SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3614

UČINKOVITOST OPTIMIZACIJSKIH ALGORITAMA U OVISNOSTI O ZNAČAJKAMA PROBLEMA

Vlaho Poluta

Zagreb, lipanj, 2014

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

Zagreb, 12. ožujka 2014.

ZAVRŠNI ZADATAK br. 3614

Pristupnik: **Vlaho Poluta**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Učinkovitost optimizacijskih algoritama u ovisnosti o značajkama problema**

Opis zadatka:

Opisati svojstva optimizacijskih problema s obzirom na značajke krajolika dobrote. Ispitati uspješnost više algoritama optimizacije na skupu kontinuiranih optimizacijskih problema. Odrediti ovisnost uspješnosti proizvoljnog algoritma stohastičke optimizacije o značajkama optimizacijskog problema. Razviti programski sustav za određivanje prikladnog algoritma optimizacije za zadani problem temeljen na strojom učenju. Ispitati učinkovitost ostvarenog sustava u ovisnosti o metodi strojnog učenja i odabranim značajkama optimizacijskih problema. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:

Izv.prof.dr.sc. Domagoj Jakobović

Djelovođa:

Doc.dr.sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof.dr.sc. Siniša Srbljić

ZAHVALA

Ovim putem zahvalio bih mentoru Doc.dr.sc. Domagoju Jakoboviću na vodstvu, entuzijazmu i pomoci koju mi je pružio u postupku nastajanja ovog rada.

Također želio bih se zahvaliti svima koji su mi pomogli u lektoriranju ovog rada.

Sadržaj

1. Uvod

Razvojem računarstva i računalne znanosti počeli su se javljati problemi velike složenosti, uglavnom eksponencijalne. Uzmimo za primjer SAT problem [20] koji je na malim primjerima dosta jednostavan i ne zahtijeva puno računalne moći, ali s porastom varijabli potrebno je eksponencijalno više procesorskog vremena tako da je već za 30 varijabli potrebno izračunati 1.073.741.824 kombinacija. Možda se razvoj računalne opreme čini brz, no nažalost nije ni blizu eksponencijalnom rastu složenosti ovakvog problema.

Tome u pomoć dolaze optimizacijski algoritmi koji pokušavaju u prihvatljivom vremenu riješiti takve složene probleme. Nažalost imaju svoju cijenu u tome da ne moraju pronaći optimalno rješenje tog problema. No ipak daju poprilično dobra rješenja, ako ne i optimalna. Kako je rasla popularnost optimizacijskih algoritama, tako je rasla i njihova raznolikost. Ta velika raznolikost je posljedica toga što nema jednog algoritma optimizacije koji bi bio najbolji, nego je svaki algoritam bolji odnosno lošiji nasuprot nekog drugog optimizacijskog algoritma za bilo koja 2 različita problema.

Sljedeći korak je odrediti kako opisati zadane probleme, grupirati te zadane probleme i onda pronaći optimizacijski algoritam koji bi bio najbolji ili barem vrlo dobar za optimizaciju te skupine problema. U ovom radu se za opis problema koristi krajolik dobrote tog problema, to jest funkcije koja opisuje taj problem. U nastavku je dan detaljniji opis krajolika dobrote, čije se značajke krajolika dobrote mogu opisati kao vektor koji opisuje karakteristike neke funkcije.

Cilj ovog rada je predviđanje kako bi se postojeći algoritam optimizacije ponašao za neku novu optimizacijsku funkciju. Za postupak predviđanja u ovom radu koristila se baza od 52 optimizacijske funkcije [18,19] i 10 različitih optimizacijskih algoritama. Isto tako korištene su 4 metode učenja: simbolička regresija genetskim programiranjem, generiranje stabla odluke genetskim programiranjem, učenje neuronske mreže algoritmom selekcije klona, i učenje neuronske mreže diferencijskom evolucijom.

2. Krajolik dobrote

Glavna stvar kod domene optimizacije nije u oblikovanju najboljeg algoritma za sve optimizacijske probleme, nego u tome da se nađe algoritam koji je najprilagođeniji danoj skupini problema. Pitanje je li dani optimizacijski algoritam A apsolutno bolji od optimizacijskog algoritma B je besmisleno. Dapače, „*No free lunch*“ teorem to dokazuje. Pod određenim pretpostavkama nijedan nije superiorniji od nijednog drugog na svim mogućim optimizacijskim problemima; to znači ako je algoritam A bolji od algoritma B na određenom problemu, uvijek postoji drugi problem na kojem je B bolji od A [2]. Nijedna metaheuristika ne može biti u potpunosti bolja od neke druge metaheuristike. U tom slučaju pojam superiornosti može imati smisla samo u rješavanju određene skupine problema. [1]

Odrediti kako razlikovati određene skupine problema nije jednostavan zadatak. Tu se kao moguće rješenje javlja krajolik dobrote problema odnosno krajolik dobrote funkcije cilja koja opisuje zadani problem. Analiza krajolika dobrote se provodi u nadi predviđanja ponašanja određenih komponenata pretrage optimizacijskim algoritmima.

2.1 Definicija

Prostor pretraživanja (*engl. search space*) je određen usmjerenim grafom G = (S, E), gdje S označava skup rješenja problema koja su definirana reprezentacijom rješenja problema, a skup bridova E odgovara operatorima kretanja koji se koriste kako bi se generirala nova rješenja. [1]

Krajolik dobrote (*engl. Fitness landscape*) L možemo definirati parom (G, f), gdje G predstavlja prostor pretraživanja, a f predstavlja funkciju cilja koja vodi tu pretragu (funkcija dobrote). [1]

2.2 Karakteristike krajolika dobrote

Postoji više karakteristika koje opisuju krajolik dobrote. Neki od njih mogu biti broj lokalnih optimuma, distribucija lokalnih optimuma, ispresjecanost, itd. Samo iz njihovog naziva može se zaključiti da se većina temelji na statistici. Uglavnom se mogu razlikovati 2 skupine. Prva skupina *globalni indikatori*, indikatori su koji opisuju strukturu cijelog krajolika. Pri tome koriste cijeli prostor pretraživanja ili njegovu aproksimaciju. To možda zvuči obećavajuće, ali globalni indikatori nisu dobri za kategorizaciju metaheuristika po krajoliku dobrote jer je u optimizaciji cilj pronaći dobra rješenja, a ta dobra rješenja predstavljaju samo mali dio prostora pretraživanja, dok globalni indikatori gledaju na njih kao na obične točke. Indikatori koji se koriste u određivanju krajolika dobrote za klasificiranje metaheuristika zovu se *lokalni indikatori*. Lokalni pristup sastoji se od toga da se gleda kako metaheuristika istražuje određeni dio prostora pretraživanja.

