

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 654

**Kratkoročna prognoza potrošnje
električne energije**

Mirela Ćosić

Zagreb, lipanj 2014.

Zagreb, 13. ožujka 2014.

DIPLOMSKI ZADATAK br. 654

Pristupnik: **Mirela Ćosić**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Kratkoročna prognoza potrošnje električne energije**

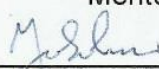
Opis zadatka:

Opisati problem predviđanja potrošnje električne energije s ciljem učinkovitog upravljanja elektroenergetskim sustavom. Posebnu pažnju posvetiti uvjetima kratkoročnog predviđanja u okruženju manjih urbanih sredina s obnovljivim izvorima energije. Izraditi okruženje za ispitivanje proizvoljnog postupka strojnog učenja za predviđanje opterećenja uz uobičajene mjere kvalitete. Ostvariti postupak predviđanja temeljen na genetskom programiranju. Ostvariti mogućnost predviđanja korištenjem skupa modela i mehanizma glasovanja. Usporediti učinkovitost ostvarenih postupaka s postojećim rješenjima. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

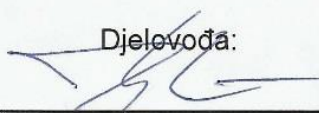
Zadatak uručen pristupniku: 14. ožujka 2014.

Rok za predaju rada: 30. lipnja 2014.

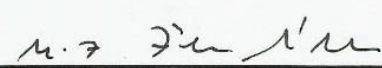
Mentor:


Izv.prof.dr.sc. Domagoj Jakobović

Djelovoda:


Doc.dr.sc. Tomislav Hrkać

Predsjednik odbora za
diplomski rad profila:


Prof.dr.sc. Siniša Srbljić

Sadržaj

Popis tablica.....	vi
Popis slika	vii
Uvod.....	1
1. Kratkoročno predviđanje potrošnje električne energije	2
1.1. Faktori koji utječu na predviđanje	3
1.2. Distribuirani sustavi	3
2. Problem predviđanja	5
2.1. Ulazne varijable	5
2.2. Izlazne varijable i horizont predviđanja.....	8
3. Opis čestih rješenja za predviđanje potrošnje	9
3.1. Neuronske mreže	9
3.2. Ansambl neuronskih mreža	11
3.3. Box - Jenkins metoda ili ARIMA model.....	12
3.4. Metoda potpornih vektora	13
4. Genetsko programiranje.....	14
4.1. Detalji implementacije GP-a	16
4.1.1. Populacija, jedinka, stablo i čvorovi	16
4.1.2. Algoritam	16
4.1.3. Selekcija	17
4.1.4. Mutacija	17
4.1.5. Križanje	18
4.1.6. Evaluacija jedinke	19
4.1.7. Uvjeti zaustavljanja.....	19
4.2. Određivanje vrijednosti parametara	20
4.3. Ulazni podaci	22
4.4. Pregled postojećih implementacija genetskog programiranja za predviđanje ..	23

4.5.	Genetsko programiranje kao prediktor potrošnje	25
5.	Mjerenja	26
5.1.	Određivanje parametara GP-a.....	28
5.1.1.	Veličina populacije.....	29
5.1.2.	Evolucijski algoritam	31
5.1.3.	Veličina turnira	34
5.1.4.	Križanje	35
5.1.5.	Mutacija	36
5.1.6.	Uvjet zaustavljanja.....	38
5.1.7.	Funkcijski čvorovi.....	41
5.1.8.	Dubina stabla	44
5.2.	Određivanje ulaznih vrijednosti	47
5.2.1.	Potrošnje u prijašnjim danima za isti sat.....	48
5.2.2.	Potrošnje iz prijašnjih 24 sata.....	49
5.2.3.	Meteorološke prilike	50
5.3.	Usporedba s drugim algoritmima.....	51
5.3.1.	Podaci za FER	51
5.3.2.	ISO – New England podaci.....	53
	Zaključak	55
	Literatura	56
	Sažetak.....	57
	Abstract	58

Popis tablica

Tablica 1 Prikaz implementiranih operatora i algoritama	21
Tablica 2 Prikaz mogućih ulaznih podataka	23
Tablica 3 Inicijalni parametri i ulazni podaci za GP	29
Tablica 4 Rezultati za različite vrijednosti veličine populacije	30
Tablica 5 Rezultati za izbor algoritma	32
Tablica 6 Iznos normiranih pogreški za izbor veličine populacije	34
Tablica 7 Prikaz normirane greške za operatore križanja	36
Tablica 8 Poboljšani inicijalni parametri s unakrsnom provjerom	38
Tablica 9 Rezultati na skupu za učenje kroz generacije	40
Tablica 10 Prikaz parametara za učenje GP-a kod određivanja ulaznih varijabli	47
Tablica 11 Usporedba prediktora na podacima za FER	52
Tablica 12 Rezultati po mjesecima za GP prediktor na podacima za FER	53
Tablica 13 Usporedba prediktora na ISO-NE 2006 podacima	54

Popis slika

Slika 1 Primjer jednostavne neuronske mreže s jednim skrivenim slojem	9
Slika 2 Osnovni princip metode potpornih vektora	12
Slika 3 Primjer regresije potpornim vektorima	13
Slika 4 Grafički prikaz rada algoritma genetskog programiranja	15
Slika 5 Primjer križanja s jednom točkom prekida [7]	18
Slika 6 Određivanje optimalnog modela metodom unakrsne provjere	27
Slika 7 Primjer <i>boxplot</i> prikaza podataka	28
Slika 8 Prikaz rezultata za veličinu populacije, bez odstupajućih vrijednosti.....	30
Slika 9 Prikaz rezultata za veličinu populacije, s odstupajućim vrijednostima.....	31
Slika 10 Prikaz rezultata za izbor algoritma, bez odstupajućih vrijednosti	33
Slika 11 Prikaz rezultata za izbor algoritma, s odstupajućim vrijednostima	33
Slika 12 Prikaz rezultata za veličine turnira, bez odstupajućih vrijednosti	34
Slika 13 Prikaz rezultata za izbor operatora križanja, s i bez odstupajućih vrijednosti	35
Slika 14 Prikaz rezultata za vjerojatnost mutacije za <i>hoist</i> mutaciju.....	37
Slika 15 Prikaz rezultata za vjerojatnost mutacije za <i>point</i> mutaciju	37
Slika 16 Greška na skupu za provjeru kroz generacije, bez odstupajućih vrijednosti	39
Slika 17 Greška na skupu za provjeru kroz generacije, s odstupajućim vrijednostima	41
Slika 18 Prikaz rezultata za nove funkcijske čvorove, bez odstupajućih vrijednosti	42
Slika 19 Prikaz rezultata za parove funkcijskih čvorova, bez odstupajućih vrijednosti.....	43
Slika 20 Prikaz rezultata za trojke funkcijskih čvorova, bez odstupajućih vrijednosti	44
Slika 21 Prikaz rezultata za najveću dubinu stabla, bez odstupajućih vrijednosti	45
Slika 22 Prikaz rezultata za najmanju dubinu stabla, bez odstupajućih vrijednosti.....	46
Slika 23 Prikaz rezultata pojedinačnih mjerenja za dubinu stabla	46
Slika 24 Prikaz rezultata za različiti broj mjerenja istog sata u prijašnjim danima.....	48
Slika 25 Prikaz rezultata za različite varijable iz prijašnjih 24 sata	49
Slika 26 Prikaz rezultata za različite meteorološke ulazne varijable	50

Uvod

Kratkoročno predviđanje potrošnje električne energije je bitno iz nekoliko osnovnih razloga: da se smanji gubitak, da se poveća efikasnost te da se sustav održava stabilnim. Predviđanje potrošnje električne energije je prisutno već neko vrijeme te se sve više ulaže u razvijanje tog područja. Danas postoje raznovrsni algoritmi koji su prihvaćeni kao uspješni prediktori i često se koriste kod određivanja uspješnosti novih algoritama. Predviđanje potrošnje je složen zadatak. Osnovno obilježje po kojima se prediktori razlikuju su za koji period predviđaju te u pravilu postoje dvije kategorije: kratkoročno predviđanje, period između jednog sata i tjedan dana, te dugoročno predviđanje, period može biti i do jedne godine unaprijed. Jedan od glavnih dijelova za dobro predviđanje je određivanje na temelju kojih podataka se predviđa. Neke opcije su: prijašnja potrošnja, kalendarski podaci, meteorološke prilike, itd. Danas se može naći jako puno radova na temu predviđanja potrošnje električne energije koji implementiraju algoritme iz širokog spektra područja. Cilj ovog rada je ukratko obraditi temu predviđanja potrošnje, pronaći preporuke i postojeća rješenja, kao i implementirati vlastiti prediktor temeljen na algoritmu genetskog programiranja. Do sada nije pronađen niti jedan rad koji spaja područje genetskog programiranja i kratkoročnog predviđanja potrošnje električne energije. Zbog toga u ovom radu se pridaje maksimalna pažnja podešavanju parametara genetskog algoritma, kao i izbor ulaznih podataka. Nakon toga slijedi usporedba vlastite implementacije s nekim postojećim algoritmima.

1. Kratkoročno predviđanje potrošnje električne energije

Kratkoročno predviđanje potrošnje električne energije (engl. *Short Term Load Forecasting*, STLF) je metoda kojom se predviđa potrošnja električne energije unutar nekog sustava za blisku budućnost. Kratkoročna predviđanja obično imaju horizont predviđanja (period za koji se predviđa) veličine jedan sat do tjedan dana.

Kratkoročno predviđanje opterećenja električne potrošnje dobiva sve veći značaj u zadnjih nekoliko desetljeća. Razlog tomu je činjenica da se proizvodnja električne energije decentralizirala, što znači da ima više ponuđača na tržištu, povećalo se korištenje alternativnih oblika energije zbog kojih se manje troši energija proizvođača ili se čak i prodaje proizvođaču. To znači da se u potrošnju energije unijelo još nekoliko dodatnih nepoznanica i da je sve teže procijeniti kolika bi mogla biti buduća potrošnja. Proizvođačima je informacija o budućoj potrošnji bitna jer potrošnja veća od proizvodnje znači prejak pad napona i probleme s održavanjem stabilnosti sustava. Dok sa suprotne strane, kada je potrošnja električne energije manja od proizvodnje iste, znači da se bez razloga generira energija te je proizvođač u gubitku jer ne prodaje proizvedenu energiju.

Predviđanje potrošnje električne energije znatno utječe na pouzdanost i stabilnost elektroenergetskog sustava. Zbog znanja o budućoj potrošnji, proizvođač može pravovremeno reagirati na predviđenu potrošnju te prilagoditi sustav budućoj potražnji. Tako može izbjeći probleme u radu sustava, padove napona ili generiranje električne energije koja se ne prodaje. Ukratko, smanjuje se ukupna cijena održavanja sustava.

Također, kratkoročno predviđanje električne energije može biti još korisnije u sustavima u kojima postoji „reagiranje na potražnju“ (engl. *Demand Response*). Reagiranje na potražnju je sustav u kojemu krajnji korisnik prima informacije od elektroenergetskog sustava o trenutnom ili budućem stanju potražnje i cijene električne energije te tako može prilagoditi svoju potrošnju.[5] Na primjer, neke potrošnje električne energije se mogu odgoditi za vrijeme u kojem je cijena niža, npr. pranje odjeće ili suđa. Tako potrošač može više ujednačiti potrošnju i smanjiti jaka odstupanja koja se pojavljuju u kritičnim satima potrošnje električne energije.

Naravno, da bi predviđanje potrošnje električne energije imalo jak i dobar utjecaj, predviđanja moraju biti „dobra“. Naime, zbog „loših“ predviđanja, proizvođač može biti u još većem gubitku nego što bi bio da predviđanja nije ni bilo.

1.1. Faktori koji utječu na predviđanje

Potražnja električne energije je nestacionaran proces na koji utječu brojni faktori, od vremenskih uvjeta preko sezonskih utjecaja do ekonomskih čimbenika zbog kojih je opterećenje vrlo teško predvidjeti. Također, faktori u pojedinim sustavima mogu značajno varirati. Neki od uobičajenih faktora koji mogu utjecati na potrošnju električne energije su vremenske prilike kao što su iznos temperature, količina vlage ili razina naoblake. Vanjske faktore općenito se može podijeliti u tri veće skupine ovisno o vrsti potrošača koji mogu biti stambeni, poslovni i industrijski.

Neka od obilježja potrošnje jesu periodičnost (ponavljanje) i sezonalnost (engl. Seasonality). Periodičnost znači da potrošnja u nekom satu može ovisiti o potrošnji u prijašnjem satu, prijašnjem danu te da postoji neki uzorak u potrošnji. Sezonalnost znači da može postojati nekoliko većih perioda, kao što su godišnja doba, unutar kojega se može primijetiti uzorak potrošnje električne energije. To također može značiti da se te velike sezone međusobno mogu razlikovati te da bi se onda i to trebalo uzeti kao faktor prilikom predviđanja potrošnje električne energije.

Izbor faktora kod predikcije potrošnje je iznimno bitan. Najbitnija je količina parametara jer premalo parametara može dati premalo informacija za uspješnu predikciju dok previše parametara može dati istu informaciju na različite načine čime se bespotrebno povećava dimenzija prostora pretraživanja i vrijeme izračuna buduće potrošnje.

1.2. Distribuirani sustavi

Zbog velike raznovrsnosti potražnje električne energije i zbog „skupog“ prijenosa električne energije na velike udaljenosti, u elektroenergetske sustave se sve više uvode manji distribuirani sustavi. [3] Svaki takav sustav ima svoje generatore električne energije i blizu je potrošačima. Na taj način, elektroenergetski sustav raspodjeljuje ukupnu potrošnju na

manje jedinice. Time se dobiva na dinamičnosti i prilagodljivosti sustava te se smanjuje udaljenost na koju se prenosi električna energija.

Kako se sve više razvijaju distribuirani sustavi, isto tako se počela razvijati i „distribuirana inteligencija“. Prednost distribuiranja logike na manje, lokalne generatore je ta da je lakše učiti manji sustav jer ima manje potrošača, a s time i manje vanjskih varijabli koji utječu na potrošnu. Tada postoji podjela posla pa svaka manja skupina odlučuje najbolje za sebe, a s time i za druge skupine. Tako se može rasteretiti centralni sustav koji onda ne mora donositi sve odluke.

Razlika između potrošnje velikom sustavu ili nekom njegovom manjem podsustavu može biti velika te učenje manjeg podsustava ne mora biti isto kao i učenje cijelog elektro-energetskog sustava. [3] Krivulje potrošnje manjih podsustava ne moraju pratiti iste trendove kao i krivulje potrošnje velikog sustava. Krivulje manjih podsustava često imaju veća odstupanja i izraženije „šiljke“.

2. Problem predviđanja

Uspješno predviđanje potrošnje električne energije zahtjeva napredne sustave učenja i poman odabir ulaza i parametara sustava. Problem kratkoročnog predviđanja električne energije je prisutan već dulji niz godina te postoje različiti prijedlozi kako napraviti (uspješan) prediktor, koji algoritam koristiti, koje ulazne varijable, itd. U ovom poglavlju će se algoritmu predviđanja pristupiti kao crnoj kutiji (engl. *Black Box*) te analizirati mogućnosti za ulazne i izlazne varijable.

2.1. Ulazne varijable

Ulazne varijable se većinom odnose na dan ili sat za koji se predviđa, a to je zapravo buduće vrijeme za koji neke varijable ne moraju biti poznate. Fiksne varijable, npr. kalendarske varijable, mogu se izračunati unaprijed, a varijabilne se mogu predvidjeti jednostavnim metodama kao što je linearna aproksimacija. Samo treba biti oprezan da se predviđeni podaci ne koriste previše u daljnjim predviđanjima radi mogućeg akumuliranja greške.

U svakoj implementaciji kratkoročnog predviđanja potrošnje je predložen drugačiji skup ulaznih varijabli. Sve ulazne varijable se mogu podijeliti u nekoliko većih skupina. U nastavku teksta će se dati pregled nekih predloženih ulaznih varijabli.

Prijašnja potrošnja

Prijašnja potrošnja je varijabla koja se sigurno mora nalaziti u skupu ulaznih varijabli, iako postoje iznimke [14], [15]. Ideja je da prediktor na osnovu prijašnjih potrošnji nauči trend potrošnje i zna kako će taj trend izgledati u bližoj budućnosti. Jedini je problem da nije trivijalno odrediti koliko prijašnje potrošnje treba za STLF. Isto tako pitanje je iz kojeg vremenskog okvira uzimati prijašnje potrošnje. U [1] su uzete vrijednosti proteklih 24 sata, vrijednost pred tjedan dana ($t-168$) i pred dva tjedna ($t-336$) u odnosu za odabrani sat, znači 26 ulaznih varijabli prijašnje potrošnje. U [2] (paralelni 24-satni model) uzete su vrijednosti u prošlih tjedan dana, ukupno 168 vrijednosti, od kojih se algoritmom odabira značajki (engl. *Feature Selection Algorithm*) biraju one koje se smatraju najbitnijima. U [3] analizom

je određeno da su ulazi opterećenje u prijašnjem danu i ukupna potrošnja u prijašnjem danu.

Temperatura

Temperatura se isto pokazala kao dobra ulazna varijabla, prvenstveno zbog toga što se s padom temperature povećava potrošnja energije zbog grijanja, a s povećanjem temperature se povećava potrošnja zbog hlađenja. Temperatura se može prediktoru dati u različitim oblicima: trenutna temperatura (temperatura u satu za koji se predviđa), temperatura u prijašnjem satu, prosječna temperatura u danu za koji se predviđa, itd. U [2] korištena su tri seta varijabli, za svaki skup uzete su vrijednosti temperature od prethodna tri dana (72 vrijednosti) te za dan za koji se predviđa (24 vrijednosti). Prvi skup određuje kolika je potreba za hlađenjem te je izražena s formulom (1), drugi skup određuje kolika je potreba za grijanjem te je izražena s formulom (2), a treći skup određuje kolika je potreba za dodatnim grijanjem te je izražena s formulom (3). Iz formula se vidi da su ulazne temperature u Celzijevim stupnjevima.

$$T_{hlađenja} = \max(0, T - 20 \text{ }^{\circ}\text{C}) \quad (1)$$

$$T_{grijanja} = \max(0, 16.5 \text{ }^{\circ}\text{C} - T) \quad (2)$$

$$T_{jakogGrijanja} = \max(0, 5 \text{ }^{\circ}\text{C} - T) \quad (3)$$

Sat u danu

Oznaka sata u danu je još jedna od čestih ulaznih varijabli zbog toga što je uočeno da je potrošnja periodična po danima, odnosno da je u istom satu u susjednim danima potrošnja jako slična. Također je bitna zbog toga što se u kućanstvima ili poslovnim zgradama potrošnja može jako povećati u određenim satima. Sat u danu je varijabla koja određuje koji je to sat za koji se predviđa i najčešće je cjelobrojna varijabla na intervalu [0, 23] ili [1, 24], ovisno o načinu gledanja. U [2] sat u danu je skup od 24 binarne varijable, od kojih je postavljena točno jedna varijabla na odgovarajućem mjestu.

