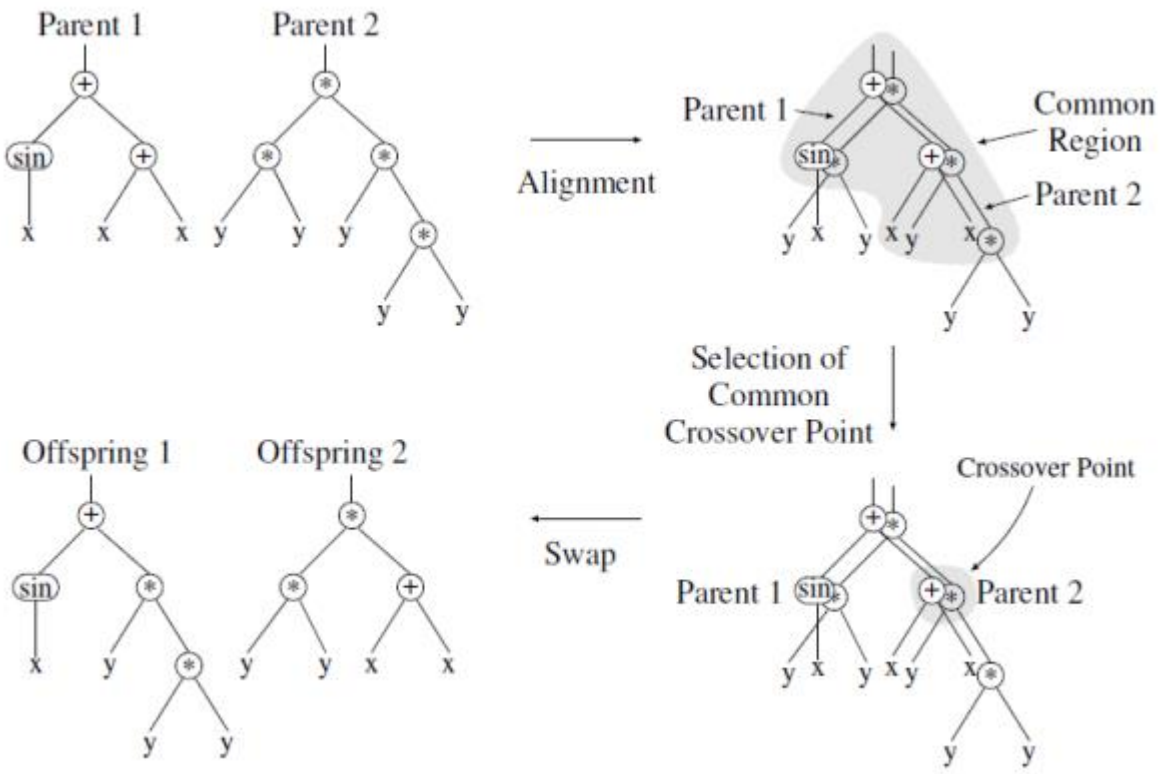
Ako koristimo prikaz rješenja pomoću stabla, građa rješenja se može sastojati od ADF funkcija (*automatically defined functions*). ADF funkcije su potprogrami, procedure koje dinamički evoluiraju tijekom izvođenja genetskog programa i mogu biti pozvane od strane glavnog programa koji također evoluira tijekom izvođenja genetskog programa. Koza u svom radu (Koza 1994) dokazuje da je pristup rješavanju pomoću ADF funkcija brži u pronalasku rješenja i bolji u rješavanju težih problema od običnog GP-a.

2.2. Osnovni operatori genetskog programiranja

Najbitniji dio genetskog programiranja su genetski operatori križanja i mutacije. Oni omogućavaju stvaranje novih jedinki iz postojećih te nasumičnu promjenu postojećih jedinki i time omogućavaju pretraživanje šireg prostora rješenja. Po Darwinovoj teoriji kod vrsta koje se seksualno razmnožavaju nema identičnih jedinki, već samo postoje varijacije iste te se najveći dio varijacija prenosi nasljeđivanjem. Iterativno iz trenutne generacije stvaramo sljedeću, a potomci imaju veću šansu stvoriti bolja rješenja. Operatorom križanja istražujemo okolinu roditelja dok nam operator mutacije omogućava da iskočimo iz lokalnih ekstrema, ako zaglavimo u njima.

2.2.1. Križanje

Operator križanja je operator koji uzima dvije jedinke, roditelje i vraća dvije nove jedinke, djecu. Spomenut ćemo dvije vrste križanja: križanje s jednom točkom prekida (engl. *One Point Crossover*) i uniformno križanje (engl. *Uniform Crossover*). Ono što je zajedničko operatorima je to da prije samog križanja traže zajedničko područje kod roditelja. U zajedničkom području roditelji imaju isti oblik, tj. u zajedničkom području se nalaze čvorovi koji imaju istu dubinu, isti broj djece i čiji roditelji su u zajedničkom području. Kod križanja s jednom točkom prekida, nalazi se jedna veza iz zajedničkog područja te se ta veza zamijeni kod roditelja. Kod uniformnog križanja, za svaku vezu koja se nalazi u zajedničkom području se bira hoće li ostati ili će se zamijeniti među roditeljima. Primjer određivanja zajedničkog područja i primjene križanja s jednom točkom prekida je na slici 2.1. Prednost ovih križanja je da dubina djece koja se stvara nikada nije veća od dubine roditelja, a nedostatak je da se križanje uvijek odvija na manjim dubinama stabla, bliže korijenu.