Može se naslutiti da su se za ovaj rad koristili lokalni indikatori. Opis lokalnih indikatora koji su korišteni u ovom radu mogu se naći u [17]. Program za određivanje lokalnih indikatora koji je korišten u sklopu ovog rada je preuzet iz [17]. Određivane su značajke krajolika dobrote za ispitne (*engl. benchmark*) funkcije [18, 19].

U nastavku slijedi opis nekoliko značajki krajolika dobrote koje su korištene u sklopu ovog rada.

2.3 Neke značajke krajolika dobrote

Promjer populacije P od S definira se kao maksimalna udaljenost između elemenata populacije:

$$diam(P) = \max_{s,t \in P} dist(s,t) \tag{2.1}$$

Za populaciju P *prosječna udaljenost dmm(P)* i *normalizirana prosječna udaljenost* Dmm(P) definirane su kao:

$$dmm = \frac{\sum_{s \in P} \sum_{t \in P, t \neq s} dist(s,t)}{|P| \cdot (|P|-1)} \quad Dmm(P) = \frac{dmm(P)}{diam(S)} \tag{2.1}$$

Normalizirana prosječna udaljenost karakterizira koncentriranost populacije P u prostoru pretraživanja. Znatno manja udaljenost ukazuje da su rješenja koja pripadaju populaciji P skupljena u manji prostor pretraživanja.

Kako se može koristiti više indikatora za analizu distribucije rješenja u prostoru rješenja, odabrana je *amplituda* Amp(P) proizvoljne populacije P, kao relativna razlika najbolje i najgore jedinke.

$$Amp(P) = \frac{|P| \cdot (\max_{s \in P} f(s) - \min_{s \in P} f(s))}{\sum_{s \in P} f(s)} \tag{2.3}$$

Duljina hoda (*engl. Length of the walk*), to jest prosječna duljina silaznih koraka Lmm(P) u populaciji P definirana kao:

$$Lmm(P) = \frac{\sum_{p \in P} l(p)}{|P|}, \tag{2.4}$$

gdje je l(p) dužina hoda koja počinje od rješenja p∈ P. Na intuitivnom nivou duljina hoda daje informacije o ispresjecanosti krajolika. U ispresjecanom krajoliku hodovi su kraći, dok su u neispresjecanom krajoliku duži.

2.4 Klasifikacija krajolikom dobrote

U ovom radu značajke krajolika dobrote su korištene za klasifikaciju optimizacijskih algoritama na 2 klase. Te klase predstavljaju odgovor na pitanje je li algoritam dobar ili loš za optimizaciju danog problema. Kao klasifikatori korišteni su simbolička regresija genetskim programiranjem, stablo odluke izgrađeno genetskim programiranjem, neuronska mreža učena diferencijskom evolucijom i neuronska mreža učena algoritmom selekcije klona.

Ispitivanje u radu je podijeljeno u dvije faze. U prvoj fazi ispitivanja određuju se najbolji parametri za sva 4 modela učenja prema grešci na skupu za učenje dok se u drugoj fazi testiranja određuju kriteriji zaustavljanja i radi se usporedba modela po pogrešci nad skupom za unakrsnu provjeru.

Optimizacijski algoritmi koji su korišteni u radu u postupku ispitivanja su:

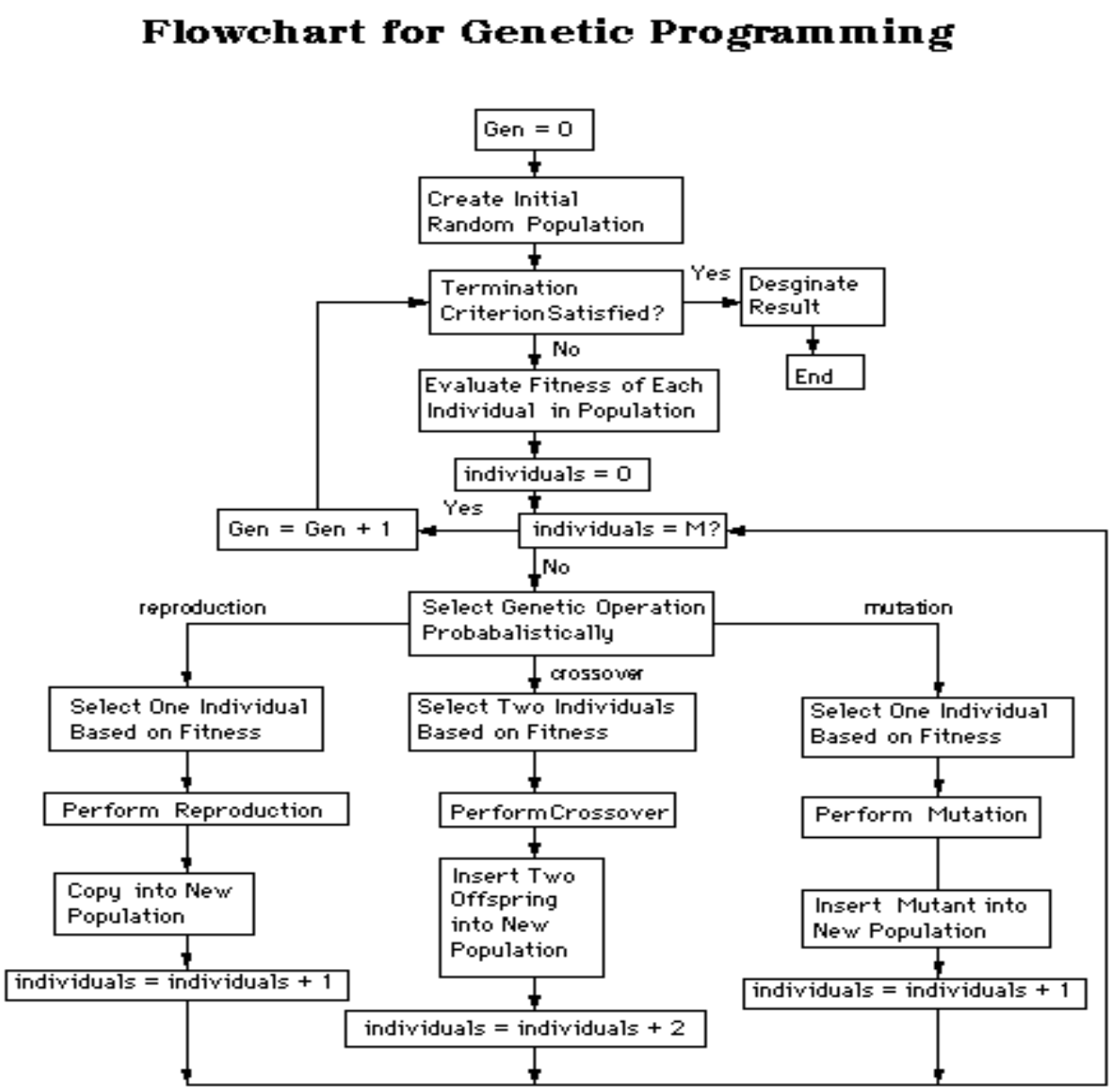
- Umjetna kolonija pčela (*engl. Artificial bee colony, ABC*)
- Diferencijalna evolucija (*engl. Differencial evolution, DE*)
- Algoritam genetskog kaljenja (*engl. Genetic Annealing, Gan*)
- SST genetski algoritam s aritm. križanjem (GA_SST_arit)
- RW genetski algoritam s aritm. križanjem (GA_RW_arit)
- Optimizacija rojem čestica (*engl. Partical swarm optimization, PSO*)
- Algoritam klonske selekcije (*engl. Clonal selection algorithm, CLONALG*)
- RW genetski algoritam (*engl. Genethic algorithm RW, GA_RW*)
- SST genetski algoritam (*engl. Genethic algorithm* SST*, GA*_SST)
- Algoritam hibrida GA i lokalne pretrage (GHJ)

3. Genetsko programiranje

U umjetnoj inteligenciji genetsko programiranje (GP) je evolucijska metaheuristika inspirirana biološkom evolucijom. Služi pronalasku računalnih programa koji izvode određene poslove. Esencijalno GP je skup instrukcija i funkcija dobrote kako bi se izmjerilo koliko je dobro računalo odradilo određeni zadatak. To je vrsta genetskog algoritma gdje svaka jedinka ima genotip strukture stabla, te tehnika strojnog učenja koja služi kako bi se optimirala populacija stabala po funkciji dobrote koja je određena sposobnošću izvođenja zadanog zadatka. [10]

Genetsko programiranje se smatra podvrstom genetskog algoritma, tako da može biti i eliminacijski i generacijski. U praksi se pokazalo da je eliminacijska izvedba uglavnom superiornija (ne mora uvijek biti, kao posljedica teorema „*No free lunch*“), tako da je u ovom radu korištena eliminacijska izvedba. Izgradnja inicijalne populacije radi se *ramped-half-and-half* metodom (o *ramped-half-and-half* metodi više u [11]). Prilikom križanja koristi se mala kazna plagiranja roditelja (2%), a za neuspjeh križanja (zbog najveće dozvoljene dubine i najvećeg dozvoljenog broja čvorova) koristi se reprodukcija prvog roditelja natrag u populaciju (uz kaznu plagiranja). Slika 3.1 iz [12] je primjer prikaza rada genetskog programa, a pseudokod algoritma korištenog u radu je:

```
Population = random initial population
Evaluate(Population)
Best = getBest(Population)
While not (stopping criterion)
        Parent1 = tournament(Population)
        Parent2 = tournament(Population)
        Child = crossover(Parent1, Parent2)
        Child = mutation(Child)
        Evaluate(Child)
        Replaced = mockTournament(Population)
        RemoveFromPopulation(Replaced)
        InsertInPopulation(Child)
        If(isBetter(Child,Best)) Best = Child
return Best
```



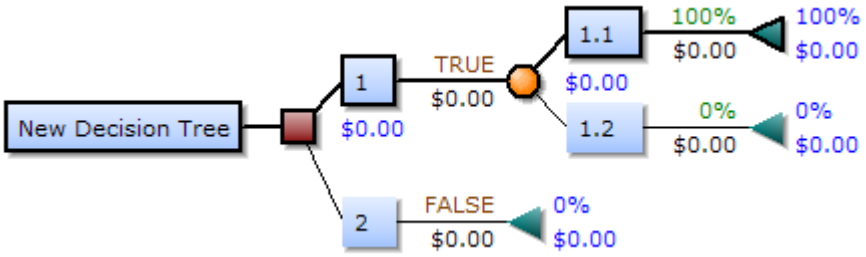
Slika 3-1 Rad genetskog programa [12]

U nastavku su prikazane dvije varijante stabla GP-a korištene u ovom radu za određivanje učinkovitosti optimizacijskih algoritmom.