Dan u tjednu

Oznaka dana u tjednu se isto često koristi da se razlikuju predviđanja za pojedine dane. Obično se prediktoru daje oznaka dana za koji se predviđa potrošnja. U [4] su pokazali da pojedini (radni) dani u tjednu imaju drugačiju krivulju dnevne potrošnje od drugih dana. Na taj se način može dodavanjem oznake dana pomoći prediktoru. U [2] korišteno je sedam binarnih varijabli od kojih je jedna istinita samo jedna na točno odgovarajućem mjestu. U [3] korištena su dvije varijable dane formulama (4) i (5). Razlog zbog kojeg su odabrani sinusi i kosinusi je taj što neuronska mreža dobro prihvaća periodične funkcije, a i sam dan u tjednu je periodičan.

$$D_1 = \sin \frac{2 \pi d}{7}, d \in [0, 6] \quad (4)$$

$$D_2 = \cos \frac{2 \pi d}{7}, d \in [0, 6] \quad (5)$$

Mjesec u godini

Oznaka mjeseca u godini se preporučuje koristiti zbog pojave sezonalnosti u podacima. Na taj način se omogućuje prediktoru da se prilagodi mjesecu u godini, odnosno pojedinoj sezoni. U [2] korišteno je dvanaest binarnih varijabli od kojih je istinita samo jedna i to na točno određenoj poziciji. U [3] su korištene dvije varijable koje predstavljaju mjesec u godini, slično kao i za oznaku dana u tjednu, te su te varijable prikazane formulama (6) i (7).

$$M_1 = \sin \frac{2 \pi m}{12}, m \in [0, 11] \quad (6)$$

$$M_2 = \cos \frac{2 \pi m}{12}, m \in [0, 11] \quad (7)$$

Radni dan

Oznaka radnog dana može doći kao zamjena ili dodatak varijabli za oznaku dana u tjednu. Jedina je razlika što oznaka radnog dana u sebi sadrži informaciju o praznicima i blagdanima te unaprijed sve subote i nedjelje proglašava neradnim danom. Oznaka radnog dana može biti bitna jer potrošnja radnim danom nikada nije ista kao neradnim. Kod kućanstva je potrošnja veća u neradnim danima, a u komercijalnim i industrijskim područjima je

potrošnja veća radnim danima. U [1] koristi se binarna varijabla koja je istinita ako je dan za koji se predviđa praznik. U [2] je korištena malo proširenija varijabla, odnosno korištene su četiri binarne varijable. Prva varijabla je istinita ako se predviđa za dan koji je praznik, druga je istinita ako je prethodni dan bio praznik, a treća i četvrta varijabla su istinite ako su dva dana iza dana za koji se predviđa praznici. U [3] oznaka radnog dana nije korištena jer nije pokazala značajnije rezultate, ali su zato koristili oznaku dana u tjednu.

Ostale varijable

Osim navedenih varijabli, preporučuju se koristiti varijable koje mogu opisivati stanje okoliša, kao što je osvijetljenost [1] jer se tako može procijeniti kako vanjski parametri utječu. Osim njih, mogu se koristiti i varijable koje opisuju aktivnost ljudi u području za koje se mjeri pa se tako, na primjer, može koristiti potrošnja tople vode [1] jer se s njom dobiva informacija o aktivnosti unutar zgrade. Izbor varijabli ovisi o njihovim prethodnim uspjesima, ali i o njihovoj dostupnosti na područjima za koja se predviđa.

2.2. Izlazne varijable i horizont predviđanja

Još jedan odabir koji utječe na performanse prediktora je broj njegovih izlaza. Broj izlaza je direktno povezan s odabirom što se želi predvidjeti i koliko unaprijed se želi predvidjeti. Prediktor ima samo jedan izlaz ako se predviđa samo za sljedeći sat, odnosno ako je horizont predviđanja veličine jednog sata. Ako se horizont predviđanja povećava, tada se povećava broj izlaza iz prediktora te to može utjecati na stupanj složenosti prediktora. Tome se može doskočiti tako da se jedan prediktor podijeli u više manjih prediktora, npr. tako da se ne predviđa za cijeli horizont od jednom, već za svaki sat zasebno. U tom slučaju se predviđena vrijednost u trenutnom satu može koristiti kao ulaz prilikom predikcije za sljedeći sat. Negativna strana takvog predviđanja jest akumuliranje greške predviđanja ako je horizont prevelik.

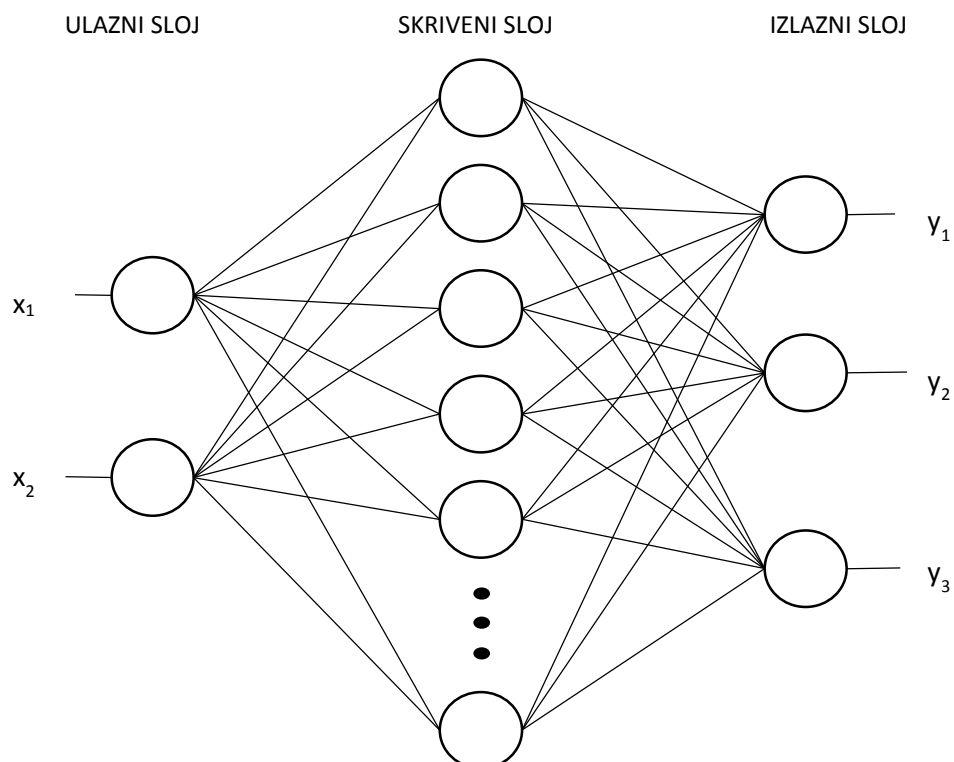
3. Opis čestih rješenja za predviđanje potrošnje

U nastavku će se prikazati i objasniti neke od najčešćih metoda koje se koriste kod kratkoročnog predviđanja potrošnje električne energije. Većina metoda su statističke metode ili postupci umjetne inteligencije.

Također, umjesto odabira samo jednog algoritma za predviđanje potrošnje, može se napraviti nekoliko algoritama koji se istovremeno izvršavaju. Nakon toga se može napraviti ansambl te kombinirati izlaze svih algoritama u jedan izlaz.

3.1. Neuronske mreže

Neuronska mreža je općenito najčešće korišteni prediktor pa se tako i najčešće koristi kao prediktor potrošnje električne energije. Neuronska mreža se pokazala kao dobra metoda zbog lakog učenja zavisnosti između ulaznih varijabli.



Slika 1 Primjer jednostavne neuronske mreže s jednim skrivenim slojem

Umjetna neuronska mreža je algoritam koji u osnovi oponaša živčani sustav. Neuronska mreža se sastoji od neurona koji su međusobno povezani i komuniciraju te svaki neuron za sebe nešto izračuna i pošalje dalje sljedećem neuronu. U pravilu su neuronske mreže podijeljene u slojeve i mogu se prepoznati tri osnovne vrste slojeva: ulazni sloj, izlazni sloj te jedan ili više skrivenih slojeva. Primjer jednostavne neuronske mreže je dan na slici 1 na kojoj se vidi ulazni sloj s dva ulaza, jedan skriveni sloj te izlazni sloj s tri izlaza.

Najčešće korištene verzije neuronskih mreža su unaprijedne neuronske mreže (engl. *feed-forward neural networks*) kao što su višeslojni perceptron (engl. Multilayer Perceptron, MLP) i neuronske mreže s radijalnim baznim funkcijama (engl. Radial Basis Function Networks, RBFN).

Prednost neuronskih mreža je mogućnost nelinearnog modeliranja zavisnosti između ulaznih varijabli. Dakako, to može biti i nedostatak jer neuronska mreža može modelirati podatke s prevelikim stupnjem nelinearnosti i time bespotrebno zakomplicirati (prenaučiti¹) model.

Predviđanje potrošnje neuronskom mrežom se može prikazati formulom:

$$x_i = f(X, W) + e_i \quad (8)$$

Gdje je f funkcija koja predstavlja neuronsku mrežu kojoj su ulazi prijašnje potrošnje X te težine neurona W , a e_i dodatna funkcija greške koja može biti npr. regularizacija koja kažnjava prevelike težine neuronske mreže. Isto tako se u ulaz neuronske mreže mogu dodati vanjski faktori.

Izbor arhitekture neuronske mreže

Izbor arhitekture neuronske mreže je jako bitan jer neuronska mreža može imati proizvoljan broj skrivenih slojeva i proizvoljan broj neurona u skrivenim slojevima. Na žalost, ne postoji neka pouzdana metoda odabira arhitekture neuronske mreže, već postoji samo nekoliko naputaka. Veličina neuronske mreže je kompromis između uspješnosti

¹ Model je prenaučan ako se jako dobro prilagodi podacima za učenje, ali loše generalizira. Više o tome u poglavlju 5. Mjerenja.

neuronske mreže i vremena za izračun izlaza. Kod nekih neuronskih mreža vrijeme trajanja izvođenja raste eksponencijalno s brojem neurona (funkcija).

U [1] je odabrana neuronska mreža s radijalnim funkcijama i jednim skrivenim slojem sa 64 neurona. Vrijednosti težina su postavljene nasumično prilikom inicijalizacije neuronske mreže. Učenje neuronske mreže je napravljeno algoritmom skaliranog konjugiranog gradijenta (engl. *Scaled-Conjugate Gradient*, SCG). U [3] je odabran potpuno povezani višeslojni perceptron s jednim skrivenim slojem sa 16 neurona. Za učenje neuronske mreže je zadužen popularni backpropagation algoritam. Razlog zbog kojeg je taj algoritam toliko rasprostranjen je njegova najveća prednost: veoma je uspješan u učenju neuronske mreže i jednostavan je za implementiranje. Njegov nedostatak je to što aktivacijske funkcije u neuronu moraju biti derivabilne.

3.2. Ansambl neuronskih mreža

Ansambl neuronskih mreža je skupina neuronskih mreža koje se izvršavaju neovisno i pokušavaju riješiti, tj. izračunati isti problem. Jedan od načina kako „spojiti“ sve neuronske mreže da izvana izgledaju kao jedan program može se napraviti aritmetičkom sredinom njihovih izlaza:

$$\hat{y}_{ansambl}(x_k) = \frac{1}{M} \sum_{i=1}^M \hat{y}_i(x_k) \quad (9)$$

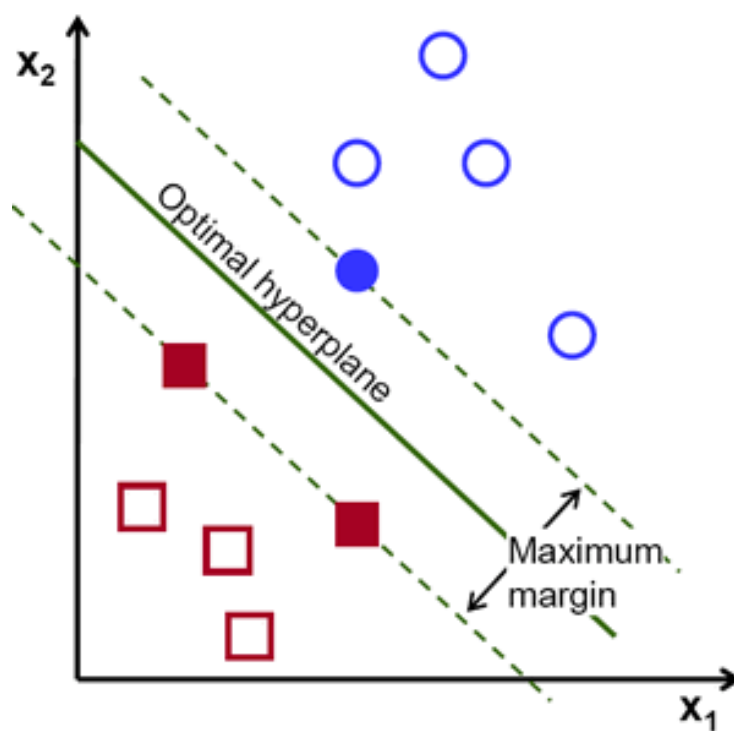
Drugi, malo kompliciraniji pristup je algoritam učenja negativne korelacije. Ideja algoritma jest da se ne uče sve inačice algoritma odvojeno, već da se uvede funkcija kazne kojom se pokušava smanjiti korelacija između inačica. U [1] je predložen model učenja regulariziranom negativnom korelacijom u kojoj je uveden regularizacijski faktor kako bi se spriječila prenaučenosť. Tada se greška svake neuronske mreže e_i računa formulom:

$$e_i = \frac{1}{M} \sum_{k=1}^N (\hat{y}_i(x_k) - y_i(k))^2 - \frac{1}{M} \sum_{k=1}^N (\hat{y}_i(x_k) - \hat{y}_{ansambl}(x_k))^2 + \alpha_i w_i^T w_i \quad (10)$$

Gdje je prvi dio formule greška prediktora, drugi dio je korelacija između jedne mreže i cijelog ansambla, a treći je regularizacija težina s parametrom $\alpha \in [0, 1]$. Ukupna greška cijelog ansambla je zbroj grešaka svih neuronskih mreža.

3.3. Box - Jenkins metoda ili ARIMA model

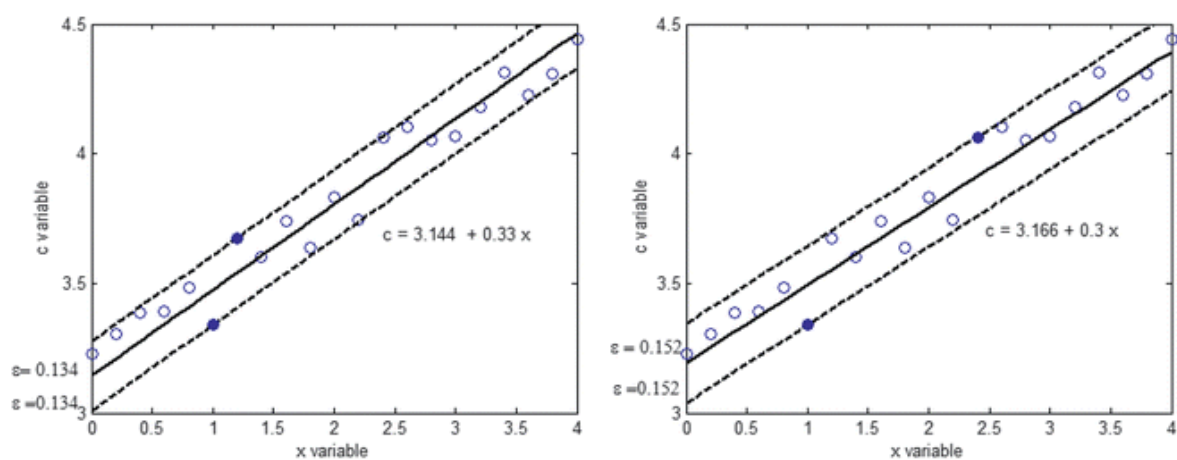
Vremenski nizovi (engl. *time series*) su česti pristup kod rješavanja problema predviđanja potrošnje i bitan su dio kod modeliranja vremenskih nizova Box - Jenkins metodom. Ideja je naći model koji najbolje opisuje niz podataka koji je nastao u vremenu. U [1] je pretpostavljeno da je budući signal linearna funkcija prethodnih mjerenja potrošnje s dodatkom šuma. Zbog sezonalnosti podataka je korištena SARIMA, koja je sezonalna verzija modela ARIMA (engl. *Autoregressive Integrated Moving Average*). ARIMA je jedan od značajnijih modela koji Box - Jenkins metoda koristi. Zbog toga što je ideja ARIMA modela da aproksimira vremenski niz, njeni ulazi su u pravilu prijašnje potrošnje iako to ne mora biti pravilo. ARIMA model se može proširiti i da prima vanjske faktore te se tada model zove ARIMAX. Jednako se i SARIMA može proširiti na SARIMAX. Jednom kada se odrede svi parametri, ARIMA može predviđati za proizvoljan broj vrijednosti u budućnosti. Kako je predviđanje iterativni postupak, za predviđanje u trenutku t je potrebna vrijednost u trenutku $t-1$. Ako se predviđa više od jedne vrijednosti, tada se sljedeća predviđanja odvijaju na temelju prijašnjih predviđanja, odnosno za predviđanje vrijednosti u trenutku $t+1$ se koristi predviđena vrijednost iz trenutka t iz prethodnog koraka.



Slika 2 Osnovni princip metode potpornih vektora

3.4. Metoda potpornih vektora

Metoda potpornih vektora (engl. *Support Vector Machines*, SVM) je algoritam nadziranog učenja koji se koristi za klasifikaciju. Ideja je da se podaci podijele u dva skupa (klase) koje su međusobno najudaljenije moguće, kao što se vidi na slici 2. Vrijednosti koje su najbliže granici (na slici su to ispunjeni elementi) se zovu potporni vektori. SVM je u osnovi binarni klasifikator, no može se proširiti da klasificira u proizvoljan broj klasa. Regresija potpornih vektora (engl. *Support Vector Regression*, SVR) je algoritam koji je nastao iz metode potpornih vektora, ali služi za regresiju. Cilj mu je pronaći funkciju koja ima odstupanje najviše ϵ od stvarnih vrijednosti podataka. Primjer jedne regresije je dan na slici 3. Zanimljivo je primjeriti da su primjeri za klasifikaciju i regresiju klasificirani, odnosno aproksimirani s linearnim krivuljama. Tom problemu se može doskočiti preslikavanjem primjera iz ulaznog prostora u prostor značajki. Za taj postupak se najčešće koriste radijalne funkcije. U [2] je SVR korišten kao algoritam za predikciju potrošnje električne energije i za to su korištena 24 paralelna modela, za svaki sat u danu po jedan model. Kod izbora parametara modela (γ u radijalnoj baznoj funkciji, ϵ i regularizacijski parametar) autori su se odlučili na pretragu rojem čestica (engl. *Particle Swarm Pattern Search Method*). Također, autori su koristili algoritam pretraživanja značajki kako bi unaprijedili točnost predviđanja. Unaprijed su generirali preko 200 mogućih ulaznih parametara te su ostavili algoritmu da odabere njih dvadesetak kao ulaz u SVR.



Slika 3 Primjer regresije potpornim vektorima

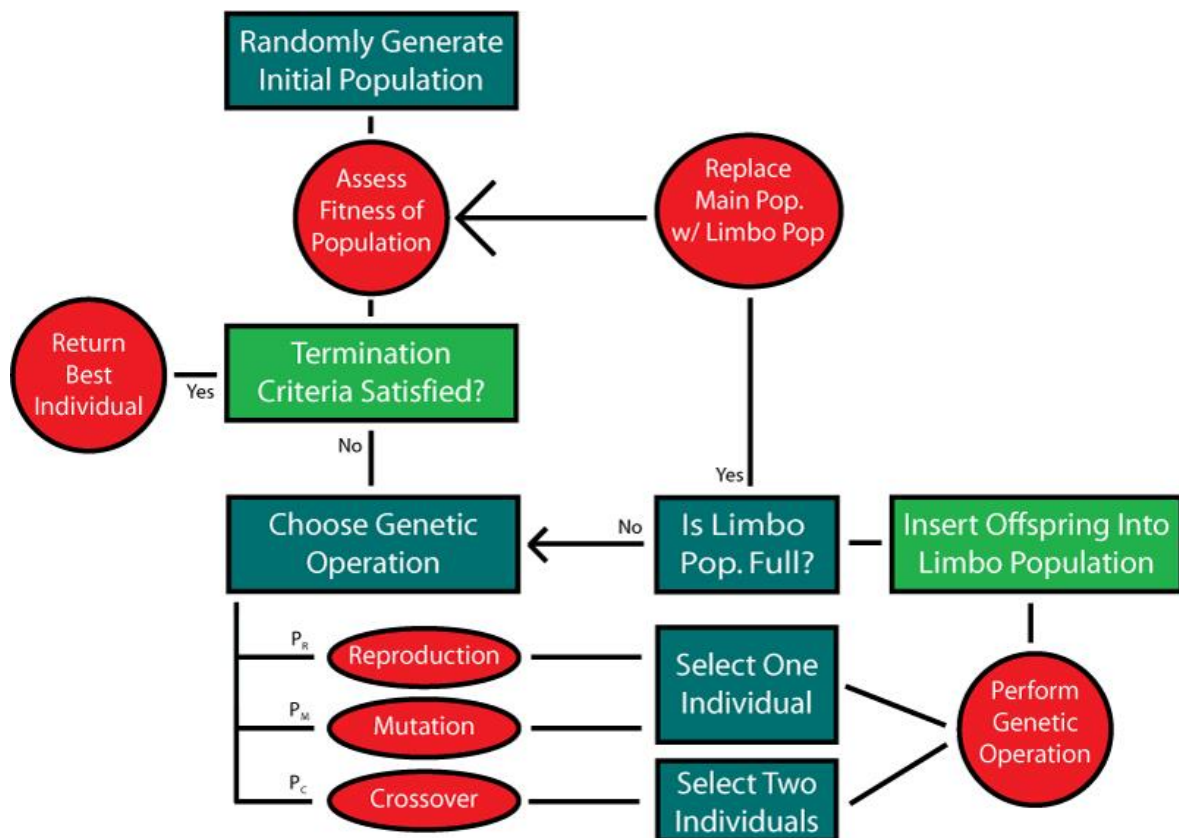
4. Genetsko programiranje

U sklopu rada je trebalo odabrati algoritam za predikciju te je odabran algoritam genetskog programiranja. Genetsko programiranje je algoritam iz skupine evolucijskih algoritama koji je inspiriran idejom učenja i evolucije računalnih programa. Evolucijski algoritmi su dobili ime zbog učenja oponašanjem evolucije na način da postoje jedinke koje se tijekom generacija sve više poboljšavaju. U genetskom programiranju u pravilu se jedinkom smatra računalni program, najčešće prikazan stablom. U daljnjem tekstu će se pod pojmom jedinka podrazumijevati da se radi o stablu. Kada se kaže genetski kod, također se misli na stablo jer je stablo ono što definira kakva je koja jedinka.