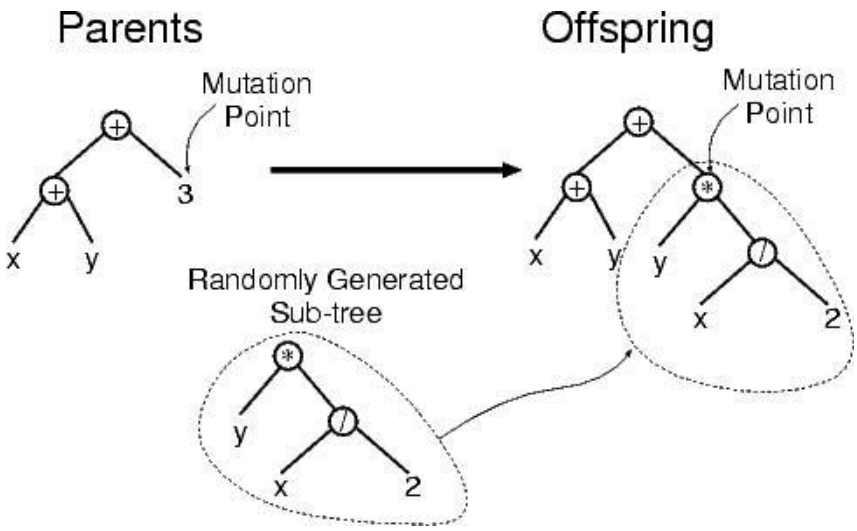
Slika 2.2.1 Primjer križanja s jednom točkom prekida (preuzeto s [6])

2.2.2. Mutacija

Mutacija služi kako bi se u nove jedinke uveli neki geni koji će rezultirati jedinkama s boljim rješenjem. Općenito gledajući, mutacijama se može dobiti poboljšanje, ali i pogoršanje prosječne dobrote jedinke u populaciji. Jedinke sa smrtonosnim mutacijama bit će vrlo brzo eliminirane u selekcijskom procesu, tako da iste ne bi smjele dugotrajno narušiti prosječnu dobrotu jedinki u procesu.

Mutacija se za jedinke GP-a najčešće radi na sljedeći način: iz populacije se odabere mutirajuća jedinka te se u stablu jedinke nasumce odabere jedan čvor. Slučajno je moguće odabrati i čvor i list iz istog stabla. Odabrani čvor (list) i sva njegova podstabla brišu se iz stabla. Na odabranome mjestu se zatim slučajnim

odabirom stvara novo podstablo. Veličina i način stvaranja novog podstabla obično se definiraju prilikom pokretanja algoritma.



Slika 2.2.2 Primjer mutacije (preuzeto s [6])