3.1 Stablo odluke

Stablo odluke je alat potpore odlučivanju koje koristi graf u obliku stabla ili model odluka i njihovih mogućih posljedica, uključujući i nedeterminističke posljedice (čvorovi koji generiraju slučajne izlaze). Može se smatrati vrstom prikaza algoritma. Obično se koriste u istraživanju operacija, posebno u analizi odluka, da se pomogne

identificirati strategija koja će najvjerojatnije stići do cilja. Još jedna upotreba stabla odluke je kao opisno sredstvo računanja uvjetne vjerojatnosti (Slika 3.2). [14]



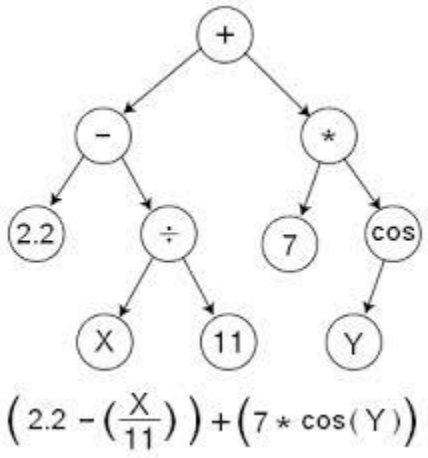
Slika 3-2 Stablo odluke [14]

U sklopu ovog rada korišteno je stablo odluke gdje listovi stabla predstavljaju pripadnost određenoj klasi, dok su ostali čvorovi deterministčke IF funkcije koje vraćaju binarnu odluku. Svaki optimizacijski algoritam ima svoje stablo odluke. Stablo se gradi tako da su funkcijski čvorovi sagrađeni od više ili manje upita. Neka za primjer upita kakav se nalazi u funkcijskim čvorovima uzmemo: $IF(x[n] > CONST)$. X predstavlja zadani vektor krajolika dobrote, a x[n] je slučajno izabrani element tog vektora. Ovdje treba napomenuti da su svi elementi zadanog vektora skalirani na domenu [-1,1]. CONST je slučajno izabrana konstanta koja je element domene [-1,1]. Vrsta upita (veće ili manje) je isto tako slučajno generirana prilikom stvaranja čvora. U ovom radu listovi predstavljaju klase koje određuju koliko dobro algoritam optimizacije optimizira zadanu funkciju čiji krajolik dobrote je zadan. U postupku ispitivanja korištene su klasifikacije na 2 ili 3 klase.

3.2 Simbolička regresija

Simbolička regresija (engl *symbolic regression*, SR) predstavlja proces tijekom kojega se izmjereni podaci pokušavaju predočiti odgovarajućom matematičkom formulom poput „x^{2+C}“, „$\sin(x) + 1/e^x$“, itd.

Taj proces je među matematičarima poznat i korišten kad dobiju podatke nepoznatog izvora. Mnogo vremena SR su mogli raditi samo ljudi, ali zadnjih nekoliko desetljeća mogu i računala. Glavni atributi simboličke regresije na računalu je taj da je izvodi evolucijski algoritam i cijeli evolucijski proces radi sa skupom funkcija i skupom terminala. Funkcijski skup je skup svih funkcija zadanih od strane korisnika, dok je terminalni skup, skup od svih konstanti i varijabli. [15]



Slika 3-3 Simbolička regresija

U ovom radu funkcijski skup se sastoji od operatora $+, -, *, /, \sin(x), \exp(x), \ln(x), \log_{10}(x)$. Odmah treba napomenuti da u ovom

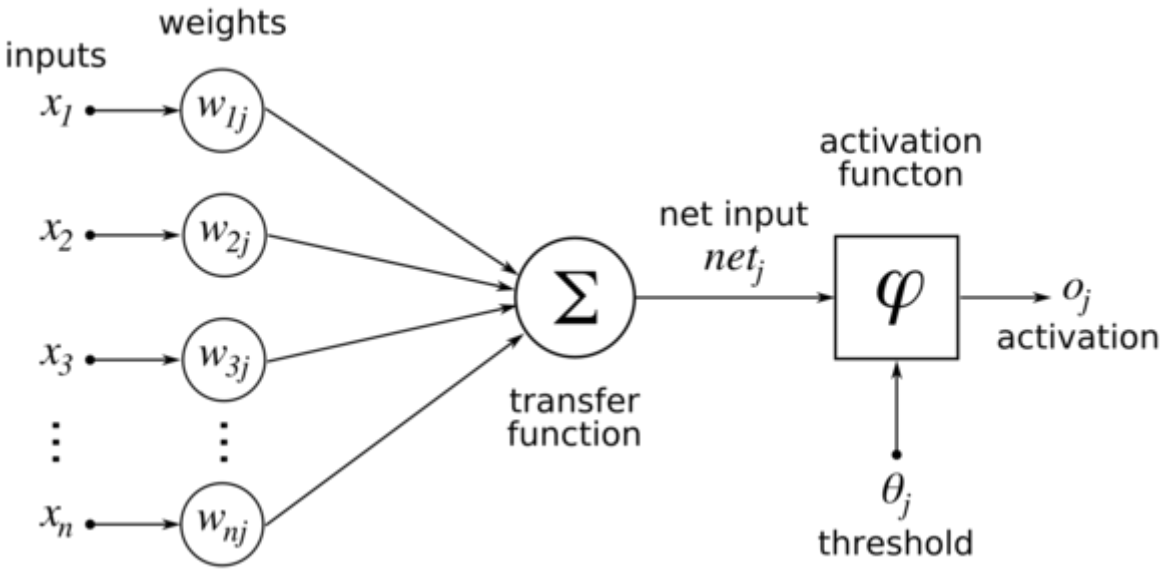
radu, kao i kod stabla odluke, jedno stablo, to jest jedna generirana formula pripada jednom algoritmu optimizacije i označava koliko dobro taj algoritam opisuje tu funkciju. Terminalni skup sastoji se od slučajno izabranih konstanti i značajki krajolika dobrote koji opisuje zadanu funkciju. Kako se kao izlaz takve formule očekuje realan broj, taj broj se skalira na domenu [0,1]. Taj novi broj predstavlja mjeru koliko dobro optimizacijski algoritam kojem pripada formula dobivena simboličkom regresijom optimizira funkciju čiji je krajolik dobrote zadan. Što je taj broj bliže 1 to predstavlja da će optimizacijski algoritam dobro optimizirati zadanu funkciju, a što je taj broj bliže 0 predstavlja da će taj optimizacijski algoritam loše optimizirati zadanu funkciju.