Cilj genetskog programiranja, kao i svih evolucijskih algoritama, je napraviti neki početni skup jedinki, populaciju, te kroz iteracije birati bolje jedinke, a odbacivati lošije. Bitna stvar je da se ne smiju sve loše jedinke odbaciti. Iako je neka jedinka loša, ne znači da je sav njen genetski kod loš. Zbog toga i lošije jedinke u nekoj mjeri moraju preživljavati.

Osnovni način rada genetskog programiranja je prikazan na slici 4. Prvi korak genetskog programiranja je **generirati početnu, nasumičnu populaciju**. Nakon toga, sve jedinke u populaciji se moraju evaluirati. **Evaluacija** je postupak određivanja koliko je neka jedinka kvalitetna. Evaluacijom se izračunava dobrota, odnosno greška jedinke. Odluka o tome da li se izračunava dobrota ili greška ovisi o problemu koji se rješava i o funkciji kojom se jedinka evaluira. U slučaju predikcije potrošnje električne energije, funkcija greške može biti razlika između stvarne vrijednosti potrošnje i predviđene vrijednosti potrošnje.

Postoje dvije osnovne vrste algoritama genetskog programiranja: **generacijski i eliminacijski algoritam**. Generacijski algoritam u svakoj iteraciji odabire nove jedinke ovisno o njihovoj dobroti, nad njima provodi genetske operatore te ih stavlja u novu generaciju. Eliminacijski algoritam u svakoj iteraciji odabire nove jedinke ovisno o njihovoj dobroti, primjeni operator nad njima te ih stavlja u istu generaciju umjesto nekih slučajno odabranih loših jedinki. Na slici 4 je prikazan generacijski algoritam.



Slika 4 Grafički prikaz rada algoritma genetskog programiranja

Bez obzira radi li se o eliminacijskom ili generacijskom algoritmu, jedinke se nekako moraju odabrati iz populacije. Za to je zadužen **operator selekcije**. Njegova zadaća je izabrati dobru, odnosno lošu jedinku iz populacije uz pomoć dobrote ili greške jedinke. Ako se odabire dobra jedinka, tada se uz veću vjerojatnost mora uzeti neka bolja jedinka iz populacije. Primjer operatora selekcije je turnirska selekcija koja će biti objašnjena kasnije u tekstu.

Nakon što se izaberu dobre jedinke iz populacije, nad njima se izvršava operator, iako može biti i obrnuto, da se prvo odabire operator, a potom jedinke iz populacije. Postoje dvije osnovne vrste operatora: **operator križanja** i **operator mutacije**. Operator križanja uzima dvije jedinke kao roditelje te miješajući njihov genetski kod (podstabla ili čvorove) te iz njih kreira jednu ili više novih jedinki, djecu. Operator mutacije uzima samo jednu jedinku i unutar nje same napravi slučajnu promjenu, npr. promijeni vrijednost slučajno izabranog čvora.

Do sada opisanim metodama se stvaraju nove jedinke i stavljaju u populaciju jedinki, no ništa nam nije reklo do kada se to mora raditi. Zato postoje **uvjeti zaustavljanja**. Njih može biti puno, a neki od najčešćih su broj odrađenih generacija ili iznos dobrote najbolje jedinke. Nakon što se algoritam zaustavi jer je došao do kraja, iz zadnje napravljene populacije se uzima najbolja jedinka te se ona smatra rješenjem algoritma.

4.1. Detalji implementacije GP-a

U nastavku su dani detaljni opisi implementacije pojedinih dijelova, operatora i algoritama genetskog programiranja.

4.1.1. Populacija, jedinka, stablo i čvorovi

Jedna populacija je niz jedinki. Svaka jedinka je prikazana jednim stablom. Svako stablo je prikazano nizom čvorova koji su određenim odnosima. Svaki čvor ima svog roditelja (osim korijena) i svoju djecu. Broj djece ovisi o tipu čvora. Postoje dvije osnovne skupine čvorova: funkcijski i završni. Funkcijski čvorovi imaju jedno ili više djece (ovisno o kojoj se funkciji radi). Završni čvorovi nemaju djece i oni mogu biti konstante ili varijable. Broj konstanti i varijabli ovisi o broju ulaznih varijabli koji se koriste za predviđanje. Njihov omjer je uvijek 1:1.

Početno generiranje jedinki se radi metodom rasta (engl. *Grow Method*). Metoda rasta gradi stablo na sljedeći način:

- ako je generirani čvor na razini manjoj od najmanje, generira se funkcijski čvor,
- ako je između najmanje i najveće dopuštene razine, slučajnim odabirom se bira između funkcijskog i završnog čvora,
- ako je na najvećoj dopuštenoj dubini, bira završni čvor.

Na takav je način dubina pojedinog podstabla prepuštena slučajnom odabiru te su stabla iregularnog oblika i promjenjive dubine.

4.1.2. Algoritam

Algoritam određuje osnovni način na koji genetsko programiranje funkcionira. Algoritam može biti generacijski ili eliminacijski. Također, algoritam može u jednoj iteraciji primijeniti

samo jedan operator (ili mutaciju ili križanje, uz određenu vjerojatnost) ili oba operatora na odabrane jedinke. Implementirana su četiri različita algoritma: eliminacijski s jednim operatorom, eliminacijski s dva operatora, generacijski s jednim operatorom i generacijski s dva operatora. Eliminacijski algoritmi su implementirani na način da je selekcija već ugrađena u njih. Algoritam na početku iteracije radi turnir željene veličine, nakon čega uzima jednu ili dvije najbolje jedinke iz turnira (ovisno o operatoru), izvrši operator nad njima te ih stavi na mjesto najlošije jedinke iz istog turnira. Na taj način se turnir radi samo jednom po iteraciji. Generacijski algoritmi su implementirani standardno. [7]

4.1.3. Selekcija

Kao operator selekcije je napravljena turnirska selekcija. Turnirska selekcija slučajno odabire n jedinki iz populacije te vraća najbolju od njih. Selekcija se može konfigurirati s veličinom turnira o kojoj ovisi koliki je selekcijski pritisak. Za potrebe eliminacijskog algoritma, turnirska selekcija može vratiti i najgoru jedinku iz turnira.

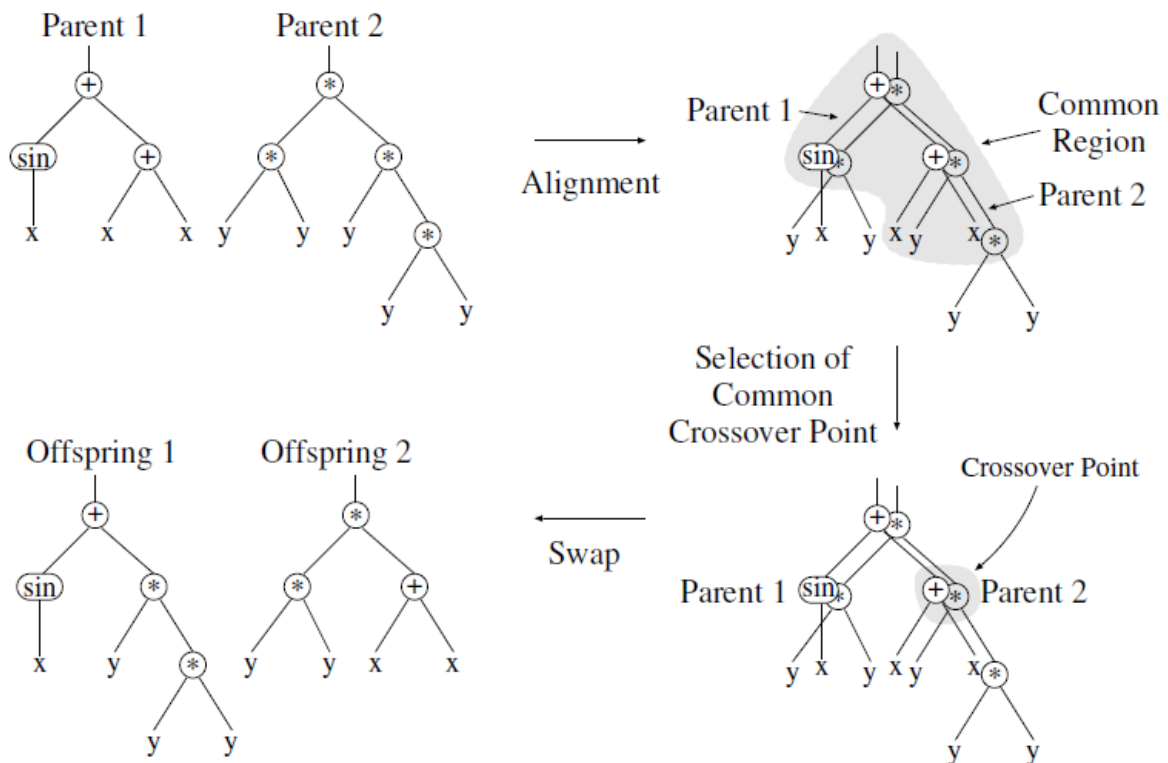
4.1.4. Mutacija

Operator mutacije je operator koji djeluje nad samo jednom jedinkom tako da neki njeni dio malo ili malo više promijeni. U radu su implementirane dvije vrste mutacija: *point* i *hoist* mutacija. *Point* mutacija je mutacija čvora. Ona obilazi stablo te s određenom vjerojatnošću, vjerojatnost mutacije, mijenja odabrani čvor za neki drugi čvor. Jedino ograničenje je da novo-odabrani čvor mora imati isti broj djece, da se mogu sačuvati sva podstabla početnog čvora. *Hoist* mutacija je mutacija stabla. Ona nalazi jedan čvor u početnom stablu te taj novi odabrani čvor postavlja kao novi korijen stabla, čime se odbacuje cijeli ostatak stabla. Prednost ove mutacije je smanjivanje najveće dubine stabla.

Kao dodatak, u mutaciju čvora je dodana dodatna (pod)mutacija konstante. To znači da kada se želi mutacijom čvora mutirati konstanta, 50% šanse su da se to napravi regularnom mutacijom čvora, a 50% šanse su da se trenutnoj konstanti doda neka vrijednost iz normalne razdiobe.

4.1.5. Križanje

Operator križanja je operator koji uzima dvije jedinke, roditelje, i vraća dvije nove jedinke, djecu. U ovom radu su implementirane dvije vrste križanja: križanje s jednom točkom prekida (engl. One Point Crossover) i uniformno križanje (engl. *Uniform Crossover*). Imena križanja dosta podsjećaju na križanja iz genetskih algoritama s binarnom reprezentacijom te je i namjera bila oponašati ta križanja. 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 5. 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 5 Primjer križanja s jednom točkom prekida [7]

4.1.6. Evaluacija jedinke

Da bi se odredila greška pojedine jedinke, ona mora biti evaluirana. 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:

$$MAPE = 100\% \cdot \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (11)$$

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

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (12)$$

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

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (13)$$

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

4.1.7. Uvjeti zaustavljanja

Uvjeti zaustavljanja se provjeravaju nakon svake iteracije/generacije. Kod generacijskog algoritma je to nakon svakog kreiranja nove generacije, a kod eliminacijskog nakon što se u postojeću populaciju doda onoliko novih jedinki koliko je populacija velika. U ovom radu je implementirano da se može zadati više uvjeta zaustavljanja odjednom. Implementirani uvjet zaustavljanja je broj odrađenih generacija.

4.2. Određivanje vrijednosti parametara

Osim određivanja strukture GP-a, kao što su vrsta algoritma i operatori, bitan dio je podešavanje parametara. Problem određivanja parametara za GP može biti jako kompleksan [6], [13] te ne postoji neki univerzalni skup parametara koji dobro rade na svim problemima, tako da je određivanje i podešavanje parametara jedan od osnovnih dijelova implementacije genetskog programiranja. Ispod teksta je prikazana konfiguracija genetskog programiranja koju koristi algoritam implementiran u ovom radu:

```
<GeneticProgramming>
  <Algorithm name="SteadyStateTournamentTwoOperators">
    <PopulationSize>500</PopulationSize>
    <ParamK>3</ParamK>
    <Termination>
      <Entry name="NumberOfGenerations">100</Entry>
    </Termination>
  </Algorithm>
  <Tree>
    <MaxDepth>8</MaxDepth>
    <MinDepth>3</MinDepth>
    <FunctionSet>+ - * / ifRadniDan</FunctionSet>
  </Tree>
  <Crossover>
    <Name>UniformCrossover</Name>
  </Crossover>
  <Mutation>
    <Name>PointMutation</Name>
    <MutFactor>0.01</MutFactor>
    <ExtraConstantMutation>true</ExtraConstantMutation>
  </Mutation>
  <Evaluation>
    <TrainEvaluator>MAPE</TrainEvaluator>
    <Crossvalidation>>false</Crossvalidation>
    <Data>
      <DataPath>PoSatima/sat{ID}.txt</DataPath>
      <PreviousLoads>5</PreviousLoads>
    </Data>
  </Evaluation>
  <Log>
    <GenerationFrequency>5</GenerationFrequency>
    <FileName>Logovi/log{ID}.txt</FileName>
    <BatchNo>10</BatchNo>
  </Log>
</GeneticProgramming>
```

Tablica 1 Prikaz implementiranih operatora i algoritama

	Parametri	Moguće vrijednosti
Algoritam	Ime (vrsta) algoritma	Eliminacijski s dva operatora, Eliminacijski s jednim operatorom, Generacijski s dva operatora, Generacijski s jednim operatorom
	Veličina populacije	Bilo koji cijeli broj, uobičajeno na intervalu [10, 1000]
Uvjet zaustavljanja	Broj generacija	Bilo koji cijeli broj, uobičajeno na intervalu [10, 1000]
Selekcija	Veličina turnira	Bilo koji cijeli broj, uobičajeno na intervalu [2, 10]
Stablo	Najveća dubina	Bilo koji cijeli broj, uobičajeno je veći od 5
	Najmanja dubina	Bilo koji cijeli broj, uobičajeno je manji od 5
	Funkcijski čvorovi	Niz znakova odvojenih razmakom; + - / * sin log ifRadniDan
Križanje	Ime (vrsta) križanja	Križanje s jednom točkom prekida, Uniformno križanje
	Vjerojatnost križanja	Realni broj na intervalu [0, 1], uobičajeno oko 0.9
Mutacija	Ime (vrsta) mutacije	Point mutacija, hoist mutacija
	Vjerojatnost mutacije	Realni broj na intervalu [0, 1], uobičajeno oko 0.01
	Dodatno mutiranje konstante	Binarna vrijednost, da/ne
Evaluacija	Procjenitelj pogreške	MSE, MAPE, MAE
	Broj prijašnjih mjerenja	Cijeli broj, u pravilu na intervalu [1, 10]
Unakrsna validacija	Rotiraj preklope	Binarna vrijednost, da/ne
	Skup za evaluaciju	Binarna vrijednost, da/ne
	Broj preklopa	Cijeli broj, uobičajeno oko 5

U tablici 1 su prikazani svi operatori, njihovi parametri i moguće vrijednosti za parametre koji su napravljeni u sklopu ovog rada. Kratke napomene vezane za implementaciju:

- Vjerojatnost križanja se koristi samo u algoritmima s jednim operatorom gdje služi za odabir između mutacije i križanja,
- Kod eliminacijskih algoritama, jedna generacija jest kada se u trenutnu populaciju doda onoliko jedinki koliko je velika populacija,
- U eliminacijskom algoritmu s dva operatora, najmanja veličina turnira je tri zbog toga što se u istom turniru biraju i roditelji i najlošija jedinka za eliminaciju,
- Dodatno mutiranje konstante je aktivno samo kada se koristi mutacija čvora,
- Unakrsna validacija može, a i ne mora biti uključena.
- Prijašnje mjerenje je mjerenje koje je napravljeno u istom satu, samo prijašnjem danu.

4.3. Ulazni podaci

Podaci koji su korišteni u ovom radu su dobiveni iz dva izvora:

- **Podaci o potrošnji električne energije** su dobiveni od Fakulteta elektrotehnike i računarstva kao stvarni podaci izmjereni 2011. godine. Podaci su uglavnom u 15-minutnoj rezoluciji, s podatkom u kojem vremenu je izmjerena potrošnja, iznos potrošnje radne (kW) i jalove snage (kVar).
- **Podaci o vremenskim prilikama** u Zagrebu su dobiveni od Državnog hidrometeorološkog zavoda kao stvarni podaci izmjereni u Maksimiru 2011. godine. Podaci su u satnoj rezoluciji i dostupni su podaci o vremenu uzimanja mjerenja, temperaturi (°C), vlažnosti (%), tlaku (hPa), trajanje sijanja sunca (0.1h), količina (mm) i trajanje (min) oborine, naoblaka (0-10) i vidljivost (km).

Iako je na raspolaganju bilo puno podataka, nisu svi od njih korišteni. Od svih navedenih podataka, podaci koji će se koristiti u daljnjim testiranjima genetskog programiranja su vrijeme mjerenja, potrošnja radne snage i temperatura, svi u satnoj rezoluciji. Uz to će se koristiti još i kalendarski podaci kao što su podaci o mjesecu i radnom danu. Podatak o satu za koji se predviđa nije potreban jer se razvija sustav koji ima 24 prediktora, svaki specijaliziran za jedan sat u danu.

Svi podaci koji su ulaz u genetsko programiranje su prethodno obrađeni da budu lakši za učitavanje i predviđanje. Pregled mogućih podataka dan je u tablici 2.

Tablica 2 Prikaz mogućih ulaznih podataka

Podatak	Tip podatka	Raspon	Mj. jedinica	Opis
Potrošnja el. energije	<i>Double</i>	[100, 500]	kWh	Iznos potrošnje električne energije
Temperatura	<i>Double</i>	[-20, 50]	°C	Iznos temperature
Vlažnost zraka	<i>Double</i>	[0, 1]	%	Postotak vlažnosti zraka
Tlak	<i>Double</i>	[800, 1200]	hPa	Iznos tlaka zraka
Radni dan	<i>Bool</i>	{0, 1}	-	Oznaka da li je radni dan
Mjesec	<i>Integer</i>	[0, 11]	-	Redni broj mjeseca u godini; 0 = Siječanj, 1 = Veljača, ...