2.2.3. Izbor roditelja

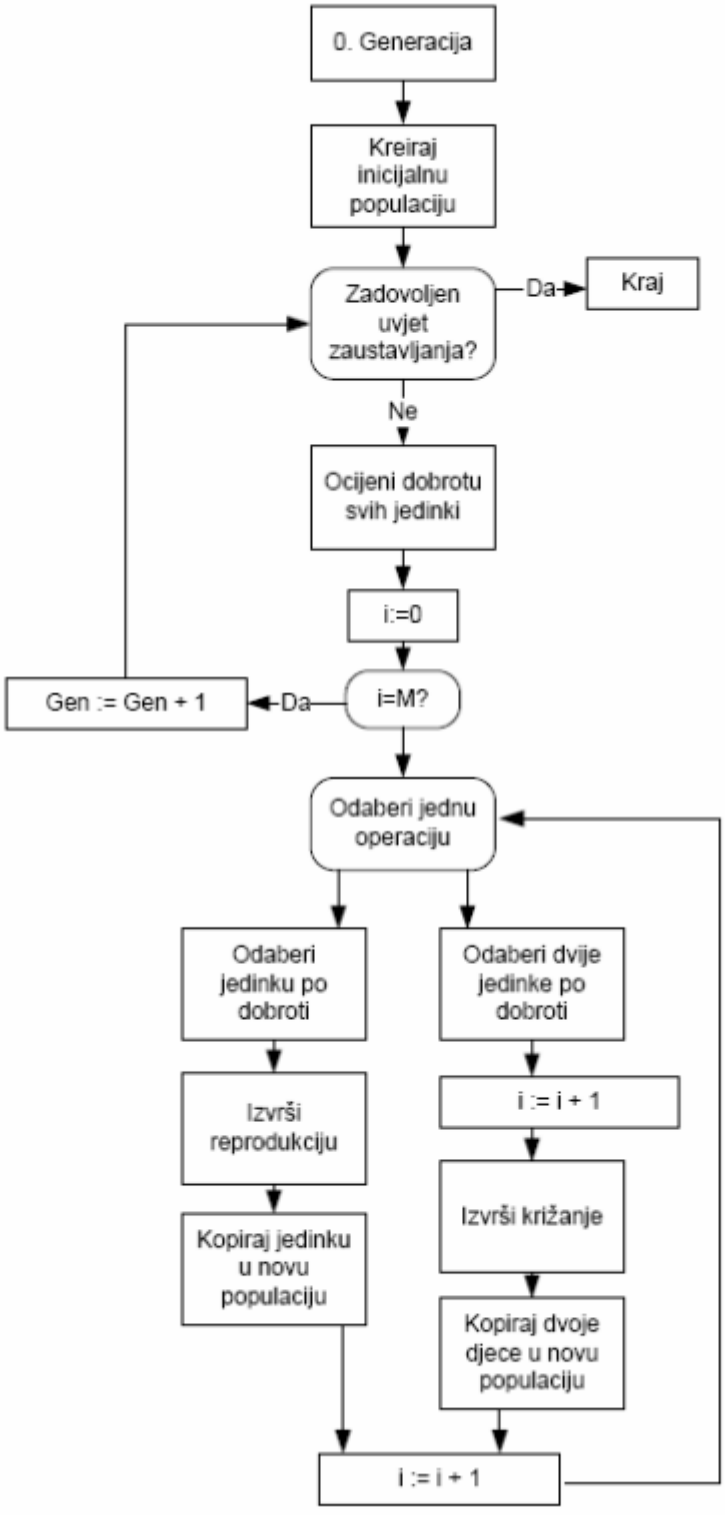
Kod izbora roditelja možemo primijeniti više tehnika. Tehnika proporcionalne selekcije omogućuje jedinkama da uđu u proces križanja, ali što je jedinka bolja ima veću šansu biti izabrana. Ovo možemo prikazati formulom:

$$selProbability(i) = \frac{fitness(i)}{\sum_{j=1}^{n} fitness(j)}$$

Ovu tehniku možemo prikazati kao jedan veliki kotač ruleta koji se okreće. Polja ruleta predstavljaju jedinke populacije, a proporcionalna veličini vjerojatnosti jedinke.

2.3. Algoritam GP-a

Algoritam genetskog programiranja započinje stvaranjem slučajne populacije temeljem funkcijskih i podatkovnih skupova. Veličina svakog slučajno stvorenog programa treba biti ograničena iz praktičnih razloga, npr. za stablo je to minimalna i maksimalna dubina stabla. Za svaki stvoreni program evaluiramo njegovu dobrotu pomoću funkcije dobrote. Odaberemo jedinke iz skupa koje će nam predstavljati roditelje nove populacije i nad njima provodimo operacije križanja i mutacije. Nakon stvaranja nove populacije algoritam ponovno bira skup jedinki koje će predstavljati roditelje i nad njima obavlja operacije križanja i mutacije. Algoritam staje kada se ispuni neki od uvjeta zaustavljanja algoritma. Selekciju možemo vršiti pomoću *roulette-wheel* algoritma, turnirskog algoritma i slično. Turnirski algoritam provodi selekciju nad k jedinki s veličinom turnira k. Pobjednik turnira je jedinka sa najboljom dobrotom i ona se sprema u tzv. *mating pool*, odnosno odabire se za roditelja sljedeće generacije. U ovom radu korištena je druga inačica k-turnirskog odabira. K-turnirski algoritam radi na način da odaberemo k jedinki iz postojeće populacije i za svaku odabranu jedinku izračunamo njenu dobrotu. Jedinka sa najmanjom dobrotom se zamjenjuje sa novom jedinkom nastalom od križanja dvije jedinke koje su preživjele turnir. Primijetimo da k-turnirski algoritam podržava elitizam, odnosno osigurava da najbolje jedinke prežive. Kako odabiremo k jedinki, a zamjenjujemo samo najlošiju, ostalih k-1 jedinki ostaje nepromijenjeno.