4. Neuronske mreže

U računalnoj znanosti, umjetne neuronske mreže (*engl artificial neural networks, ANN*) su računalni modeli inspirirani centralnim živčanim sustavom mozga. Ovi modeli se koriste u strojnom učenju (engl. machine learning) i prepoznavanju uzoraka (engl. pattern recognition). [4]

Umjetne neuronske mreže predstavnik su konektivističkog pristupa unutar područja umjetne inteligencije koji se temelji na upogonjavanju niza vrlo jednostavnih procesnih elemenata koji sami po sebi ne ispoljavaju nikakva inteligentna obilježja, ali kada ih se međusobno skupi velik broj, dobiva se sustav koji iskazuje vrlo interesantna svojstva.[5]

U ovom radu je korišten osnovni oblik neuronske mreže (slika 4.1) sa sigmidalnom funkcijom [6] kao aktivacijskom funkcijom. O načinu rada neuronske mreže može se pročitati više u [4, 5].



Slika 4-1 Neuronska mreža [22]

Neuronska mreža je korištena kako bi se strojno naučila klasifikacija optimizacijskih algoritama nad krajolikom dobrote. U postupku ispitivanja na ulaze se dovodi vektor značajki krajolika dobrote neke funkcije (ta funkcija za optimizacijske algoritme predstavlja funkciju dobrote). Broj skrivenih čvorova se mijenjao u postupku ispitivanja (više o tome u nastavku), a izlazni čvorovi neuronske mreže određuju kvalitetu klasifikacije za jedan određeni optimizacijski algoritam nad funkcijom čiji je krajolik dobrote dan na ulaz te neuronske mreže.

Za neuronsku mrežu klasifikacija na 2 klase znači da ima dva izlazna čvora. Jedan od njih predstavlja klasu u kojoj je funkcija dobra, dok drugi predstavlja klasu u kojoj

je funkcija loša. Kako izlazni čvorovi primaju vrijednosti iz domene [0,1], čim čvor teži 0 to znači da ne pripada toj klasi, čim teži 1 to znači da pripada toj klasi. Tako da ako je prvi čvor onaj koji predstavlja klasu u kojoj je optimizacijski algoritam dobar za tu funkciju, možemo imati izlaze: 10 - Zadani optimizacijski algoritam je dobar za zadanu funkciju. 01 – Zadani optimizacijski algoritam je loš za zadanu funkciju. Za preostale izlaze 11, 00, optimizacijski algoritam je i dobar i loš, odnosno optimizacijski algoritam nije ni dobar ni loš ne može se ništa konkretno zaključiti, tj. takvi izlazi se mogu ili odbaciti ili smatrati trećom klasom. U postupku učenja neuronske mreže takvi izlazi su smatrani pogreškom klasifikacije.

U nastavku su opisana dva algoritma koja su korištena u ovom radu za učenje neuronske mreže.

4.1 Algoritam selekcije klona

U umjetnim imunološkim sustavima, algoritam selekcije klona (*engl. Clonal selection algorithm, Cloang*) je metaheuristika inspirirana teorijom selekcije klona stečenog imuniteta koji objašnjava kako B i T limfociti razvijaju tijekom vremena odgovor na antigene. Ovi algoritmi se fokusiraju na Darwinove atribute teorije gdje je selekcija inspirirana afinitetom interakcija antigena i antitijela, reprodukcija je inspirirana diobom stanica, a mutacija je inspirirana somatskom hipermutacijom (pojačana mutacija stanica). [8]

Opis rada Clonalg algoritma, koji je korišten u ovom radu može se naći u [5,16]. Više o Clonalgu može se naći u [9], odakle je preuzet pseudokod (Slika 4.2).

```
1  begin
2      affinities ⟵ calcAffinities(antibodies);
3      while not stopCondition() do
4          selectedAntibodies ⟵ select(antibodies, affinities, nSelection);
5          clones ⟵ clone(selectedAntibodies, affinities, β);
6          clones ⟵ hypermutate(clones, affinities, ρ);
7          cloneAffinities ⟵ calcAffinities(clones);
8          update(antibodies, affinities, clones, cloneAffinities);
9      bestAffinity ⟵ getBestAffinity(antibodies);
10 end
```