4.4. Pregled postojećih implementacija genetskog programiranja za predviđanje

Genetsko programiranje nije algoritam koji će prvo pasti na pamet ako se želi implementirati predviđanje, bilo da je predviđanje potrošnje bilo da je išta drugo. Za to postoji više razloga. Jedan od njih je da postoje dokazano učinkoviti algoritmi za predviđanje, kao što je neuronska mreža. Drugi razlog može biti taj da GP ima puno parametara koji se moraju podesiti prije nego što se može dobiti bolje predviđanje. No, unatoč svemu tome, GP je istraživao na polju predviđanja i postoji nekoliko radova koji opisuju implementaciju GP-a radi predviđanja.

Jedan od istraživanja koji koristi GP za predviđanje je [8] u kojem se opisuje primjena GP-a na dijagnozu raka usne šupljine. Ideja je da se svi pacijenti klasificiraju u dvije skupine: rizična i nerizična. Svi završni čvorovi su binarne varijable, a funkcijski su binarne funkcije koje rade s binarnim varijablama (I, ILI, NE). GP je uspoređen s neuronskom mrežom i ručnim unosom, gdje je bio lošiji od ručnog unosa, ali približno dobar kao i neuronska

mreža. Razlog zbog kojeg se rezultati ovog rada ne mogu primijeniti na STLF je taj da se u [8] radi klasifikacija, dok je za STLF potrebna regresija.

U [9] je napravljena implementacija GP-a za predviđanje cijena dionica koristeći vremenske nizove. Učenje algoritma je podijeljeno na dvije faze: učenje na neobrađenim podacima (stanje dionica u prethodnim danima) i učenje na izvedenim podacima (u cijenu dionica su unesene dividende). Napravljeni modeli nisu uspoređeni s drugim algoritmima, ali je učenje na izvedenim podacima bilo točnije, prvenstveno jer je GP-u dano više informacija na ulaz. Ovaj rad može biti relevantan za STLF, ali nisu dani konkretni detalji implementacije, već samo opis ulaznih podataka.

U [10] je napravljena implementacija GP-a za predviđanje hrapavosti površina nakon glodanja. Ulazni parametri su brzina svrdla, brzina micanja objekta, dubina glodanja i vibracije. Kao operatori su korišteni križanje i reprodukcija (kopiranje jedinke). Autori su zadovoljni predviđanjem, ali rezultati nisu uspoređeni s nekim drugim modelima. Razlog zbog kojeg je ovaj rad teško primjenjiv na STLF je to što se kod obrade materijala uvijek mogu očekivati isti rezultati za iste ulazne parametre (ako su uzeti u obzir svi relevantni parametri), dok je kod STLF problem da je previše kompleksno odrediti sve ulazne parametre i kako oni utječu.

U [11] je napravljena implementacija GP-a za dugoročno predviđanje potrošnje električne energije, gdje se predviđa ukupna potrošnja za cijelu godinu. Ulazne varijable su BDP i broj stanovništva. Korišten je generacijski algoritam s operatorima reprodukcije i križanja. GP je uspoređen s fiksnim regresijskim modelom te je pokazao bolje rezultate.

Genetsko programiranje kao prediktor nije čest odabir, kao što se i vidi u ovim radovima. Neki drugi algoritmi, npr. neuronska mreža, toliko su obrađivani, implementirani i testirani da već postoji neki model koji se može primijeniti na određenu problematiku. S genetskim programiranjem nije takav slučaj. U svim ovim radovima nije obrađena analiza utjecaja samog GP-a, već samo ulazne varijable. Također, GP se češće koristi za klasifikaciju jer može poslužiti kao stablo odluke, kao što je prezentirano u [8].

Za razliku od ostalih radova opisanih u ovom poglavlju, u [12] se u detalje opisuje kako GP može koristiti za predviđanje cijene dionica. To je dobar primjer jer se cijena dionica kroz dan može dosta mijenjati, isto kao i potrošnja električne energije. U radu je korištena jako

velika populacije, 2000 jedinki za testiranje i 5000 jedinki za predviđanje. No, bitna razlika je u tome da se cijene dionica „naizgled“ bez razloga padaju i rastu, dok kod potrošnje energije postoje brze promjene (tijekom dana) i spore promjene (najveća potrošnja u danu). Takva frekvencije podataka ne postoje u cijenama dionica i za dionice je teško izolirati vanjske faktore i kako oni mogu utjecati. Unatoč svemu, autor daje pozitivnu ocjenu GP-u za naznakom da je tu potrebno još dosta istraživanja.

Zbog nemogućnosti pronalaska preporuke za genetsko programiranje koje se koristi za predviđanje kratkoročne potrošnje električne energije, u ovom radu su napravljeni eksperimenti radi podešavanja parametara samog GP-a i određivanja ulaznih varijabli.

4.5. Genetsko programiranje kao prediktor potrošnje

Osnovna ideja je implementirati 24 paralelna, neovisna prediktora algoritmom genetskog programiranja koji predviđaju za vremenski raspon od 24 sata, za svaki sat po jedan prediktor. Svaki prediktor se uči na svojem skupu podataka, iako se neki podaci mogu koristiti za učenje više različitih prediktora.

Prediktori mogu primiti razne ulazne varijable dok god su zadani u određenom formatu. U teoriji, prediktori se mogu proširiti i da predviđaju za vremenski horizont koji je veći od 24 h, no tada treba prilagoditi da se predviđena potrošnja sprema među podatke i daje na ulaz prediktorima koji predviđaju za iza inicijalnih 24 sata. Inicijalno postavljeni ulazi u prediktore su temperatura u satu za koji se predviđa, 5 mjerenja u istom satu u prijašnjim danima, potrošnja u prijašnjem satu, informacija da li se predviđa za dan koji je radni ili neradni i koji je mjesec. Inicijalni vremenski raspon podataka nad kojima se uči je od 1.1.2011 do 1.6.2011. Izlaz iz pojedinog prediktora je predviđanje potrošnje za pojedini sat u danu te se tako pokušava postići specijalizacija prediktora za pojedini sat.

Također, kasnije se mogu neki slični i bliski prediktori spojiti tako da postoje zapravo prediktori za određene dijelove dana.

5. Mjerenja

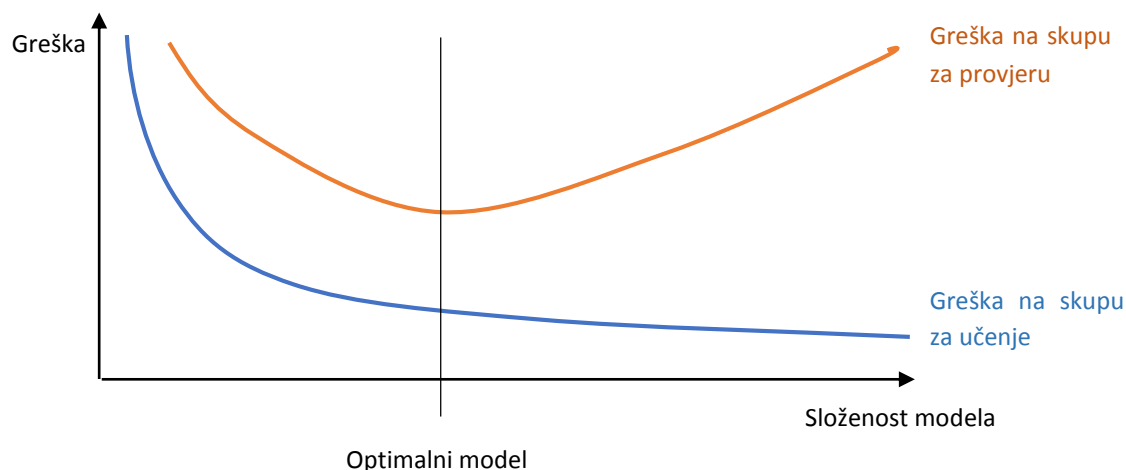
U ovom poglavlju će biti opisana sva mjerenja koja su napravljena kako bi se poboljšao rad GP-a. Mjerenja se grubo mogu podijeliti u tri skupine: podešavanje parametara GP-a konvergencijom, određivanje parametara u unakrsnu validaciju i odabir ulaznih varijabli, odnosno odabir značajki.

Za pravilan odabir parametara bi se trebao raditi kartezijski produkt svih kombinacija za sve parametre, no u ovoj implementaciji postoji previše parametara da bi se mogli tako odrediti. Zbog toga će se pokušati naći optimalni parametar po parametar. Takav pristup je jednako kompliciran jer su parametri zavisni te se mijenjanjem jednog parametra može smanjiti dobar utjecaj vrijednosti nekog drugog parametra, kao što je pokazano u [13].

Unakrsna provjera

Parametri se mogu određivati na nekoliko različitih načina. Prvi način je koristiti konvergenciju. Ideja je pustiti algoritam da se vrti jako dugo i tako puno puta s različitim parametrima. Kad završi, proučiti kvalitetu rješenja (dobrotu ili grešku) i uzeti parametar koji je najkvalitetniji. Problem s nekim parametrima je taj da što više mijenjamo parametar u nekom smjeru (npr. povećavamo), to dobivamo bolje rezultate algoritma. Naravno, to ne mora značiti da je i naučeni model dobar. U takvom slučaju može doći do prenaučenosti modela u kojem se on previše prilagodi podacima na kojima uči i da jako loše generalizira. U takvim slučajevima, vrijednost parametra se određuje unakrsnom provjerom.

Postoji nekoliko različitih vrsta unakrsnih provjera, no u ovom radu je korištena unakrsna provjera s k preklopa. Osnovna ideja je da se cijeli skup primjera za učenje podijeli na k dijelova i ukupno se napravi k iteracija. U svakoj iteraciji, algoritam se uči na $k-1$ preklopu, a grešku provjerava na preostalom preklopu. Svaka sljedeća iteracija provjerava grešku na drugom preklopu. Jako je bitno da presjek svih preklopa bude prazan skup jer se jedino tako može provjeriti sposobnost generalizacije. Na kraju, greške na skupu za učenje i greške na skupu za provjeru se mogu prikazati na grafu kao na slici 6. Kako želimo algoritam koji najbolje generalizira, odabrani parametar će biti onaj koji ima najmanju grešku na skupu za provjeru.

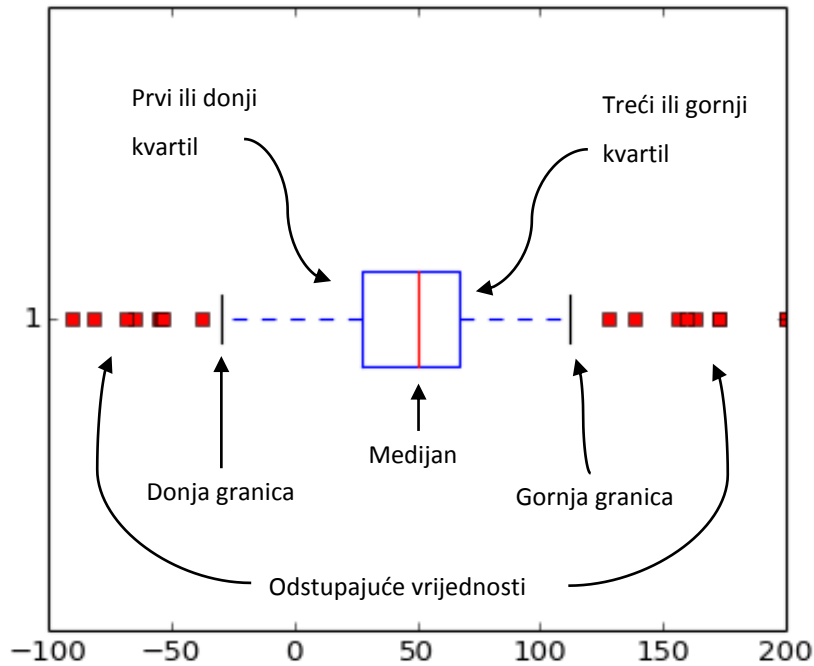


Slika 6 Određivanje optimalnog modela metodom unakrsne provjere

Boxplot prikaz

U daljnjem tekstu, rezultati mjerenja su većinom prikazani tablično s relevantnijim podacima (najmanja ili najveća vrijednost, prosjek, itd.) ili grafički uz pomoć kutijastog dijagrama (engl. *boxplot*, *box-and-whiskers plot*). *Boxplot* je vizualan prikaz nekih statističkih obilježja podataka. Dosta govori o raspodijeli podataka jer označava gdje se nalazi prvi, drugi i treći kvartil, raspon u kojemu se nalazi podaci kao i odstupajuće vrijednosti koje su izvan tog raspona. Detalji su prikazani na slici 7.

Postoje različite inačice *boxplota* koje se razlikuju u tome kako se određuje raspon u kojemu se nalaze podaci. U ovom radu je korišten *boxplot* koji donju granicu računa najmanju vrijednost koja je na udaljenosti od 1.5 IQR od donjeg kvartila, a gornju granicu kao najveću vrijednost koja je na udaljenosti od 1.5 IQR od gornjeg kvartila, gdje je IQR interkvartilni raspon koji se računa kao razlika (udaljenost) između gornjeg i donjeg kvartila.



Slika 7 Primjer *boxplot* prikaza podataka

Normiranje prediktora

Zbog 24 prediktora i zbog višestrukog pokretanja svakog prediktora, problematično je odrediti utjecaj pojedinih parametara. Zbog toga se sve greške unificiraju tako da se normiraju sve greške istog prediktora. Normiranje jednog prediktora se radi tako da se sve njegove greške podijele s prosječnom pogreškom. Takvim postupkom se dobivaju normirane greške koje su međusobno usporedive.

5.1. Određivanje parametara GP-a

U nastavku će biti dan pregled mjerenja i određivanja parametara za genetsko programiranje, za sve parametre iz tablice 3. Za svaki parametar će se isprobati nekoliko različitih vrijednosti da bi se utvrdilo da li ima utjecaj i koliki. Za svaki sat, odnosno prediktor, napravljeno je deset mjerenja s istim parametrima, kako bi se smanjio utjecaj stohastičkih elementa. To znači da je za jednu konfiguraciju, npr. jednu veličinu populacije, napravljeno 240 mjerenja, za svaki od 24 sata po 10 mjerenja.

Tablica 3 Inicijalni parametri i ulazni podaci za GP

	Parametri	Inicijalne vrijednosti
Algoritam	Ime (vrsta) algoritma	Eliminacijski s dva operatora
	Veličina populacije	150
	Veličina turnira	3
Uvjet zaustavljanja	Broj generacija	500
Stablo	Najveća dubina	8
	Najmanja dubina	3
	Funkcijski čvorovi	+ - / * ifRadniDan
Križanje	Ime (vrsta) križanja	Križanje s jednom točkom prekida
Mutacija	Ime (vrsta) mutacije	Point mutacija
	Vjerojatnost mutacije (čvora)	0.01
	Dodatno mutiranje konstante	Da
Evaluacija	Procjenitelj pogreške	MSE
	Broj prijašnjih mjerenja	5

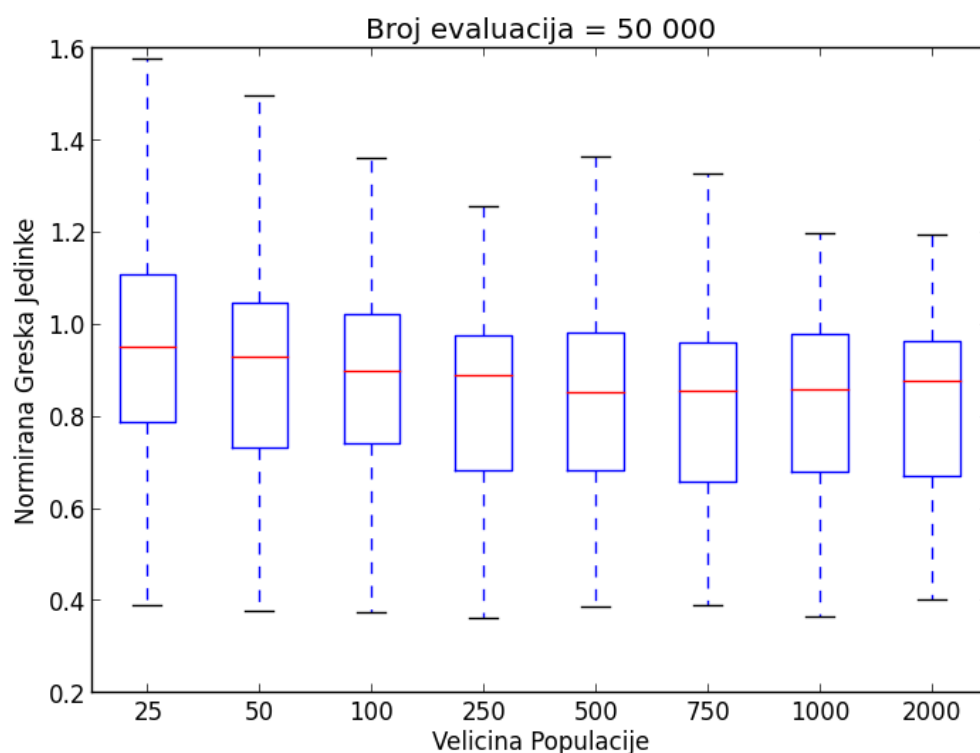
5.1.1. Veličina populacije

Veličina populacije je jedan od osnovnih i značajnijih parametara te se zato prvi ispituje. U pravilu, veličina populacije se stavlja ili da je jako mala (između 10 i 20 jedinki) ili jako velika (preko 100, nekad i 1000 jedinki). Ako je populacija manja, onda se pušta veći broj generacija (>100) da nađe optimalno rješenje. Ako je populacija veća, onda se pušta manji broj generacija (<50) da nađe optimalno rješenje. Kada se tako prezentira, i jedna i druga inačica naprave otprilike jednak broj iteracija. Ono što je značajnija razlika među njima, koja se možda ne primijeti odmah, jest ta da kod veće populacije postoji veća raznolikost jer se teže eliminiraju sve lošije jedinke. Zbog toga u ovom eksperimentu veličina populacije ide od malih vrijednosti, oko 10, pa sve do 1000. Jedini je uvjet da svaki algoritam napravi 50000 iteracija. Konfiguracija ostalih parametara je ista kao u tablici 3.

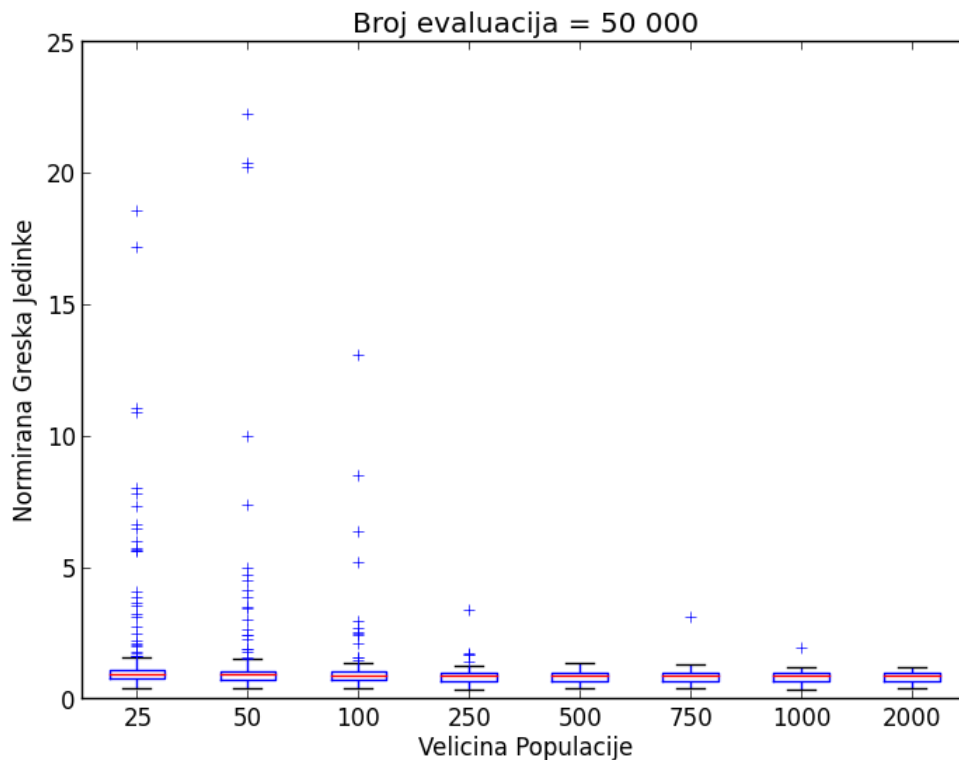
Tablica 4 Rezultati za različite vrijednosti veličine populacije

Br.Jed	25	50	100	250	500	750	1000	2000
<i>Max</i>	17.2136	20.5291	12.1525	3.22246	1.31990	2.97703	1.86245	1.19497
<i>Avg</i>	1.42915	1.29134	1.00776	0.84023	0.81169	0.80914	0.81066	0.82412
<i>Min</i>	0.36052	0.34705	0.34504	0.33492	0.35573	0.36053	0.33582	0.40236

U tablici 4 su prikazani rezultati mjerenja. Iz nje se vidi da je prijelomna crta negdje između 100 i 250 jedinki. Mjerenja za populacije manje od 250 jedinki jasno pokazuju lošije rezultate, odnosno da je veličina populacije premala. Ostala mjerenja za populacije preko 250 jedinki pokazuju da nema neke značajnije razlike među njima. Konačno, za veličinu populacije je odabrano 500 jedinki jer ima najmanje odstupajućih vrijednosti (slika 9) i jer ima najmanju vrijednost srednjeg elementa (engl. *Mean*) (slika 8). S tim odabirom se mijenja i uvjet zaustavljanja zbog toga da se sačuva konstantni broj evaluacija. Novi najveći broj generacija prije zaustavljanja algoritma je 100 generacija.