Slika 2.3. Algoritam optimiranja GP-om

Algoritam Općeniti pseudokod genetskog algoritma

```
brojGeneracija ← 0
populacija ← generirajPocetnuPopulaciju()
while uvjetZaustavljanjaNijeIspunjen() do
    for i = 1 → brojJedinki do
        jedinka_i.dobrota ← izracunajDobrotu()
    end for
    populacija ← provediSelekciju(populacija)
    provediKrizanje(populacija)
    provediMutaciju(populacija)
    brojGeneracija = brojGeneracija + 1
end while
```

3. Programski sustav za predviđanje

U okviru ovog rada implementiran je programski sustav za predviđanje uporabom genetskog programiranja, odnosno izgradnjom stabla koje na osnovu definiranih ulaznih podataka izračunava iduću vrijednost vremenskog slijeda. Programska implementacija rješenja pisana je u programskom jeziku *Java,* a sama je struktura inspirirana onom koju koristi *Evolutionary Computation Framework (ECF)* od kojeg je preuzeta ideja razvoja okruženja koje omogućava osnovne GP algoritme te koje je lako proširivo dodatnim operatorima i funkcijskim čvorovima po potrebi, uz mogućnost lake konfiguracije kroz jedinstvenu konfiguracijsku datoteku. Primjer jedne takve datoteke prikazan je u nastavku:

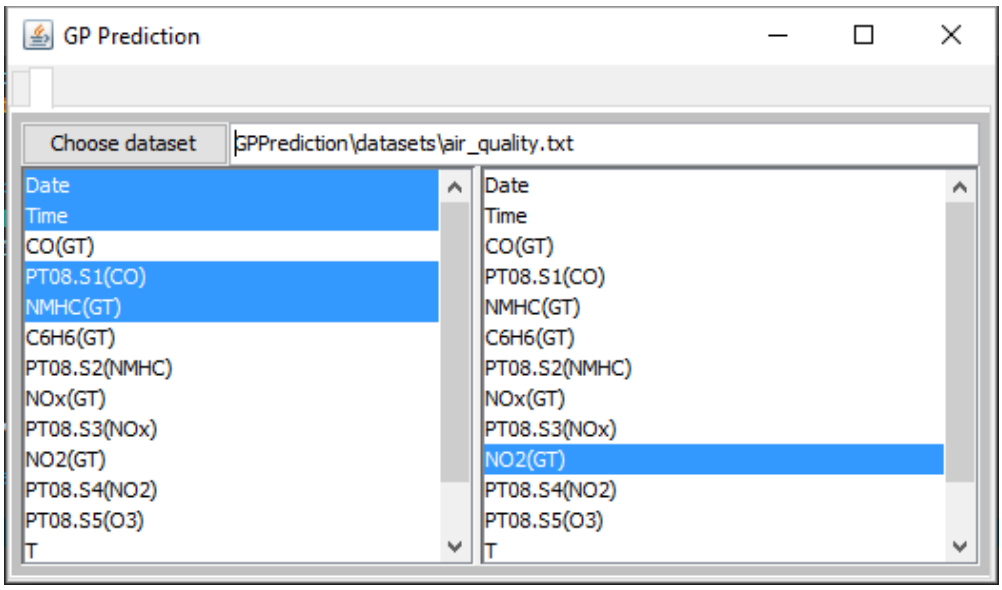
```
<ECF>
        <Algorithm>
                <SteadyStateTournament>
                        <Entry key="tsize">3</Entry>
                </SteadyStateTournament>
        </Algorithm>
        <Genotype>
                <Tree>
                        <Entry key="maxdepth">5</Entry>
                        <Entry key="mindepth">2</Entry>
                        <Entry key="functionset">+ - * /</Entry>
                        <Entry key="terminalset">X Y [0 1]</Entry>

                        <Entry key="mut.nodereplace">0.5</Entry>
                        <Entry key="crx.uniform">0.5</Entry>
                        <Entry key="mut.permutation">0.5</Entry>
                        <Entry key="crx.simple">0.5</Entry>
                        <Entry key="functionprob">0.2</Entry>
                </Tree>
        </Genotype>
        <Registry>
                <Entry key="population.size">1000</Entry>
                <Entry key="population.demes">1</Entry>
                <Entry key="mutation.indprob">0.5</Entry>
                <Entry key="mutation.genotypes">random</Entry>
                <Entry key="crossover.genotypes">random</Entry>
                <Entry key="term.maxgen">100</Entry>
                <Entry key="log.level">4</Entry>
                <Entry key="log.filename">log.txt</Entry>
                <Entry key="log.frequency">1</Entry>
        </Registry>
</ECF>
```