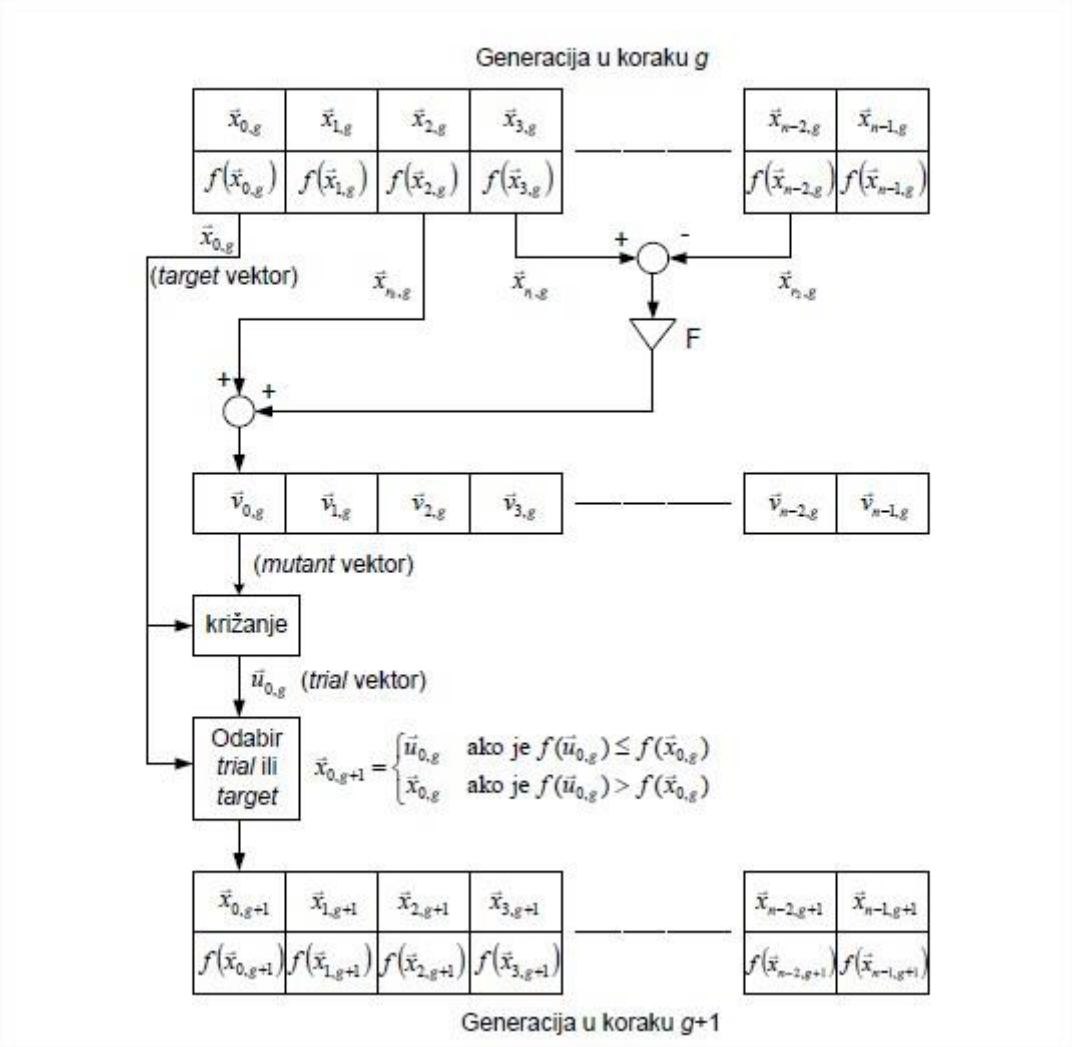
Slika 4-2 Algoritam selekcije klona [9]

4.2 Algoritam diferencijske evolucije

Algoritam diferencijske evolucije, iako izravno ne pripada među biološki inspirirane algoritme, ima dosta sličnosti s genetskim algoritmom. To je također eliminacijski algoritam koji nove potomke stvara uporabom operatora diferencijacije.[5]

Osnovna varijanta algoritma diferencijske evolucije radi tako da ima jednu populaciju jedinki. Kandidati se pomiču po prostoru pretraživanja koristeći jednostavne matematičke formule koje kombiniraju pozicije ostalih jedinki. Ako je nova pozicija bolja ili jednako dobra kao trenutna, trenutna jedinka je zamijenjena novom. Taj postupak se ponavlja dok uvjet zaustavljanja nije zadovoljen.[7]

U radu je korišten algoritam opisan u [5]. Kako je već u inicijalnim fazama ispitivanja strategija DE/best/1/ [5] bila daleko bolja od ostalih implementiranih, iskorištena je za sva daljnja ispitivanja.



Slika 4-3 Algoritam diferencijske evolucije [5]

5. Određivanje parametara modela učenja

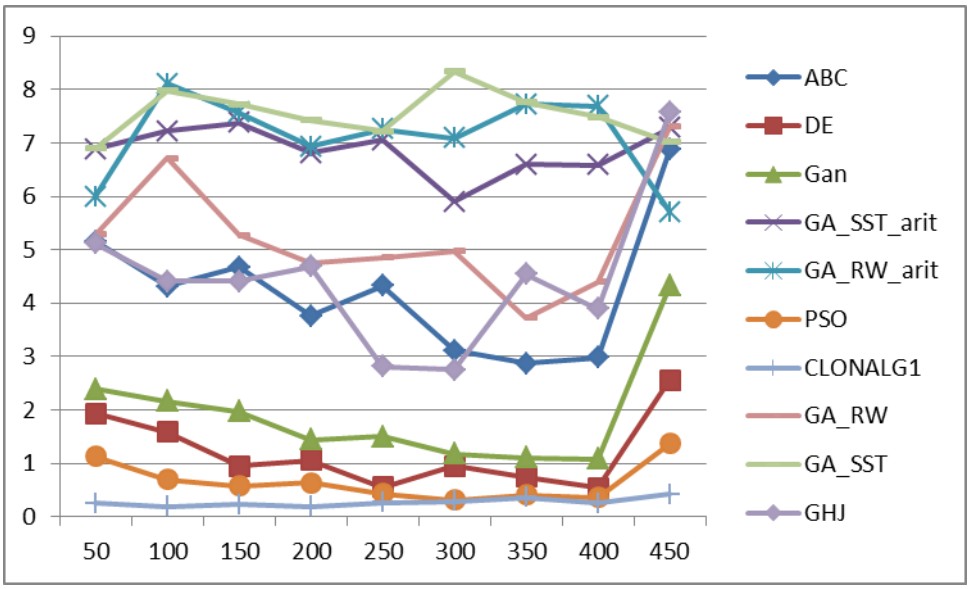
U prvoj fazi ispitivanja određivali su se najbolji parametri za svaku od 4 metode učenja. Za ovo ispitivanje sve funkcije [18,19] bile su korištene kao skup za učenje. Kako su sve metode učenja zapravo stohastički, algoritmi ispitivanje određene vrijednosti za svaki parametar je bilo ponovljeno 30 puta i za dobrotu te vrijednosti za dani parametar je bio uzet medijan tih testova. Svaki test je trajao 100000 evaluacija. Svako ispitivanje se provodilo za određen broj evaluacija. Pri završetku ispitivanja jednog parametra dobivene su prosječne greške za različite vrijednosti tog parametra. Kao najbolja vrijednost za zadani parametar bila je uzeta ona vrijednost pri kojoj je greška bila minimalna.

Inicijalni skup vrijednosti parametara bio je ručno zadan, i kad bi se pronašla nova najbolja vrijednost određenog parametra, ta nova vrijednost je bila korištena kao vrijednost za taj parametar u postupku određivanja najboljih vrijednosti ostalih parametara.