Slika 8 Prikaz rezultata za veličinu populacije, bez odstupajućih vrijednosti



Slika 9 Prikaz rezultata za veličinu populacije, s odstupajućim vrijednostima

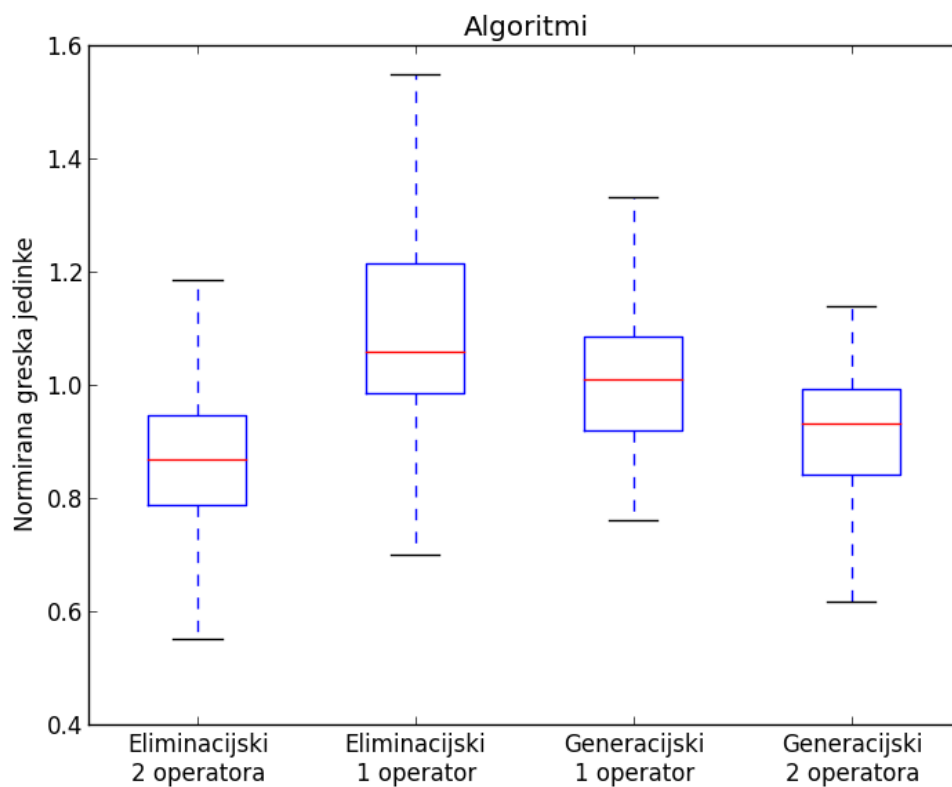
5.1.2. Evolucijski algoritam

Određivanje evolucijskog algoritma teško se može nazvati parametrom, ali svejedno su napravljena mjerenja nad različitim vrstama algoritama. To je napravljeno da se odredi da li sam algoritam utječe na rezultate i kako. Mjerenja su napravljena nad četiri različita algoritma objašnjena u poglavlju 4.1 - Detalji implementacije GP-a. Za sve algoritme, osim za eliminacijski s dva operatora, su napravljena dodatna testiranja kako bi se našao najbolja vjerojatnost križanja. Inicijalni parametri su prikazani u tablici 3, s razlikom da je za veličinu populacije uzeto 500 jedinki, a za najveći broj generacija 100 generacija. Nakon što je za svaki algoritam odabrana najbolja vjerojatnost križanja, četiri odabrana algoritma su uspoređena te su rezultati prikazani u tablici 5.

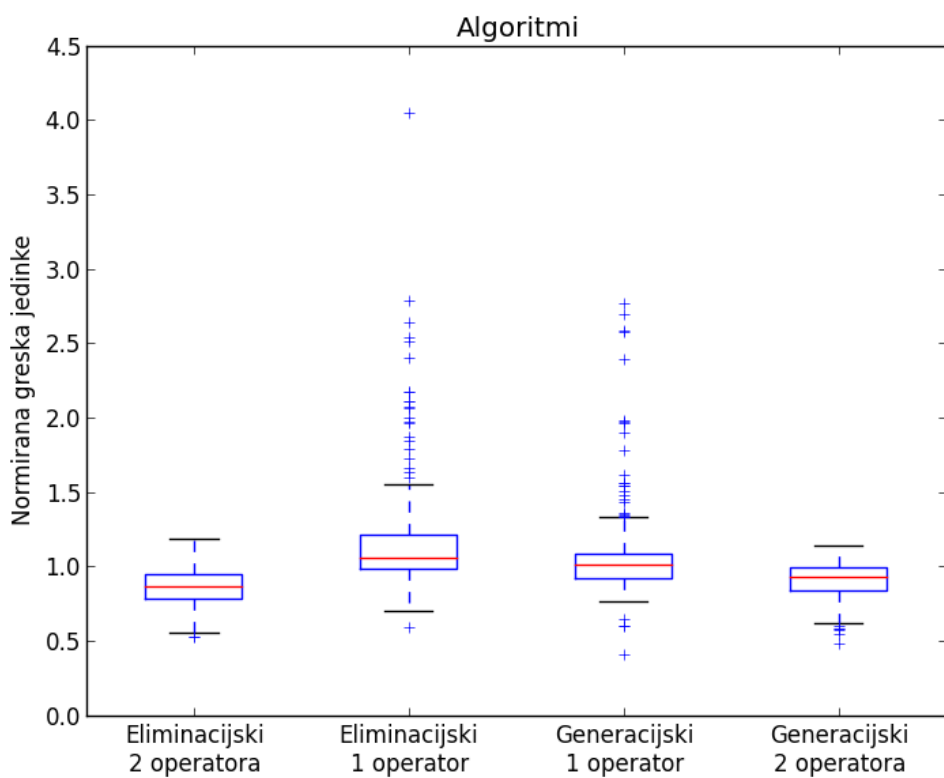
Tablica 5 Rezultati za izbor algoritma

Algoritam	Eliminacijski 2 operatora	Eliminacijski 1 operator	Generacijski 1 operator	Generacijski 2 operatora
<i>Maksimum</i>	1.183651897	4.051531145	2.765987920	1.137065983
<i>Prosjek</i>	0.861857734	1.164906759	1.066222302	0.907013205
<i>Minimum</i>	0.522363010	0.586062764	0.410511182	0.483743596

Radi boljeg pregleda rezultata, napravljen je *boxplot* prikaz rezultata koji je prikazan na slici 10 i 11. Razlika između slika je u tome prikazuju li odstupajuće podatke ili ne. Na prvoj slici nisu prikazani odstupajući podaci te se na njoj može primijetiti trend algoritama. Može se primijetiti kako su algoritmi s dva operatora bolji od algoritama s jednim operatorom. Također, ne postoje neke značajnije razlike između eliminacijskog i generacijskog algoritma. Kao rezultat ovog mjerenja, uzet je eliminacijski algoritam s dva operatora. Razlog jest taj da je to najjednostavnija implementacija koja se može najlakše paralelizirati, prilikom selekcije napravi se jedan turnir te je pokazao jedne od boljih rezultata, s jako malim brojem odstupajućih vrijednosti.



Slika 10 Prikaz rezultata za izbor algoritma, bez odstupajućih vrijednosti



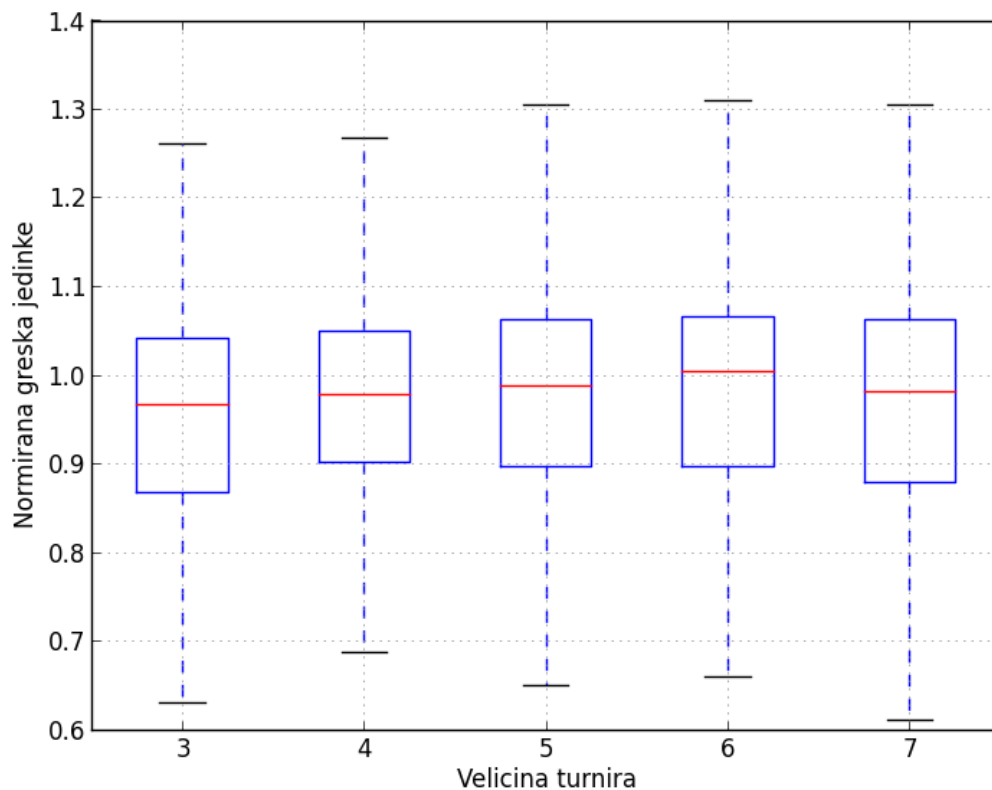
Slika 11 Prikaz rezultata za izbor algoritma, s odstupajućim vrijednostima

5.1.3. Veličina turnira

Veličina turnira je parametar koji određuje koliko je jak selekcijski pritisak. S većom veličinom turnira raste selekcijski pritisak i manja je vjerojatnost da će se odabrati lošije jedinke. Prevelik selekcijski pritisak može uzrokovati ranu konvergenciju, u lokalni optimum. Inicijalni parametri su prikazani u tablici 3, s razlikom da je za veličinu populacije uzeto 500 jedinki, a za najveći broj generacija 100 generacija. Rezultati su prikazani u tablici 6.

Tablica 6 Iznos normiranih pogreški za izbor veličine populacije

Veličina turnira	3	4	5	6	7
<i>Maksimum</i>	3.366964	11.45243	5.043010	2.220169	4.036733
<i>Prosjek</i>	0.971774	1.029924	1.012433	0.995399	0.990469
<i>Minimum</i>	0.592312	0.572855	0.506406	0.559750	0.610662

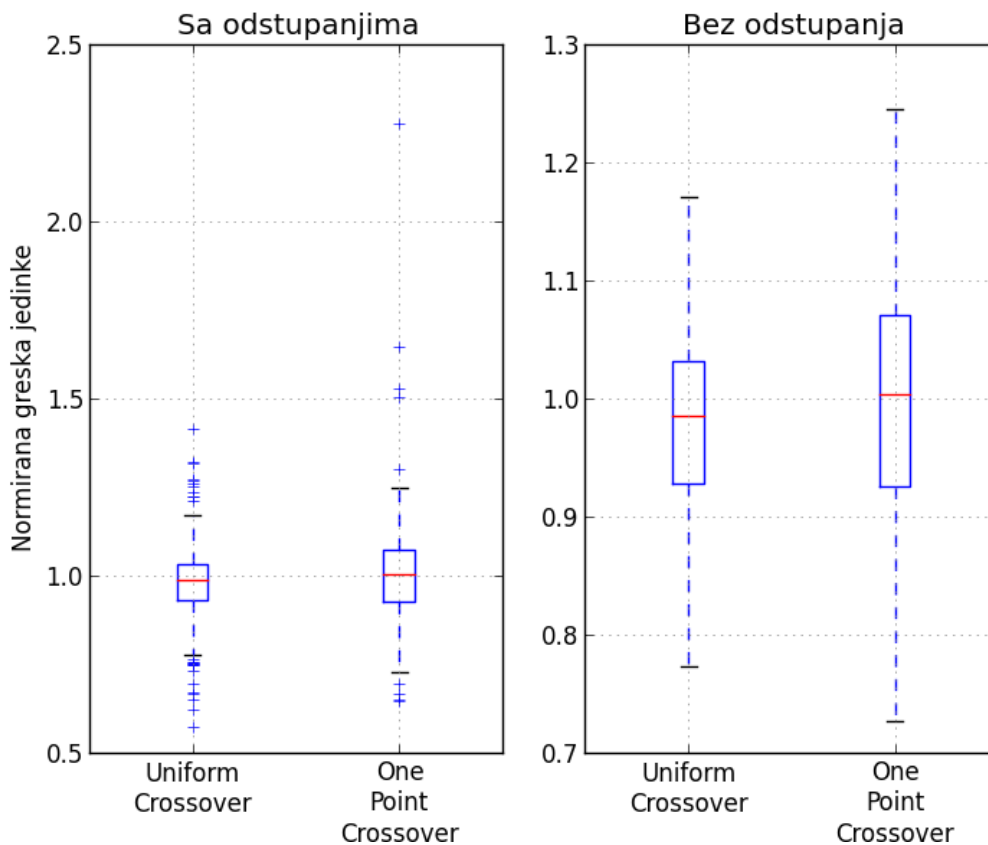


Slika 12 Prikaz rezultata za veličine turnira, bez odstupajućih vrijednosti

Na temelju ovog mjerenja, za veličinu turnira je ostavljeno 3 jedinke. Razlog tomu je što ima najmanju srednju pogrešku, greške za veće turnire imaju tendenciju rasta (Slika 12) te se smanjuje selekcijski pritisak. Mjerenja nisu napravljena za turnir od dvije jedinke jer za reprodukciju trebaju najmanje tri, dvije za križanje i jedna za eliminaciju iz populacije.

5.1.4. Križanje

Za potrebe mjerenja utjecaja operatora križanja, implementirana su dva različita operatora: križanje s jednom točkom prekida i uniformno križanje. Detalji su objašnjeni u poglavlju 4.1 Detalji implementacije GP-a. Zbog prirode implementiranog algoritma, operatori križanja nemaju vjerojatnost križanja već se nakon svake selekcije odabrane jedinke križaju. Inicijalni parametri su prikazani u tablici 3, s razlikom da je za veličinu populacije uzeto 500 jedinki, a za najveći broj generacija 100 generacija. Rezultati mjerenja su prikazani u tablici 7.



Slika 13 Prikaz rezultata za izbor operatora križanja, s i bez odstupajućih vrijednosti

Tablica 7 Prikaz normirane greške za operatore križanja

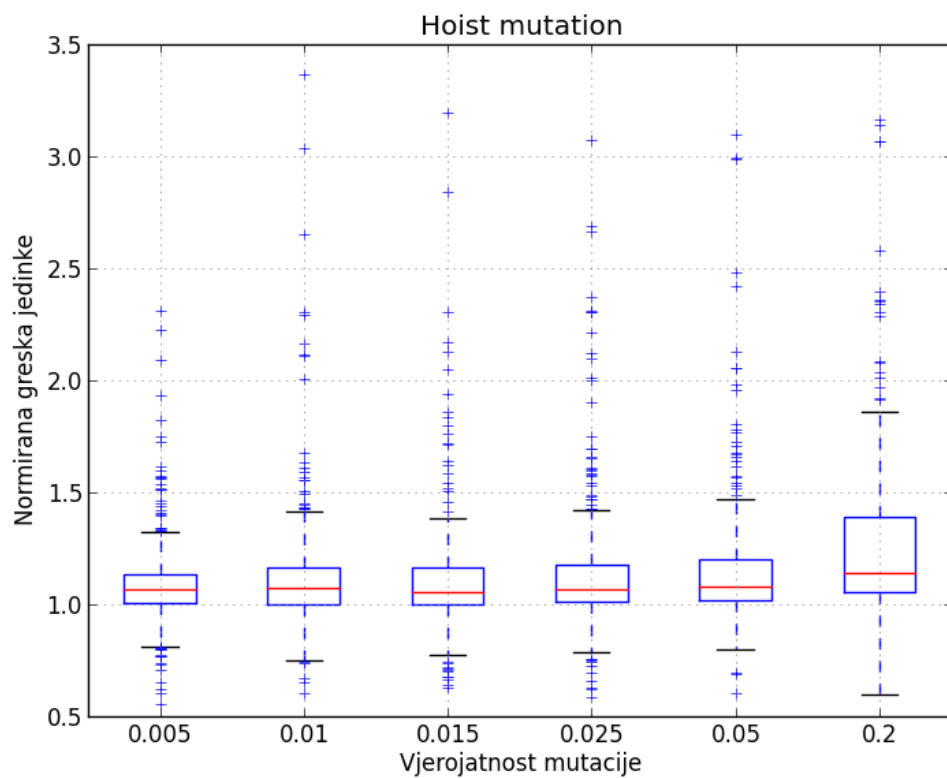
Križanje	Uniformno	S jednom točkom
<i>Maksimum</i>	1.413126643	3.671035391
<i>Prosjek</i>	0.979848474	1.020151526
<i>Minimum</i>	0.572592254	0.645803459

Na temelju rezultata, za operator križanja je uzeto uniformno križanje. Nedostatak uniformnog križanja jest to što ima malo veću složenost (mora proći kroz cijelo zajedničko područje), ali je pokazalo malo bolje rezultate u mjerenjima što se i vidi na slici 13. U tablici 7 se vidi da je uniformno križanje u svim aspektima pokazalo bolje rezultate.