Primjer konfiguracijske datoteke za ECF

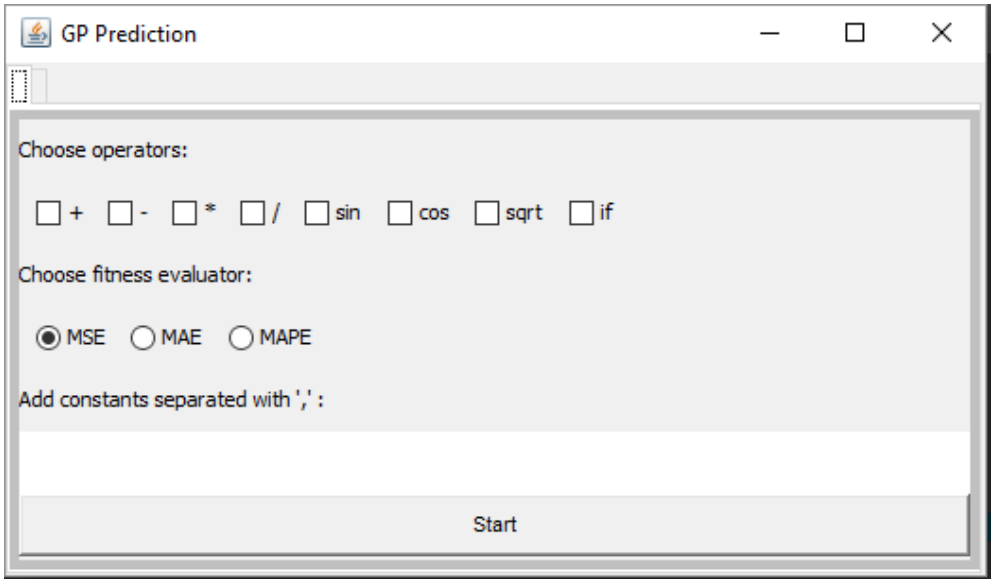
Na slici 3.1 prikazan je programski sustav prilikom pokretanja. Grafičko sučelje sastoji se od dviju kartica. Na prvoj kartici pritiskom gumba „Choose dataset“ odabire se datoteka koja sadrži skup podataka na kojima se pokreće učenje. Na lijevoj listi odabiru se ulazne varijable koje će biti korištene prilikom izgradnje stabla, a na

desnoj listi odabire se vrijednost koja se želi predvidjeti na temelju odabranih ulaznih varijabli.



Slika 3.1 Prikaz izvođenja programa

Na slici 3.2 prikazana je druga kartica koja korisniku omogućava odabir jednog ili više operatora koji će se koristiti prilikom učenja. U sljedećem redu odabire se način na koji se evaluira dobrota svake jedinke. Podržana su 3 evaulatora dobrote opisana u prethodnim poglavljima. Korisnik još može zadati niz intervala iz kojih će se uzimati konstante prilikom izgradnje stabla te pritiskom na gumb „Start“ pokreće se učenje te se u konzoli dobije ispis najbolje jedinke.



Slika 3.2 Prikaz izvođenja programa

3.1. Ulazni podaci

Za provjeru rezultata implementiranog sustava korišteni su javno dostupni skupovi podataka. Svaki od skupova podataka podijeljen je na skup za učenje na kojem se genetskim programiranjem evoulira stablo te skup za testiranje nad kojim se evoluirani model evaluira. Skup za učenje činio je 80% podataka, a skup za ispitivanje 20% podataka.

3.1.1. Osnovni skup podataka

U svrhu testiranja programskog sustava pripremljen je osnovni skup podataka koji je odgovarao simboličkog regresiji funkcije jedne varijable. Jednostavna linearna funkcija koja se koristila za generiranje ulaznih i za provjeru izlaznih vrijednosti jest

f(x) = 2x - 3. Domena je podijeljena na cjelobrojne vrijednosti u rasponu od 0 do 1000 koje su predstavljale određene trenutke u vremenu.

3.1.2. Air quality

Skup podataka preuzet s [5] koji sadrži 9358 podataka izmjerenih pomoću 5 senzora ugrađenih u uređaj za mjerenje kakvoće zraka. Uređaj je bio smješten na otvorenom u relativno zagađenom području, unutar jednog talijanskog grada. Podaci su mjereni svakog sata u razdoblju od ožujka 2004. do veljače 2015. U nastavku je prikazana tablica s kratkim opisima mjerenih varijabli.

Varijabla	**Tip varijable**	**Kratki opis**
Vrijeme	Cjelobrojna	Vrijeme u sekundama
Dan u mjesecu	Cjelobrojna	Redni broj dana u mjesecu
Mesec u godini	Cjelobrojna	1 za siječanj, 2 za veljaču, itd.
CO	Realna	Koncentracija CO u zraku (u mg/m^3)
NOx	Realna	Koncentracija NOx u zraku (u ppb)
NO2	Realna	Koncentracija NO2 u zraku (u mg/m^3)
T	Realna	Izmjerena temperatura zraka
RH	Realna	Relativna vlažnost zraka (u %)

3.1.3. Household power consumption

Skup podataka također preuzet s [5] koji sadrži 2 075 259 mjerenja skupljenih u razdoblju od prosinca 2006. do studenog 2010. Vrijednosti su mjerene s razmakom od jedne minute, a predstavljaju potrošnju električne energije jednog kućanstva. U nastavku je prikazana tablica s opisima mjerenih varijabli.

Varijabla	**Tip varijable**	**Kratki opis**
Vrijeme	Cjelobrojna	Vrijeme u sekundama
Dan u mjesecu	Cjelobrojna	Redni broj dana u mjesecu
Mjesec u godini	Cjelobrojna	1 za siječanj, 2 za veljaču, itd.
Global_active_power	Realna	Radna snaga (u KW)
Global_reactive_power	Realna	Jalova snaga (u KW)
Voltage	Realna	Napon (u V)
Global_intensity	Realna	Struja (u A)

4. Rezultati

U ovom poglavlju bit će opisana sva mjerenja koja su napravljena kako bi se poboljšao rad GP-a.