5.1 Simbolička regresija genetskim programiranjem

Za simboličku regresiju ispitano je sveukupno 8 parametara. Nakon ispitivanja tih parametara bile su zapisane i dalje korištene najbolje vrijednosti tih parametara.

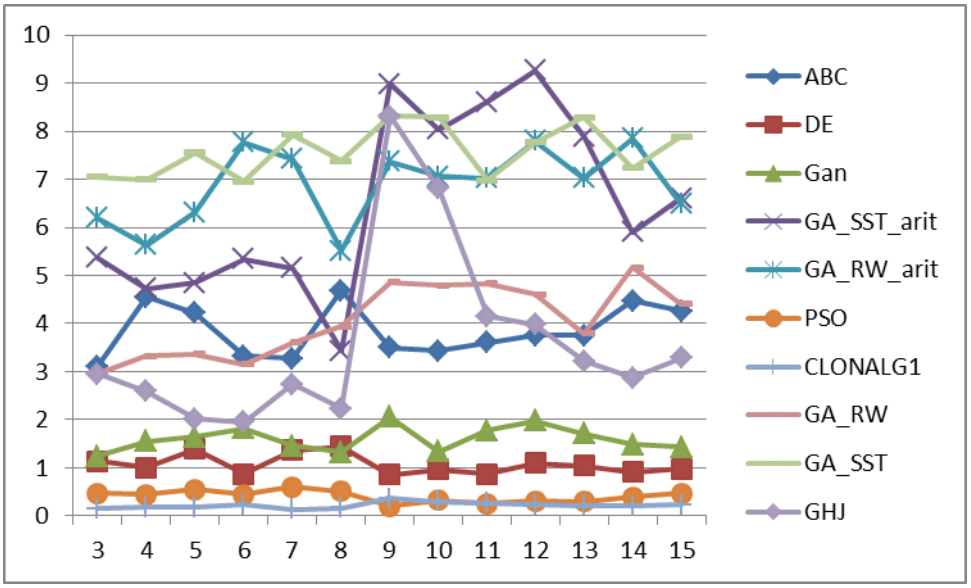
Prvi ispitani parametar je broj jedinki u populaciji. Rezultati ispitivanja se mogu vidjeti na slici 5.1.



Slika 5-1 Graf ovisnosti pogreške o veličini populacije

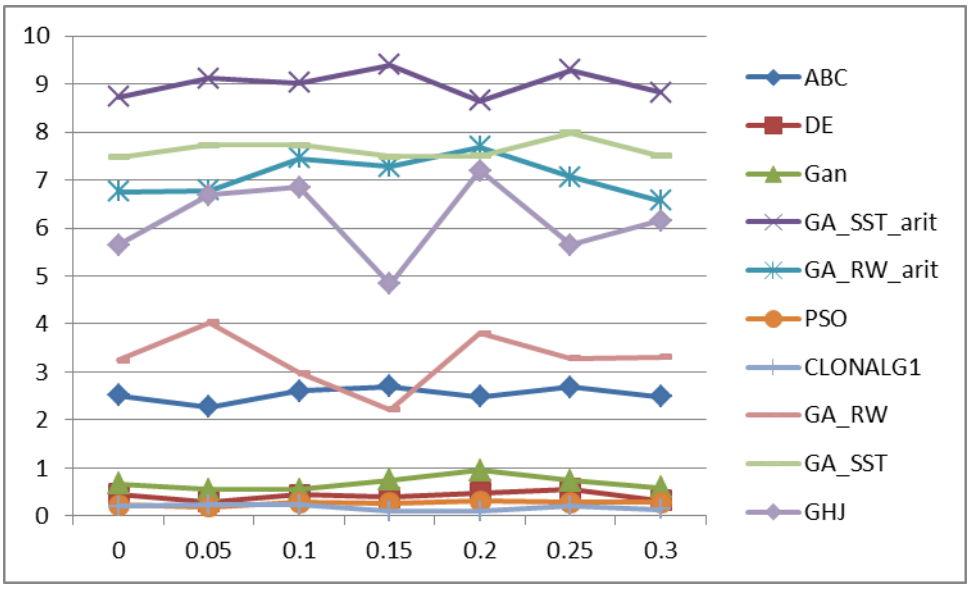
Na uspravnoj osi se nalazi vrijednost pogreške klasifikacije, koja se za simboliču regresiju računa kao apsolutna vrijednost ispravne klase (koja je predstavljena kao 0 ili 1) oduzete od vrijednosti sigmoidalne funkcije simboličke regresije. Na vodoravnoj osi se nalazi količina jedinki u populaciji za dano ispitivanje. Iz slike 5.1, a i iz sumiranja tih grešaka, može se primijetiti da 300 jedinki daje najmanju pogrešku klasifikacije.

Drugi parametar je količina slučajno odabranih jedinki za turnirski odabir svakoga roditelja. Na slici 5.2 može se vidjeti graf koji to opisuje. Na vodoravnoj osi ovog puta se mogu vidjeti različite vrijednosti količine slučajno odabranih jedinki za turnirsku selekciju. Za najbolju količinu je uzeto 7 slučajno odabranih jedinki.



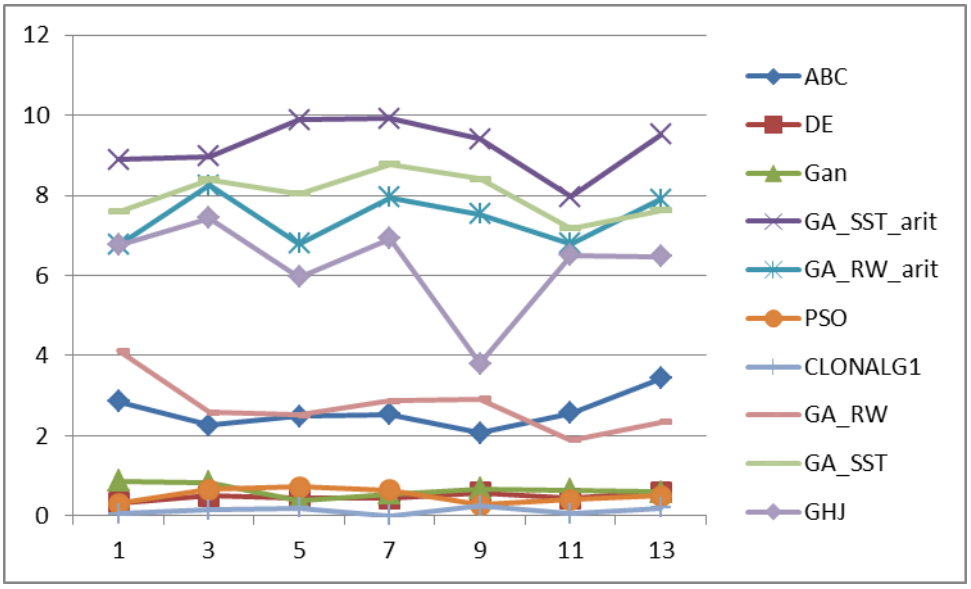
Slika 5-2 Graf ovisnosti pogreške o turnirskoj selekciji