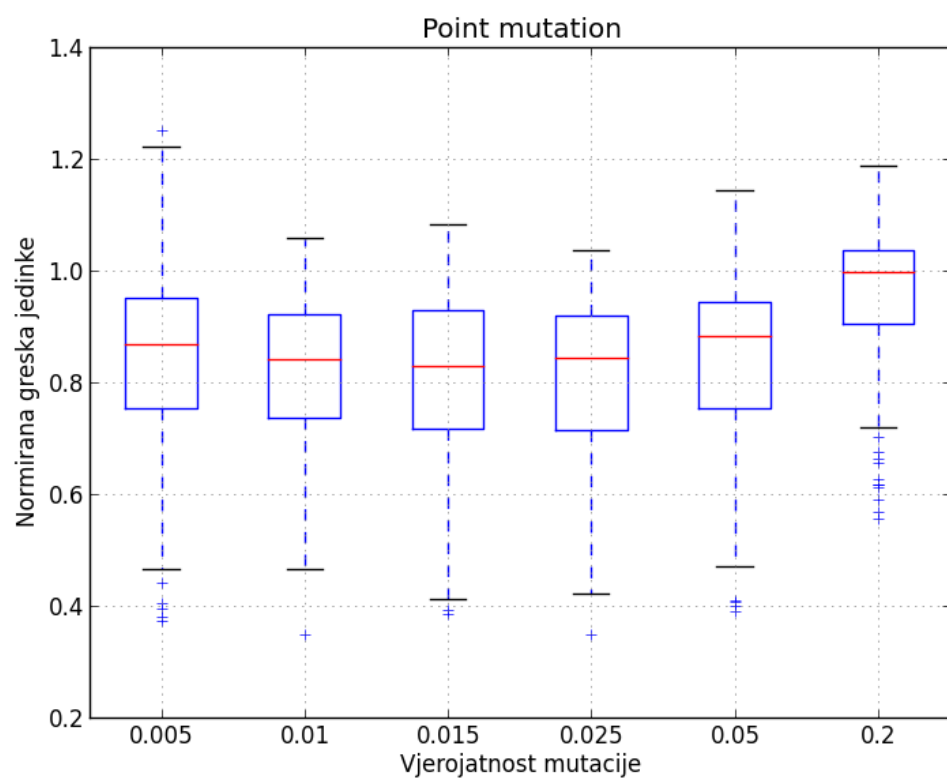
5.1.5. Mutacija

Operator mutacije zahtjeva malo veću složenost prilikom mjerenja jer osim izbora samog operatora mutacije postoji i izbor vjerojatnosti mutacije. Ta dva parametra su povezana te nije pametno mjeriti jedan bez drugoga. Pogotovo jer su implementirani operatori mutacije drugačijeg tipa: jedan mutira čvor, a drugi cijelo (pod)stablo. Zbog toga je za svaki operator prvo napravljeno mjerenje po vjerojatnostima mutacije, a tek nakon toga su operatori međusobno uspoređeni. Inicijalni parametri su prikazani u tablici 3, s razlikom da je za veličinu populacije uzeto 500 jedinki, za najveći broj generacija 100 generacija i za operator križanja uzeto uniformno križanje.

Na slici 14 i 15 se može primijetiti velika razlika u uspješnosti operatora te je *point* mutacija siguran pobjednik. *Point* mutacija ima manju srednju grešku za sve vjerojatnosti i ima puno manje odstupajućih vrijednosti. Ostavlja se jedino pitanje koliko vjerojatnost mutacije uzeti. Za vjerojatnosti mutacije 1%, 1.5% i 2.5% nema neke značajnije razlike i može se odabrati bilo koja od njih. Kao zaključak ovog mjerenja, za operator mutacije je uzeta *point* mutacija s vjerojatnošću mutacije čvora od 1%.



Slika 14 Prikaz rezultata za vjerojatnost mutacije za *hoist* mutaciju



Slika 15 Prikaz rezultata za vjerojatnost mutacije za *point* mutaciju

5.1.6. Uvjet zaustavljanja

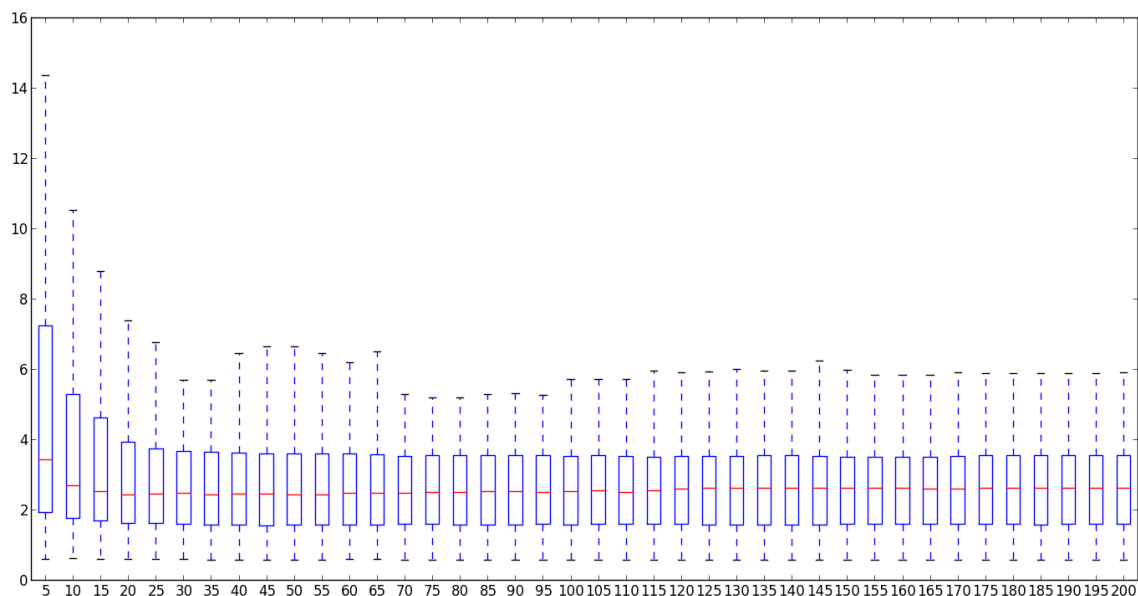
Do sada su podešeni parametri genetskog programiranja te se može reći kako su prediktori malo poboljšani. Zbog toga i zbog ubrzanja algoritma, napravljeno je mjerenje za najveći broj generacija koji algoritam odradi. Za razliku od prijašnjih parametara, ovaj parametar se ne može podešavati na isti način zbog toga što će algoritam davati sve bolja rješenja što ga se dulje pusti da radi. Najveći broj generacija će se ispitati metodom unakrsne provjere. Nakon što se prediktor nauči na skupu za učenje, njegova greška se provjerava na skupu za provjeru. Taj postupak se ponavlja nakon svake izvršene generacije. Inicijalni parametri za provjeru najvećeg broja generacija su prikazani u tablici 8. Osim što su parametri prilagođeni prijašnjim mjerenjima, razlika između tablica 8 i 3 je u procjenitelju greške. Nadalje će se sva mjerenja provoditi s MAPE procjeniteljem jer je učestaliji kod mjerenja greške za prediktore kratkoročne potrošnje energije.

Tablica 8 Poboljšani inicijalni parametri s unakrsnom provjerom

	Parametri	Inicijalne vrijednosti
Algoritam	Ime (vrsta) algoritma	Eliminacijski s dva operatora
	Veličina populacije	500
	Veličina turnira	3
Uvjet zaustavljanja	Broj generacija	200
Stablo	Najveća dubina	8
	Najmanja dubina	3
	Funkcijski čvorovi	+ - / * ifRadniDan
Križanje	Ime (vrsta) križanja	Uniformno križanje
Mutacija	Ime (vrsta) mutacije	<i>Point</i> mutacija
	Vjerojatnost mutacije (čvora)	0.01
	Dodatno mutiranje konstante	Da
Evaluacija	Procjenitelj pogreške	MAPE
	Broj prijašnjih mjerenja	5
Unakrsna validacija	Rotiraj preklope	ne
	Napravi skup za evaluaciju	ne
	Broj preklopa	5

Kako rezultata ima puno, prikazani su u tablicama i na slikama. Na Slika 16 se može primijetiti trend pogreške na skupu za učenje. Po pravilu unakrsne provjere, trebao bi se uzeti broj generacije za koji je greška minimalna. Minimum se nalazi negdje između 20-te i 70-te generacije, ali se na Slika 16 ne može razlučiti veća razlika između tih generacija. Zbog toga su detalji rezultata prikazani u tablici 9. Zbog toga što se ne mogu razlučiti značajne razlike između 20-te i 70-te generacije, za najveći broj generacija će se uzeti 50 generacija. Medijan poprima najmanju vrijednost upravo u 50. generaciji.

Još jedan zanimljiv fenomen se može primijetiti na slici 17. Slika prikazuje istu raspodjelu podataka, jedino što prikazuje i odstupajuće vrijednosti. Zgodno je primijetiti na slici lokalne optimume u kojim prediktori znaju zapeti. Jedan od očitijih lokalnih optimuma je u vrijednosti oko 45. Ta greška se pojavljuje jedino kod prediktora koji predviđa za 16h i taj prediktor do sada nije uspio spustiti vrijednost te greške.

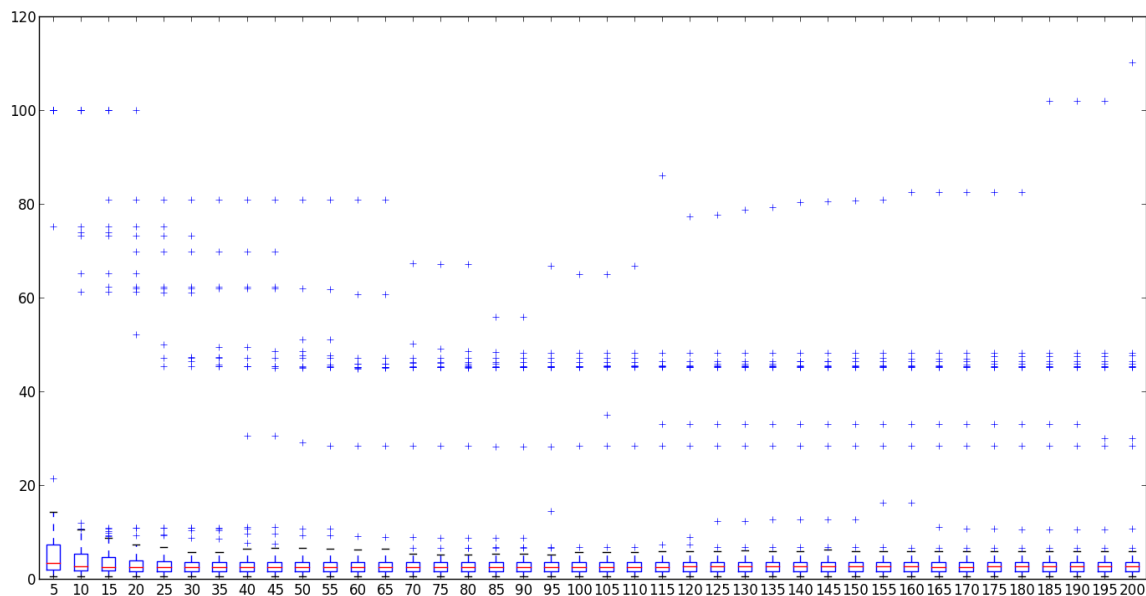


Slika 16 Greška na skupu za provjeru kroz generacije, bez odstupajućih vrijednosti

Tablica 9 Rezultati na skupu za učenje kroz generacije

br. gen.	5	10	15	20	25	30	35	40	45	50
<i>max</i>	100.017	100.016	100.016	100.016	80.8355	80.8355	80.8355	80.8355	80.8355	80.8355
<i>gg</i>	14.3571	10.5163	8.7817	7.3881	6.7709	5.7001	5.7001	6.4605	6.6317	6.6317
<i>q1</i>	7.2364	5.28115	4.6225	3.93713	3.72868	3.66055	3.63383	3.61245	3.5986	3.5955
<i>med</i>	3.43105	2.693	2.5233	2.4387	2.4593	2.47165	2.43565	2.45335	2.44765	2.41715
<i>q3</i>	1.93478	1.77243	1.6981	1.61595	1.6112	1.58955	1.57873	1.57213	1.55833	1.56548
<i>dg</i>	0.5943	0.6145	0.595	0.5917	0.5917	0.5968	0.5801	0.5734	0.5774	0.5743
br. gen.	55	60	65	70	75	80	85	90	95	100
<i>max</i>	80.8355	80.8355	80.835	67.2819	67.2279	67.2279	55.9035	55.9035	66.7214	65.074
<i>gg</i>	6.4472	6.182	6.5019	5.2813	5.1848	5.1848	5.2837	5.3004	5.2501	5.7149
<i>q3</i>	3.5854	3.59005	3.5798	3.53155	3.55643	3.55855	3.54265	3.5451	3.53585	3.5248
<i>med</i>	2.42925	2.4689	2.4842	2.48425	2.5078	2.508	2.51325	2.5317	2.5045	2.5162
<i>q1</i>	1.5639	1.57633	1.5709	1.58565	1.58458	1.5729	1.57463	1.5739	1.58678	1.57623
<i>dg</i>	0.5743	0.5892	0.5885	0.5817	0.5811	0.5811	0.5811	0.5811	0.5811	0.5811
br. gen.	105	110	115	120	125	130	135	140	145	150
<i>max</i>	65.074	66.8373	86.156	77.2534	77.5944	78.8019	79.3296	80.3312	80.4608	80.8018
<i>gg</i>	5.7149	5.7149	5.9444	5.9011	5.938	5.9884	5.9621	5.9621	6.2382	5.9654
<i>q3</i>	3.55048	3.5258	3.5058	3.51693	3.5198	3.5134	3.53823	3.5561	3.53173	3.4889
<i>med</i>	2.55275	2.5087	2.5408	2.59555	2.62305	2.6218	2.62995	2.61805	2.6182	2.62915
<i>q1</i>	1.5865	1.588	1.5879	1.58703	1.58028	1.58068	1.58068	1.57593	1.57593	1.58445
<i>dg</i>	0.5808	0.5808	0.5808	0.572	0.572	0.5726	0.5726	0.5721	0.5723	0.5723
br. gen.	155	160	165	170	175	180	185	190	195	200
<i>max</i>	80.9404	82.5834	82.583	82.5834	82.5834	82.5834	101.925	101.925	101.925	110.214
<i>gg</i>	5.8339	5.8339	5.8438	5.8995	5.8731	5.8731	5.87310	5.8897	5.8897	5.9159
<i>q3</i>	3.50565	3.50623	3.4959	3.51753	3.54963	3.53908	3.53688	3.55038	3.55247	3.55332
<i>med</i>	2.62915	2.6279	2.5869	2.5869	2.6218	2.60745	2.60745	2.62695	2.62695	2.62855
<i>q1</i>	1.5874	1.59105	1.5917	1.5957	1.5957	1.59283	1.57088	1.60045	1.59587	1.60247
<i>dg</i>	0.5726	0.5727	0.5656	0.5656	0.5656	0.5727	0.5730	0.5730	0.5729	0.5729

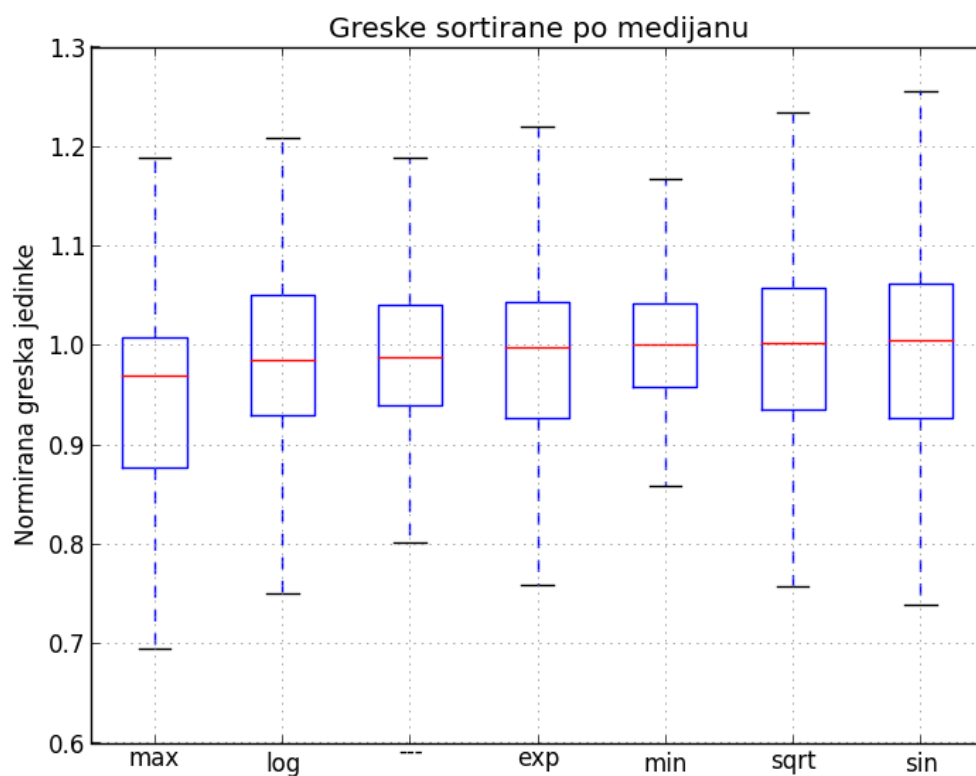
Tablica 9 je prikazana u četiri dijela zato da podaci stanu na stranicu. U tablici su po redovima prikazani: broj generacije, najveća vrijednost, gornja granica *boxplota*, gornji ili treći kvartil, medijan, donji ili prvi kvartil i donja granica *boxplota*. Najmanja vrijednost nije prikazana jer ima iste vrijednosti kao i donji kvartil. Tablica je prikazana da se mogu bolje pregledati rezultati mjerenja.



Slika 17 Greška na skupu za provjeru kroz generacije, s odstupajućim vrijednostima

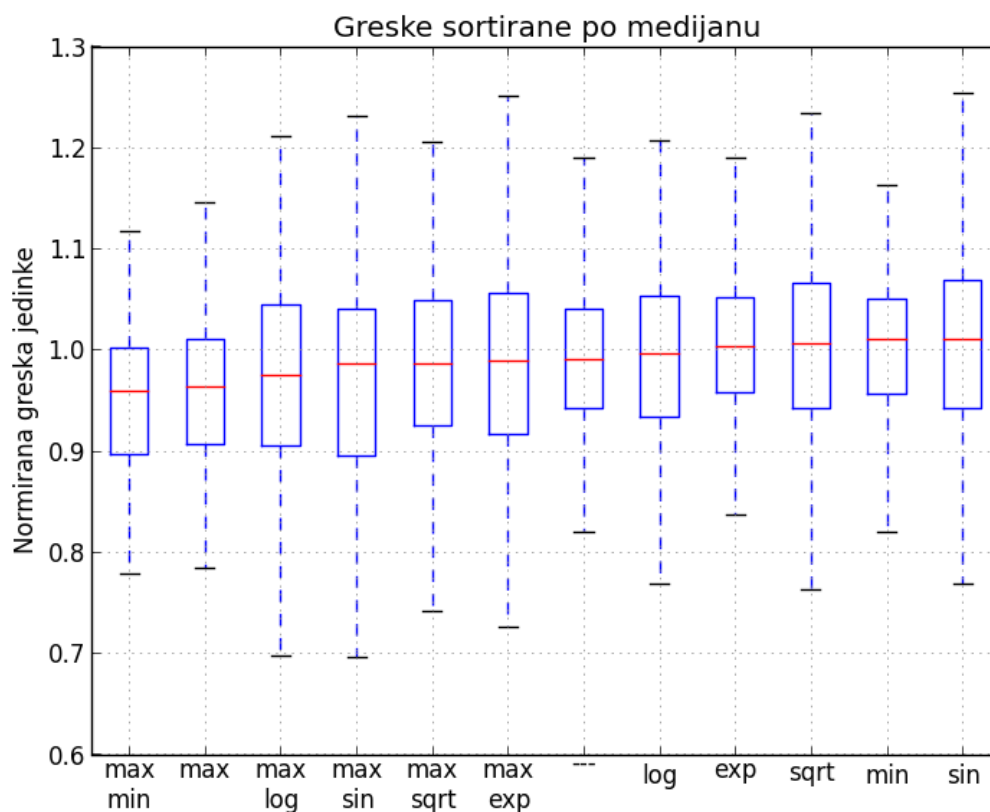
5.1.7. Funkcijski čvorovi

Funkcijski čvorovi su veoma bitni jer uz pomoć njih se aproksimiraju podaci. Ako postoji premalo „dobrih“ funkcijskih čvorova, algoritam se neće moći točno prilagoditi podacima. Ako postoji previše funkcijskih čvorova, onda se pojavljuje velika mogućnost da se algoritam prenauči. Pojam „dobar“ funkcijski čvor je 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 čak i pokvariti algoritam. U mjerenjima su korištene postavke iz tablice 8, ali bez unakrsne provjere i s uvjetom zaustavljanja na 50 generacija. Ideja je dodavati nove čvorove te vidjeti da li utječu na rezultat.