Za pravilan odabir parametara trebalo bi provesti testiranje sa svim mogućim podskupovima prethodno definiranog skupa korištenih parametara, ali u implementaciji postoji previše parametara za takav pristup. Stoga će se pokušati pronaći optimalni parametar po parametar. Takav pristup jednako je kompliciran jer su parametri zavisni te se mijenjanjem jednog parametra može smanjiti dobar utjecaj vrijednosti nekog drugog parametra.

4.1. Određivanje parametara GP-a

U nastavku će biti dan pregled mjerenja i određivanja parametara za genetsko programiranje. Za parametre će se isprobati nekoliko različitih vrijednosti kako bi se utvrdilo ima li utjecaj i koliki te će biti navedeno na kojem se skupu ili na kojim skupovima podataka je ispitivanje bilo provedeno.

4.1.1. Uvjet zaustavljanja

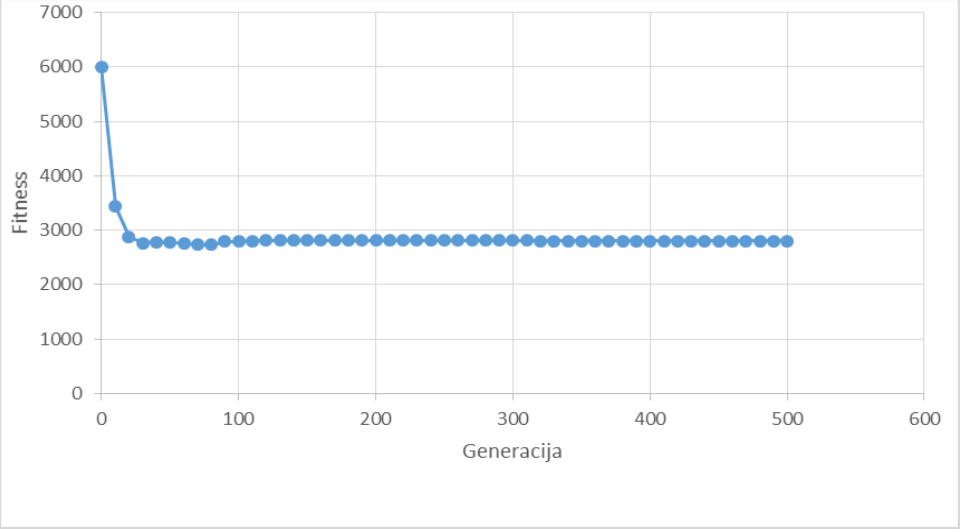
Radi ubrzanja algoritma, napravljeno je mjerenje za najveći broj generacija koji algoritam odradi. Što učenje dulje traje, odnosno što je broj generacija veći to će algoritam davati sve bolja rješenja sve dok algoritam ne počne konvergirati.

Uvjet zaustavljanja ispitan je na dva skupa podataka opisanim u prethodnim poglavljima te je određen grafički, kako je prikazano na slici 4.1 i 4.2. Nakon što se

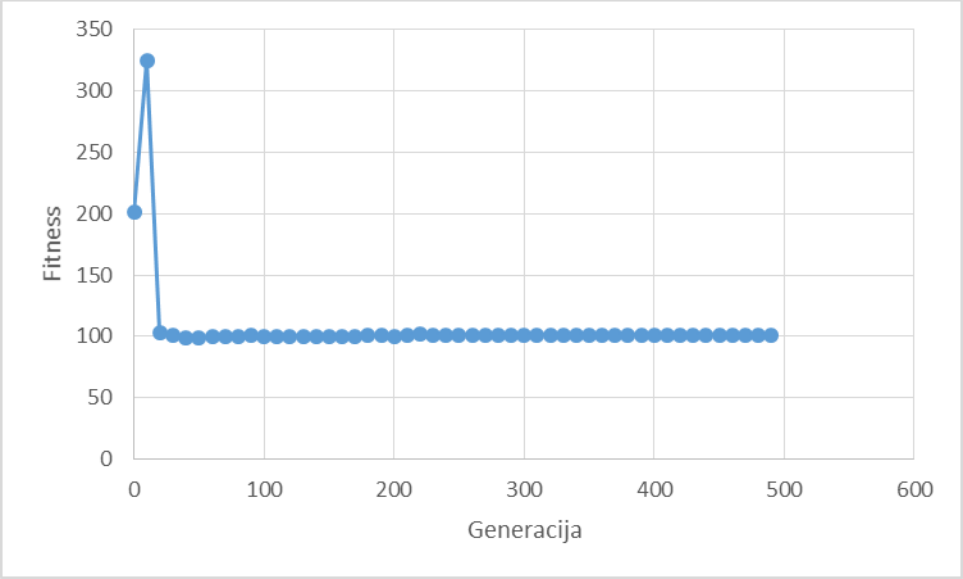
prediktor nauči na skupu za učenje, njegova greška se provjerava na skupu za provjeru. Taj postupak se ponavlja nakon svakih 10 izvršenih generacija. Očito je kako nakon 100-te generacije nema značajnijeg pomaka te je upravo taj broj uzet kao uvjet zaustavljanja u daljnim ispitivanjima.