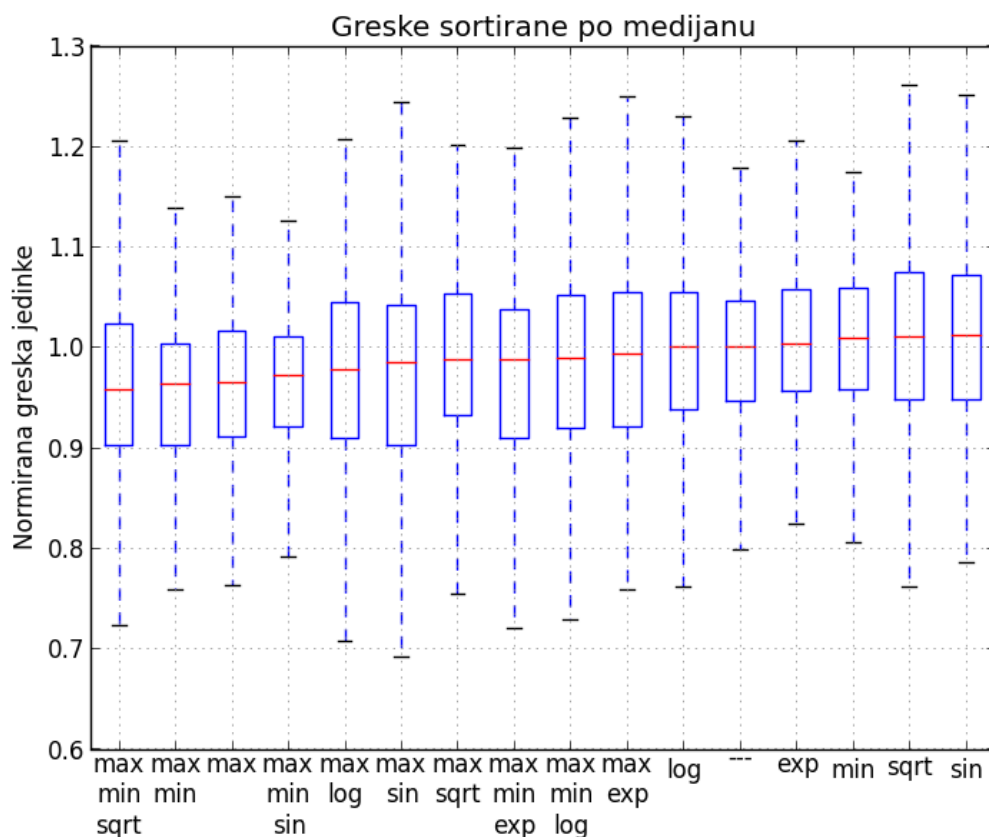
Slika 18 Prikaz rezultata za nove funkcijske čvorove, bez odstupajućih vrijednosti

Prvi rezultati mjerenja su prikazani na slici 18 na kojoj se jasno ističe funkcija maksimuma. Treće mjerenje od početka je mjerenje s osnovnim skupom čvorova, bez dodatnih čvorova, radi usporedbe. Zbog dobrih rezultata za maksimum (75% vrijednosti je ispod prosjeka), napravljen je i drugi krug mjerenja. U drugom krugu su dodani svi parovi čvorova s time da je u tom paru uvijek maksimum. Ideja je vidjeti hoće li biti poboljšanja ako se čvoru maksimum doda još jedan funkcijski čvor. Rezultati su prikazani na slici 19.



Slika 19 Prikaz rezultata za parove funkcijskih čvorova, bez odstupajućih vrijednosti

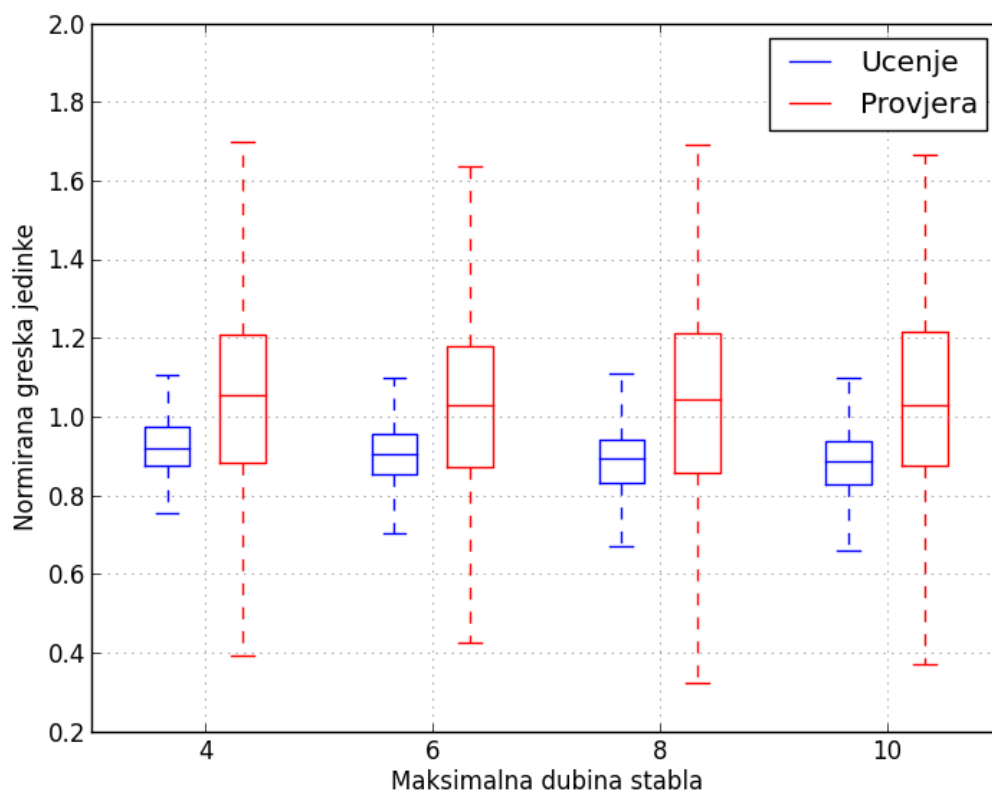
Na slici se može primijetiti zanimljiva stvar, greške za sve parove funkcija se nalaze između grešaka pojedinačnih funkcija. Jedino je za par minimum/maksimum greška manja nego od grešaka njegovih pojedinačnih funkcija. Zbog toga je napravljen i treći krug testiranja u kojemu je paru minimum, maksimum dodan po još jedan funkcijski čvor. Rezultati trećeg kruga su prikazani na slici 20. Prema slici, čini se kao da se dodavanjem funkcije korijena smanjuje greška, no greška se ne smanjuje dovoljno. Također, dodavanjem previše čvorova se povećava složenost GP-a. Zbog svega toga, kao rezultat ovog mjerenja, osnovnom skupu funkcijskih čvorova su dodani čvorovi minimuma i maksimuma.



Slika 20 Prikaz rezultata za trojke funkcijskih čvorova, bez odstupajućih vrijednosti

5.1.8. Dubina stabla

Parametar dubine stabla su zapravo dva parametra: najmanja dubina stabla i najveća dubina stabla. Ti parametri se koriste prilikom generiranja originalne populacije i mogu znatno utjecati na brzinu i uspješnost izvođenja genetskog algoritma. Za mjerenje su uzete četiri vrijednosti za najmanju dubinu i četiri vrijednosti za najveću dubinu te su napravljene sve moguće kombinacije, ukupno 14 mjerenja. Dubina stabla je definitivno parametar koji se mora podešavati uz unakrsnu validaciju jer ako se stavi jako velika dubina stabla, algoritam će se previše prilagoditi podacima na kojima uči. U mjerenjima su korištene postavke iz tablice 8, ali s uvjetom zaustavljanja na 50 generacija i s dodanim funkcijskim čvorovima minimum i maksimum. Prvi dio rezultata je prikazan na dvije odvojene slike, jedna za najmanje dubine, jedna za najveće dubine.

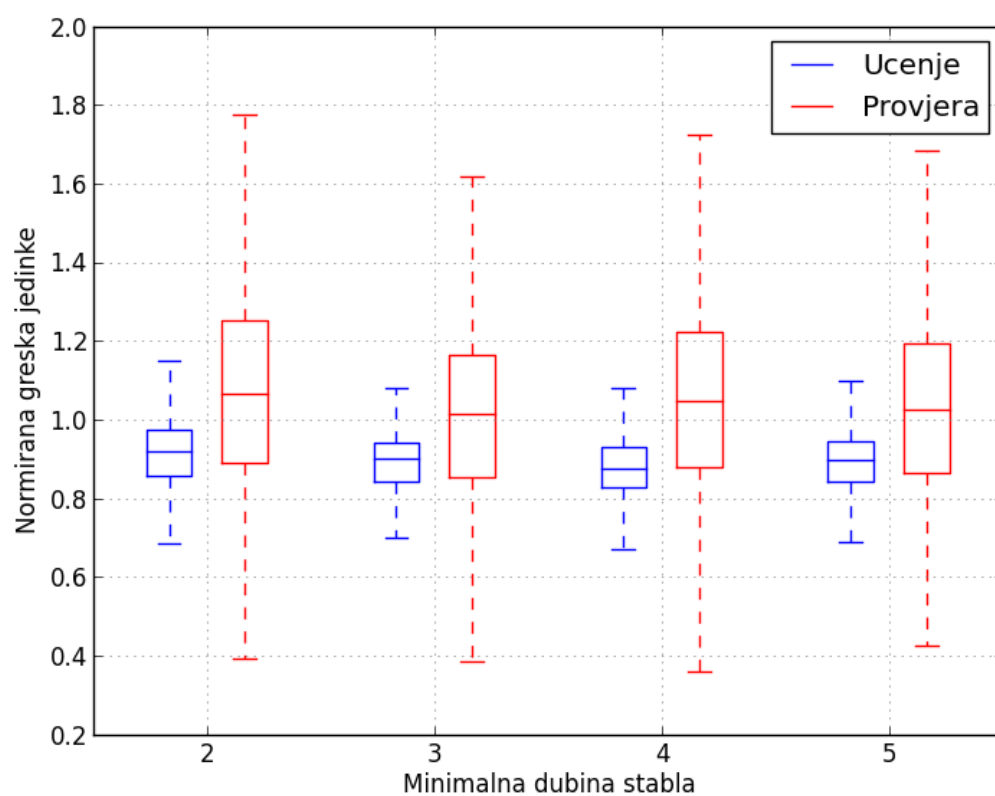


Slika 21 Prikaz rezultata za najveću dubinu stabla, bez odstupajućih vrijednosti

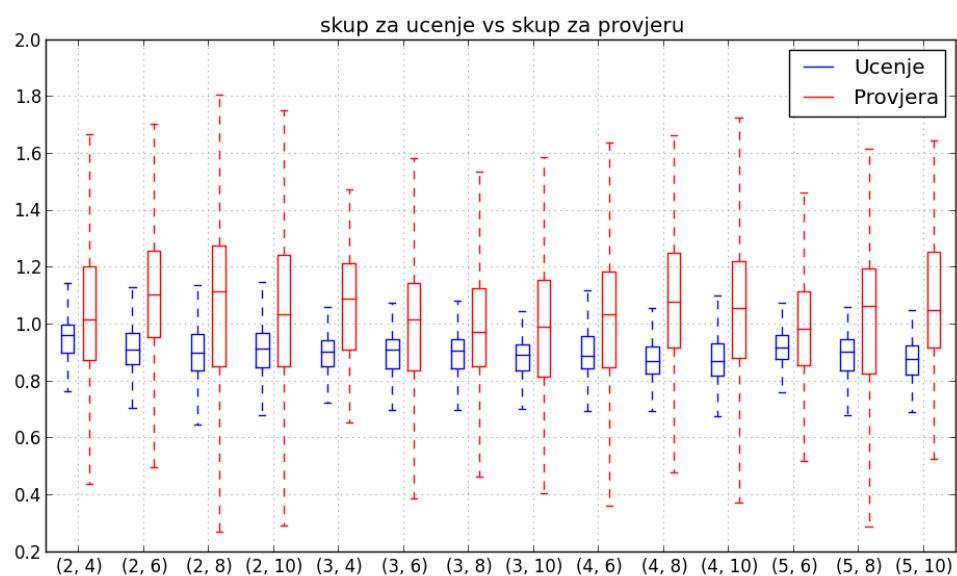
Na slici 21 su prikazani rezultati za određivanje najveće dubine. Za očekivati je da će greška generalno padati s većom dopuštenom dubinom, što se i vidi na skupu za učenje. S druge strane, na skupu za provjeru jedne od boljih rezultata je pokazala kada je najveća dubina 6, iako ni 8 nije daleko.

Na slici 22 su prikazani rezultati za određivanje najmanje dubine. Tu nije bilo za očekivati da će greška padati što je najmanja dubina veća, jer se tada „forsira“ stablo da mora imati određenu dubinu. To se i vidi kada je vrijednost najmanje dubine 5, za koju je greška na skupu za učenje veća nego na prijašnjim mjeranjima. Ako se ta dubina izbacila iz razmatranja jer je prevelika, od ostalih dubina je najbolja dubina 3 jer ima najmanju grešku na skupu za provjeru.

Slika 23 je prikazana radi bolje usporedbe mjerenja za pojedini par dubina. Trenutno je favorit par (3,6) ili (3,8) te se usporedbom ta dva mjerenja dolazi do zaključka da (3,8) pokazuje bolje rezultate te će se koristiti za daljnja mjerenja.



Slika 22 Prikaz rezultata za najmanju dubinu stabla, bez odstupajućih vrijednosti



Slika 23 Prikaz rezultata pojedinačnih mjerenja za dubinu stabla

5.2. Određivanje ulaznih vrijednosti

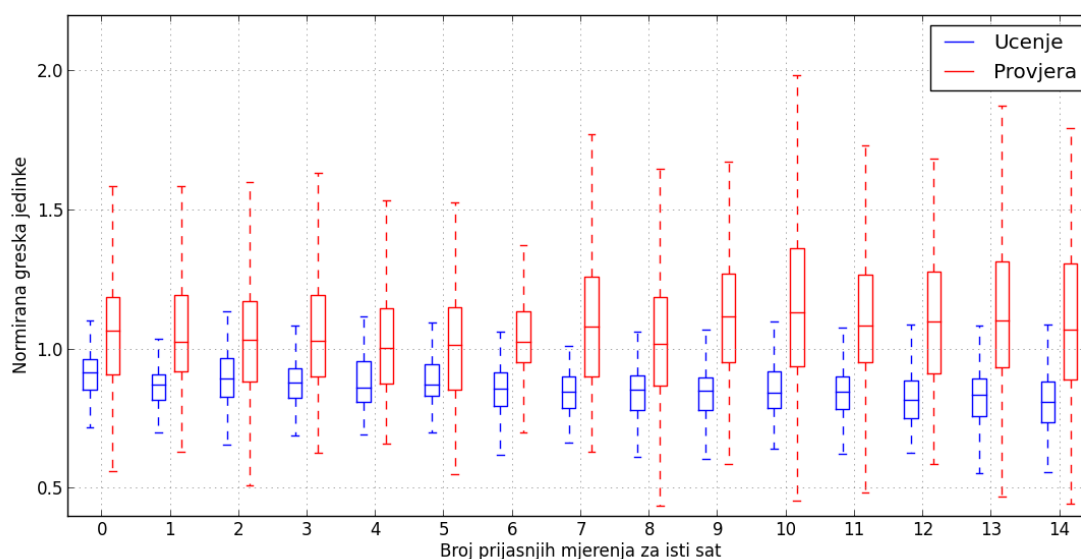
Određivanje ulaznih varijabli je bitan proces jer iznimno utječe na rezultate prediktora. Kod izbora ulaznih varijabli je bitno pravilo „zlatne sredine“: ni previše ni premalo. Ako se prediktoru da premalo ulaznih varijabli, nema na dovoljno podataka za točno predviđanje. Ako se pak prediktoru da previše varijable, čime je i vjerojatnije da su varijable zavise, prediktor će se previše prilagoditi tim ulaznim podacima i prenaučiti. U daljnjem tekstu će biti prikazani rezultati mjerenja pojedinih ulaznih varijabli. Parametri učenja genetskog programiranja su prilagođeni dosadašnjim mjerenjima i rezultatima te su dani u tablici 10. Ideja ovih mjerenja je da se odredi koje ulazne varijable pomažu pri učenju, a koje ne.

Tablica 10 Prikaz parametara za učenje GP-a kod određivanja ulaznih varijabli

	Parametri	Inicijalne vrijednosti
Algoritam	Ime (vrsta) algoritma	Eliminacijski s dva operatora
	Veličina populacije	500
	Veličina turnira	3
Uvjet zaustavljanja	Broj generacija	200
Stablo	Najveća dubina	8
	Najmanja dubina	3
	Funkcijski čvorovi	+ - / * ifRadniDan min max
Križanje	Ime (vrsta) križanja	Uniformno križanje
Mutacija	Ime (vrsta) mutacije	<i>Point</i> mutacija
	Vjerojatnost mutacije (čvora)	0.01
	Dodatno mutiranje konstante	Da
Evaluacija	Procjenitelj pogreške	MAPE
	Broj prijašnjih mjerenja	5
Unakrsna validacija	Rotiraj preklope	da
	Napravi skup za evaluaciju	ne
	Broj preklopa	5

5.2.1. Potrošnje u prijašnjim danima za isti sat

Ideja ovog mjerenja jest da potrošnja u određenom satu ima svoj vlastiti trend te da je moguće na osnovu potrošnji za određeni sat u prijašnjim danima odrediti potrošnju za ovaj dan. Zbog toga se taj skup ulaznih parametara ispituje zasebno. U mjerenjima su korištene postavke iz tablice 10 Tablica 8, ali se mijenja broj prijašnjih mjerenja u rasponu od 0 do 14 prethodnih dana.



Slika 24 Prikaz rezultata za različiti broj mjerenja istog sata u prijašnjim danima

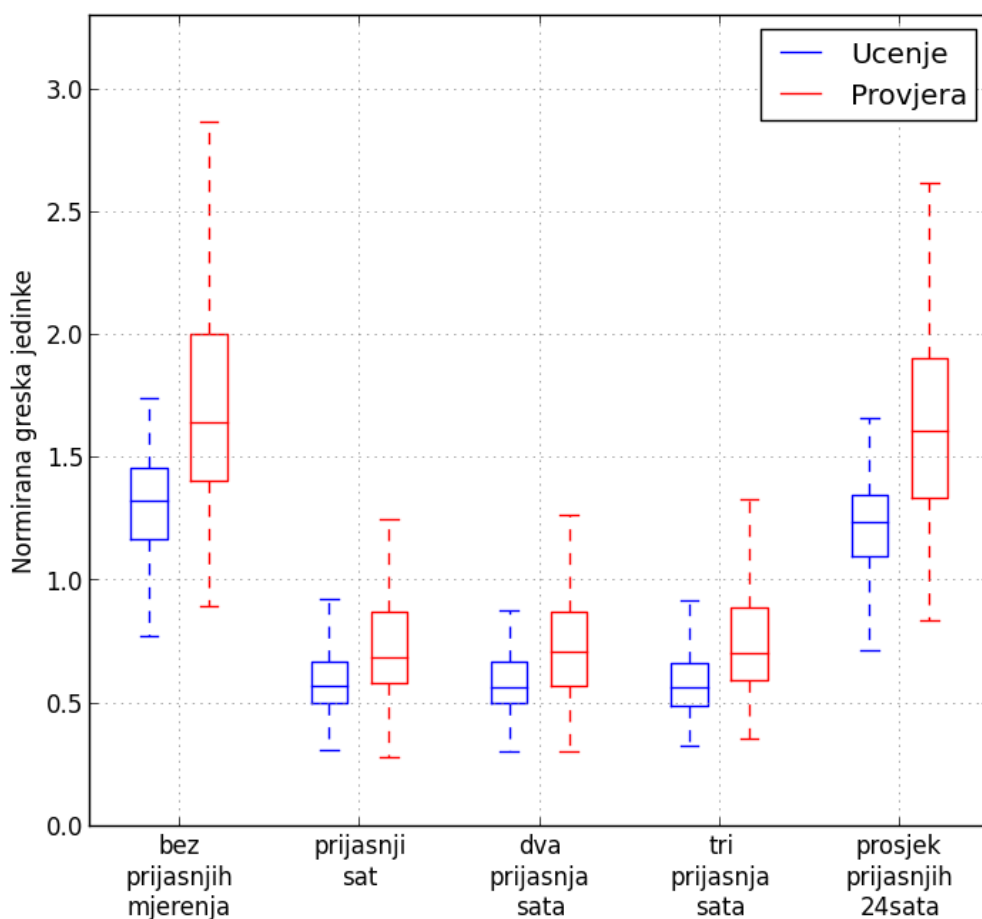
Rezultat mjerenja je prikazan na slici 24 na kojoj su prikazane greške na skupu za učenje (plava boja) i greške na skupu za provjeru (crvena boja) za različiti broj prijašnjih mjerenja istog sata. Greška na skupu za učenje pada s brojem prijašnjih mjerenja, kao što je i očekivano. S druge strane, greška na skupu za provjeru ne pada konstantno već postiže minimum između četiri i šest prethodnih mjerenja.

Genetsko programiranje prilikom učenja ne mora uzeti sve varijable koje ima na raspolaganju. Analizom jedinki koje su dobivene u rezultatima, može se primijetiti da naučeni model gotovo nikada ne koristi sve ulazne varijable već da uzme jednu ili nekoliko njih. Zbog toga genetsko programiranje neće biti preopterećeno ako ima više ulaznih varijabli. Uzimajući u obzir tu činjenicu kao i rezultate mjerenja prikazane na prijašnjoj slici,

uzeto je šest mjerenja za broj prijašnjih mjerenja istog sata. Tako se prediktoru daje dovoljno informacija da nauči i sam odabere koje su mu bitne.

5.2.2. Potrošnje iz prijašnjih 24 sata

Uz varijable koje opisuju prijašnji trend potrošnje, bitne su i varijable koje mogu odrediti kakva bi mogla biti potrošnja u trenutnom danu. Zbog toga je bitno imati neku mjeru potrošnje u proteklih 24 sata. U mjerenjima su korištene postavke iz tablice 10 Tablica 8, ali brojem prijašnjih mjerenja od 6 dana.



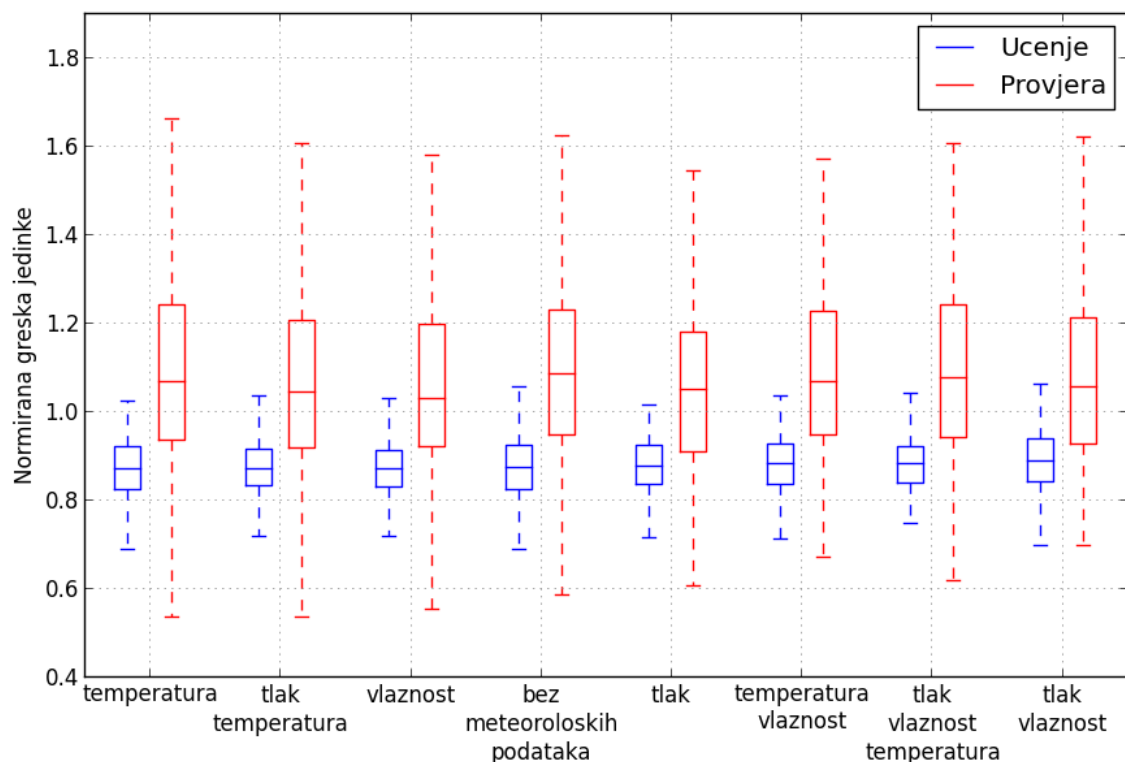
Slika 25 Prikaz rezultata za različite varijable iz prijašnjih 24 sata

Rezultati mjerenja su prikazani na slici 25 i okvirno se mogu podijeliti u dvije skupine: dobre i loše rezultate. Dobre rezultate daju ulazne varijable koje imaju informaciju o neposrednim vrijednostima potrošnje, odnosno vrijednostima potrošnje u satu/satima koji su

neposredno prije sata za koji se predviđa. S druge strane, dobiju se loši rezultati ako ulazne varijable nemaju nikakvu informaciju o potrošnji u prijašnjih 24 sata ili ako je ta informacija preopćenita, npr. prosjek. Očito je da potrošnja u trenutnom satu jako ovisi o potrošnji u prijašnjem satu, koja ovisi o potrošnji u satu prije, itd. Zbog toga će se za ulaznu varijablu dodati potrošnja u prijašnjem satu.

5.2.3. Meteorološke prilike

Podaci o meteorološkim prilikama mogu biti dragocjeni i opisivati utjecaj okoline na količinu potrošnje električne energije. Najočitija vremenska prilika koja utječe na potrošnju je temperatura zraka. U ovom mjerenju su korištene i druge prilike kao što su vlažnost i tlak zraka kako bi se odredilo da li se dodavanjem tih varijabli može poboljšati predviđanje potrošnje. U mjerenjima su korištene postavke iz tablice 10 Tablica 8, ali brojem prijašnjih mjerenja od 6 dana. Osim onih koje se ispituju, ulazne varijable su: radni dan, mjesec, potrošnja u prijašnjem satu i potrošnja u istom satu u 6 prijašnjih dana.



Slika 26 Prikaz rezultata za različite meteorološke ulazne varijable

Na slici 26 su prikazani rezultati za različite meteorološke varijable. Rezultati su poredani po medijanu skupa za učenje, no vrijednosti su toliko slične da se razlike uopće ne primjećuju. Čak nema ni značajnijih razlika na skupu za provjeru. Jedan od uzroka takvih rezultata bi mogao biti u činjenici da prediktor dobiva puno prijašnjih potrošnji koje su dovoljne za predviđanje. Rezultat ovog mjerenja jest da varijable koje opisuju meteorološke prilike u satu za koji se predviđa nisu potrebne te se u daljnjem predviđanju neće uzimati kao ulazne varijable.

5.3. Usporedba s drugim algoritmima

Dosadašnjim mjerenjima se određivala relativna uspješnost implementiranog prediktora i to u odnosu sa samim sobom. U nastavku će se napraviti usporedba s postojećim implementiranim prediktorima iz različitih radova i na različitim podacima.

5.3.1. Podaci za FER

U [14] je implementirana neuronska mreža, višeslojni perceptron, a u [15] je implementirana neuronska mreža s radijalnim funkcijama. Zajednička karakteristika tih prediktora jesu ulazne varijable koje su samo meteorološki i kalendarski podaci, bez varijabli o prijašnjim potrošnjama. Ti prediktori su namijenjeni i učen na podacima izmjerenim na FER-u 2011. godine. Nedostatak tih mjerenja jest da nedostaje nekoliko dana ili je izmjerena potrošnja 0. Takvi podaci su označeni kao nepostojeći te se ne koriste u predviđanju potrošnje. Usporedba s tim algoritmima je zanimljiva jer algoritmi na ulazu primaju sasvim različite ulaze jer GP ne prima podatke o vremenskim prilikama. Jedina zajednička varijabla je kalendarski podatak o mjesecu.

U tim radovima, kao jedan od načina mjerenja kvalitete predviđanja korišten je faktor korelacije koji inače služi za određivanje zavisnosti ili povezanosti između dviju varijabli ili skupa podataka. Najčešće je korišten Pearsonov koeficijent korelacije koji se može izračunati sljedećom formulom:

$$R = \frac{\sum_{i=1}^N (X_i - \bar{X}) (Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^N (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^N (Y_i - \bar{Y})^2}} \quad (14)$$

Gdje je \bar{X} srednja vrijednost skupa X , a \bar{Y} srednja vrijednost skupa Y .

Vrijednost Pearsonovog koeficijenta korelacije kreće se od +1 (savršena pozitivna korelacija) do -1 (savršena negativna korelacija). Tablica 11 uspoređuje rezultate različitih algoritama nad istim skupom podataka. Zbog točnijeg mjerenja pogreške genetskog programiranja nad skupom za evaluaciju, ukupni skup podataka je podijeljen na dva dijela:

- Prvih šest mjeseci, od 1.1.2011 – 30.6.2011,
- Drugih šest mjeseci, od 1.7.2011. – 31.12.2001.

Učenje je provedeno u dvije iteracije, s time da je prediktor uvijek učen na jednom skupu, a evaluiran na drugom. Rezultati su prikazani u ovisnosti o tome na kojem skupu je prediktor evaluiran. Rezultati GP-a su zapravo rezultati ansambla GP-a. Prilikom učenja za svaki sat generirano deset modela te se formulom (9) izlazi iz tih modela spajaju u jedan izlaz.

Tablica 11 Usporedba prediktora na podacima za FER

	Faktor korelacije za radne dane	Faktor korelacije za neradne dane
GP – prvih 6 mj.	0.89797	0.95978
GP – drugih 6 mj.	0.96771	0.94083
[14]	0.94177	0.67159
[15]	0.95198	0.74257

Značajna razlika se može primijetiti kod neradnih dana, gdje je GP pokazao puno bolje rezultate, s time da nisu napravljeni odvojeni modeli za radne i neradne dane kao u [14] i [15]. Ovi rezultati su indikator da GP može biti uspješan za predviđanje potrošnje električne energije.

Tablica 12 prikazuje rezultate GP prediktora po pojedinim mjesecima, kao i ukupan rezultat. Učenje i evaluacija su također podijeljeni u dva skupa pa su i rezultati tako prikazani. Može se uočiti da za pojedine mjesece greška jako raste. To se može objasniti činjenicom da skup za učenje i skup za evaluaciju su bazirani na drugačijim mjesecima pa takvo učenje nije baš najbolje. Ako se prediktori evaluiraju na skupovima na kojima su učeni, greške su 2-3%.

Tablica 12 Rezultati po mjesecima za GP prediktor na podacima za FER

Mjesec	Greška (%)	Mjesec	Greška (%)
Siječanj	14.2180	Srpanj	6.3711
Veljača	3.8362	Kolovoz	9.1656
Ožujak	2.9918	Rujan	4.2927
Travanj	3.2045	Listopad	7.0713
Svibanj	3.1855	Studenj	7.8250
Lipanj	3.8427	Prosinac	12.5022
Ukupna greška	5.5612	Ukupna greška	7.6973

5.3.2. ISO – New England podaci

ISO New England podaci su podaci izmjereni u saveznoj državi *New England* te će se nadalje u radu označavati kao ISO-NE podaci. ISO-NE podaci su stvarni podaci o potrošnji električne energije i cijenama električne energije, u satnoj rezoluciji. Podacima su dodane temperature izmjerene u osam različitih meteoroloških postaja, spojene u jednu vrijednost uz pomoć unaprijed izračunatih težinskih faktora. ISO-NE podaci su mjereni u razdoblju 1.3.2003 – 31.1.2010. godine, ali za potrebe mjerenja će se uzeti dva perioda:

- 1.1.2005. – 31.12.2005. kao skup za učenje
- 1.1.2006. – 31.12.2006. kao skup za evaluaciju.

Tablica 13 prikazuje usporedno rezultate za četiri različita algoritma: umjetnu neuronsku mrežu – ANN, *wavelet* neuronska mreža – SIWNN², regresija potpornim vektorima – SVR i genetsko programiranje – GP. Sva četiri algoritma su učena na sličnim podacima, a

² Engl. *Similar Day-based Wavelet Neural Network*.

evaluira na istim. Rezultati GP-a su zapravo rezultati ansambla GP-a. Prilikom učenja za svaki sat generirano deset modela te se formulom (9) izlazi iz tih modela spajaju u jedan izlaz. U tablici je označeno iz kojih radova su uzeti koji rezultati. U tablici se može primijetiti očita prednost GP-a te da je, kao prediktor, pokazao iznimne rezultate. Pogotovo jer se u istoj tablici nalaze dvije verzije neuronske mreže koja je preporučeni algoritam za predviđanje.

Tablica 13 Usporedba prediktora na ISO-NE 2006 podacima

	ANN [4]	SIWNN [4]	SVR [2]	GP
Siječanj	2.01%	1.6%	1.31%	0.91%
Veljača	1.5%	1.43%	1.12%	1.06%
Ožujak	1.55%	1.47%	0.93%	0.93%
Travanj	1.51%	1.26%	1.34%	0.97%
Svibanj	1.69%	1.61%	1.1%	0.98%
Lipanj	2.3%	1.79%	1.55%	1.49%
Srpanj	3.72%	2.7%	1.86%	1.65%
Kolovoz	3.33%	2.62%	1.5%	1.29%
Rujan	1.6%	1.48%	1.15%	0.97%
Listopad	1.52%	1.38%	1.2%	1.16%
Studen	1.73%	1.39%	1.09%	1.08%
Prosinac	1.91%	1.75%	1.61%	1.09%
Ukupno	2.03%	1.71%	1.31%	1.13%

Zaključak

U ovom radu je opisano kratkoročno predviđanje električne energije i predstavljeni su česti problemi koji prate to područje. Kratkoročno predviđanje potrošnje električne energije je korisno i unosno područje koje može poslužiti za izradu modela proizvodnje električne energije. Takav model bi se oslanjao na predviđanje potrošnje za sljedeći dan i na temelju tog predviđanja određivao cijenu električne energije. Mogu se i generirati rasporedi održavanja i/ili popravljivanja uređaja, a da se ne utječe na količinu proizvedene energije.

U ovom radu je predstavljen algoritam genetskog programiranja kao prediktor potrošnje električne energije. To je bio veliki izazov jer ne postoje javno-dostupni radovi koji opisuju takav način predviđanja. Zbog toga, a i zbog nekonzistentnih prijedloga za parametre genetskog programiranja, ispitivani su i mjereni svi parametri 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 određivanja parametara odgovaraju nekim preporukama kao što su velika populacija ili manja veličina turnira u selekciji.

Radi dodatnog poboljšanja prediktora, izmjereni su utjecaji neki ulaznih varijabli. Tu se pokazalo bitno da prediktor dobije informacije o potrošnji iz prethodnih 24 sata, ali da meteorološke prilike nisu toliko presudne.

Zanimljivo svojstvo genetskog programiranja jest da ima ugrađen odabir ulaznih varijabli. Ako se GP-u na ulaz da jako puno (sličnih) varijabli, zbog utjecaja evolucije će biti izbačene varijable koje utječu slabo ili nikako. Možda se to svojstvo može iskoristiti kao neka vrsta algoritma odabira značajki. Tada se genetskom programiranju može na ulaz dati puno varijabli, a on sam odlučuje koje od njih će iskoristiti za predviđanje.

Literatura

- [1] DE FELICE, MATTEO, AND XIN YAO. "Short-term load forecasting with neural network ensembles: a comparative study [application notes]." *Computational Intelligence Magazine, IEEE* 6.3 (2011): 47-56.
- [2] ČEPERIĆ, ERVIN, VLADIMIR ČEPERIĆ, AND ADRIJAN BARIC. "A Strategy for Short-Term Load Forecasting by Support Vector Regression Machines." (2013): 1-9.
- [3] HERNANDEZ, LUIS, ET AL. "Short-term load forecasting for microgrids based on artificial neural networks." *Energies* 6.3 (2013): 1385-1408.
- [4] CHEN, YING, ET AL. "Short-term load forecasting: similar day-based wavelet neural networks." *Power Systems, IEEE Transactions on* 25.1 (2010): 322-330.
- [5] ALBADI, M. H., AND E. F. EL-SAADANY. "Demand response in electricity markets: An overview." *IEEE Power Engineering Society General Meeting*. Vol. 2007. 2007.
- [6] LUKE, SEAN, AND LEE SPECTOR. "A revised comparison of crossover and mutation in genetic programming." *Genetic Programming* 98 (1998): 208-213.
- [7] POLI, RICCARDO, ET AL. "A field guide to genetic programming". *Lulu.com*, 2008.
- [8] DRACOPOULOS, DIMITRIS C., AND SIMON KENT. "Genetic programming for prediction and control." *Neural Computing & Applications* 6.4 (1997): 214-228.
- [9] HUI, ANTHONY. "Using genetic programming to perform time-series forecasting of stock prices." *Genetic Algorithms and Genetic Programming at Stanford* (2003): 83-90.
- [10] BREZOCNIK, MIRAN, MIHA KOVACIC, AND MIRKO FICKO. "Prediction of surface roughness with genetic programming." *Journal of materials processing technology* 157 (2004): 28-36.
- [11] LEE, DONG GYU, BYONG WHI LEE, AND SOON HEUNG CHANG. "Genetic programming model for long-term forecasting of electric power demand." *Electric power systems research* 40.1 (1997): 17-22.
- [12] KABOUDAN, MARK A. "Genetic programming prediction of stock prices." *Computational Economics* 16.3 (2000): 207-236.
- [13] GOLUB, M. "Skripta 2. dio: Paralelni genetski algoritmi", Zagreb (2004): 55-61, http://www.zemris.fer.hr/~golub/ga/ga_skripta2.pdf
- [14] NOVAK, H. "Predviđanje potrošnje električne energije na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva višeslojnom perceptronskom neuronskom mrežom", *Završni Rad, Fakultet elektrotehnike i računarstva* (2012)
- [15] LUBIČIĆ, J. "Predviđanje potrošnje električne energije na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva neuronskom mrežom s funkcijama s kružnom osnovicom", *Završni Rad, Fakultet elektrotehnike i računarstva* (2012)

Sažetak

Kratkoročna prognoza potrošnje električne energije

Danas je kratkoročno predviđanje potrošnje električne energije istraženo i razvijeno područje, ali i dalje se u njega ulaže. Glavni razlog tome je to što od dobrog predviđanja svi sudionici u elektroenergetskim sustavima profitiraju. U ovom radu je istraženo područje predviđanja potrošnje električne energije i opisani glavni problemi, kao i osnovne preporuke prilikom izrade prediktora. Opisane su moguće ulazne i izlazne varijable. Predstavljani su neki od značajnijih dosadašnjih implementacija predviđanja potrošnje električne energije. Također je predstavljeno genetsko programiranje kao novi način predviđanja te su napravljena mjerenja i testiranja kako bi se utvrdila kvaliteta implementacije.

Ključne riječi: Kratkoročno predviđanje potrošnje električne energije, genetsko programiranje.

Abstract

Short term electric grid load forecasting

Nowadays, short-term load forecasting, STLF, is well explored and developed area, but is still area of interest. The main reason for that is benefit for all participants in the power system. This paper explores STLF and describes the main problems, as well as basic recommendations when making predictors. Possible input and output variables are described. Some of the highlights of the existing implementation for STLF are presented. Genetic programming is presented as a new way to predict load and measurements and tests are made to determine the quality of implementation.

Keywords: Short-term load forecasting, genetic programming.