Parametri koji su korišteni prilikom ispitivanja uvjeta zaustavljanja na prethodno opisanim skupovima podataka:

- broj generacija: 500
- veličina populacije: 4000
- procjenitelj pogreške: MAE
- funkcijski čvorovi: +, -, *, /



Slika 4.1 Graf ovisnosti dobrote o trenutnoj generaciji za skup podataka „Air quailty“



Slika 4.2 Graf ovisnosti dobrote o trenutnoj generaciji za skup podataka „Household power consumption“

4.1.2. Veličina populacije

Veličina populacije jest jedan od osnovnih i značajnijih parametara. U pravilu, veličina populacije se postavlja ili na jako malu vrijednost (između 10 i 20 jedinki) ili na jako veliku vrijednost (preko 100, nekad i 1000 jedinki). Ukoliko je populacija manja tada se odrađuje veći broj generacija (>100), inače, ukoliko je populacija veća tada se broj generacija postavlja na manji broj (<50). U oba slučaja napravi se približno jednak broj iteracija. Značajnija razlika jest u tome što kod veće populacije postoji veća raznolikost jer se teže eliminiraju sve lošije jedinke.

Veličina populacije ispitana je na skupu podataka „Air quality“ te parametri genetskog programiranja koji su korišteni prilikom ispitivanja su:

- broj generacija: 100
- procjenitelj greške: MAE
- funkcijski čvorovi: +, -, *, /

Br.jed.	25	50	100	250	500	750	1000	2000
Max	2346.08	3223.11	1209.69	610.07	526.29	563.82	553.44	586.18
Min	758.72	672.21	666.43	477.15	387.54	346.36	392.12	340.07
Best	487.01	417.76	375.44	355.77	225.47	244.53	282.68	243.04

U prethodnoj tablici prikazani su rezultati dobiveni ispitivanjem parametra veličine populacije s ostalim inicijalnim parametrima. Za svaku veličinu populacije navedenu u tablici pokrenuto je učenje na podacima za učenje u 4 ponavljanja te je zatim najbolja jedinka evaluirana na podacima za testiranje. U tablici „Max“ vrijednost označava najveću vrijednost dobrote koju je neka jedinka imala prilikom učenja, „Min“ označava najmanju vrijednost te „Best“ označava vrijednost funkcije dobrote kada je najbolja jedinka, ona koja ima minimalnu vrijednosti, evaluirana nad skupom podataka za testiranje.

Vidi se kako je prijelomna os negdje oko 500 – 750 jedinki. Mjerenja provedena s populacijom od 250 ili manje jedinki pokazuju relativno lošije rezultate, a ostala mjerenja s većim brojem jedinki pokazuju da nema veće razlike između njih. Stoga je u daljnim mjerenjima za veličinu populacije odabrano 500 jedinki radi ubrzanja samog algoritma te jer ima najbolji rezultat kada se najbolja jedinka evaluirala na skupu podataka za testiranje.

4.1.3. Funkcijski čvorovi

Funkcijski su čvorovi veoma bitni jer pomoću njih aproksimiramo željene izlazne vrijednosti. Ukoliko postoji premalo „dobrih“ funkcijskih čvorova, algoritam se neće moći točno prilagoditi podacima, a s umetanjem više funkcijskih čvorova povećavamo mogućnost da se algoritam prenauči. Pojam „dobar“ funkcijski čvor jest

subjektivan, koliko je neki funkcijski čvor koristan za GP ovisi isključivo o problemu za koji se GP implementira. Isti funkcijski čvor za neke probleme može biti jako koristan, dok za druge može imati negativan utjecaj na algoritam.

Ispitivanje je odrađeno nad oba skupa podataka opisanim u prethodnim poglavljima te parametri genetskog programiranja koji su korišteni prilikom ispitivanja su:

- broj generacija: 100
- procjenitelj greške: MAE
- veličina populacije: 500
- osnovni skup čvorova: +, -, *, /

Dodatni skup čvorova	**{}**	**{sin, cos}**	**{sqrt}**	**{if}**
Max	296.87	164.56	67.04	268.21
Min	18.34	5.81	24.45	19.23
Best	32.17	18.61	23.99	27.91

Prethodna tablica prikazuje rezultate dobivene ispitivanjem različitih funkcijskih čvorova nad skupom podatka “Household power consumption”. Redci u tablici isti su kao i u prethodnim mjerenjima, “Max” predstavlja vrijednost najlošije jedinke, “Min” predstavlja vrijednost najbolje jedinke nad podacima za učenje te “Best” predstavlja vrijednost najbolje jedinke evaluirane nad podacima za testiranje.

Možemo primijetiti kako za ovaj skup podataka funkcije sinus i kosinus imaju pozitivan utjecaj s obzirom na osnovni skup funkcijskih čvorova, dok funkcijski čvorovi uvjetnog grananja te korijena nemaju značajniji utjecaj.

Dodatni skup čvorova	**{}**	**{sin, cos}**	**{sqrt}**	**{if}**
Max	792.25	607.82	946.59	609.81
Min	418.17	552.92	469.97	283.18
Best	292.16	573.84	411.09	214.59

Prethodna tablica prikazuje rezultate dobivene ispitivanjem različitih funkcijskih čvorova nad skupom podatka “Air quality”.

Za ovaj skup podataka vidimo kako jedino funkcijski čvor uvjetnog grananja ima pozitivni utjecaj, dok funkcije sinus i kosinus imaju značajno negativan utjecaj s obzirom na osnovni skup funkcijskih čvorova.

Iz cijele ove analize funkcijskih čvorova očito je kako generalno “pravi” skup ne postoji, nego on ovisi o konkretnom problemu i veličini koja se predviđa. Stoga je potrebna pažljiva analiza prilikom odabira čvorova prije pokretanja učenja nad nekim skupom podataka.

5. Zaključak

Predviđanje je jako unosno i korisno područje, pogotovo kada je riječ predviđanju cijena dionica ili sportskih rezultata, stoga je u sklopu ovog rada ostvaren programski sustav za predviđanje vremenskih nizova uporabom genetskog programiranja.

Opisane su i implementirane metode genetskog programiranja na dva relativno zahtjevna problema predviđanja vrijednosti vremenskih nizova, na primjeru predviđanja kakvoće zraka te na primjeru predviđanja potrošnje električne energije. Ispitivani su i mjereni neki od parametara GP-a kako bi se probale naći optimalne vrijednosti. Ispitivanja koja su provedena nisu detaljna i ne idu u dubinu problema, već su osnovna radi inicijalnog određivanja vrijednosti parametara.

Rezultati dobiveni primjenom genetskog programiranja zadovoljavajući su i usporedivi s postojećim rezultatima. Također, očito je kako pravilan izbor ulaznih parametra jako utječe na same rezultate. Stoga je pažljiva analiza domene problema iznimno važna prilikom predviđanja algoritmom genetskog programiranja.

Dodatno, programska implementacija napravljena za potrebe rada iznimno je modularna te su izmjene i daljne nadogradnje lako moguće. Također, primjena postojećih algoritama na druge probleme predviđanja vremenskih nizova jest jako jednostavna, sve što je potrebno jest pripremiti skup podataka u prethodno formatiranoj datoteci.

6. Literatura

[1] John R. Koza GENETIC PROGRAMMING: A PARADIGM FOR GENETICALLY BREEDING POPULATIONS OF COMPUTER PROGRAMS TO SOLVE PROBLEMS, Computer Science Department, Stanford University, Margaret Jacks, Hall Stanford, CA 94305, June 1990

[2] Koza, J. R. Genetic Programming – On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.

[3] Marko Deak. Kratkoročno predviđanje potrošnje električne energije. 2015.

[4] Mirela Ćosić. Kratkoročna prognoza potrošnje električne energije. 2014.

[5] Machine Learning Repository, Center for Machine Learning and Intelligent Systems, https://archive.ics.uci.edu/ml/datasets/, svibanj 2015.

[6] GP Field Guide, http://cswww.essex.ac.uk/staff/poli/gp-field-guide

Simbolička regresija vremenskih nizova uporabom genetskog programiranja

Sažetak: Problem predviđanja vremenskih nizova jedan je vrlo unosnih i korisnih područja te ima značajnu ulogu u različitim industrijama. Svrha ovog rada jest demonstrirati primjenu metoda genetskog programiranja na različitim problemima čije se vrijednosti mogu predstaviti kao vremenski nizovi. Ostvaren je programski sustav koji se temelji na osnovnoj implementaciji genetskog programiranja te su pomoću njega ispitani utjecaji različitih parametara genetskog programiranja. Konačno, dana su razmatranja dobivenih rezultata te mogućnosti nadogradnje programskog sustava.

Ključne riječi: genetsko programiranje, simbolička regresija, predviđanje, programski sustav

Symbolic regression based forecasting with genetic programming

Abstract: Time series forecasting is very profitable and valuable area and has an important role in different industries. The goal of this paper is to demonstrate the application of genetic programming methods on the various problems which can be presented as time series. Within this paper software system which is based on basic implementation of genetic programming is realized and which is used to determine impacts of different genetic programming parameters. Finally, further consideration of results is given as well some possible software improvements.

Keywords: genetic programming, symbolic regression, forecasting, software system