Treći parametar je postotak koji se koristio pri generaciji listova stabla. Taj postotak predstavlja kolika je šansa da taj list stabla bude slučajno odabrana konstanta, a ne jedan od elemenata vektora krajolika dobrote. Graf se može vidjeti na slici 5.3, a za najbolju vrijednost je uzeto 0.15.



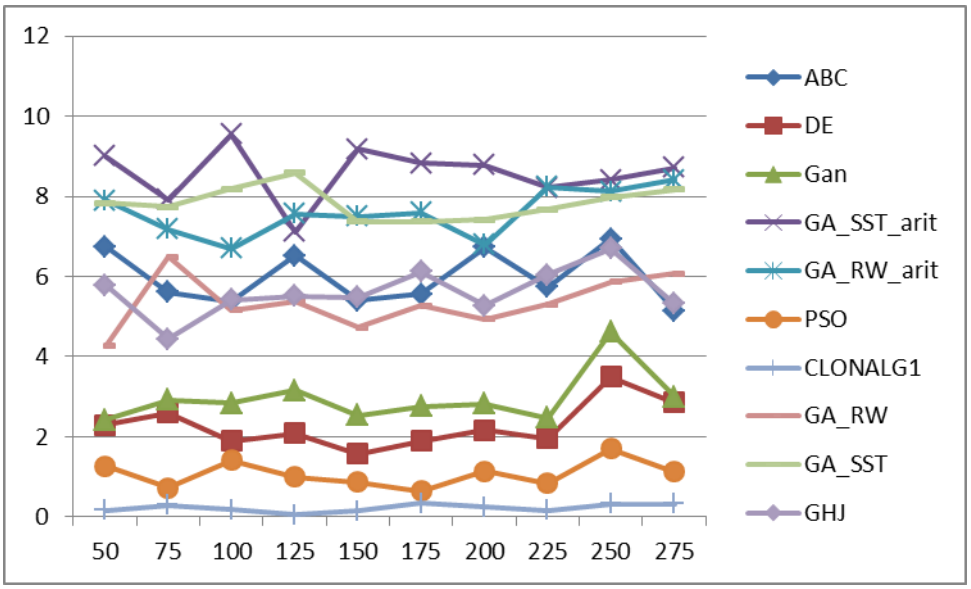
Slika 5-3 Graf ovisnosti pogreške o postotku konstanta

Četvrti parametar je radijus iz kojeg se slučajnim odabirom biraju konstante za listove. Naravno da se ispostavilo da je najbolji postotak tih konstanti 0, ovaj korak ispitivanja bi bio preskočen. Iz grafa 5.4 dobiva se da je najbolji početni radijus ±11.



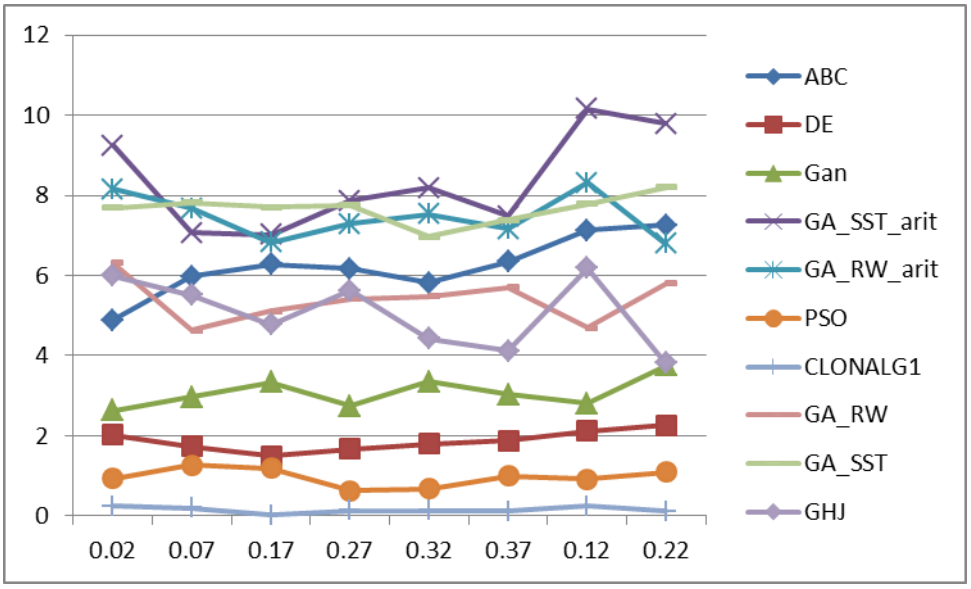
Slika 5-4 Graf ovisnosti pogreške o radijusu slučajno odabranih konstanti

Peti parametar je maksimalni broj čvorova u stablu. Ovdje naravno nije bio cilj naći što bolji broj čvorova, jer da nema ovog ograničenja, razvijala bi se stabla koja bi bila pogodna za rješavanje samo skupa za učenje. Graf dobivenih vrijednosti može se vidjeti na slici 5.5, a kao najbolje rješenje se pokazalo 150 čvorova.



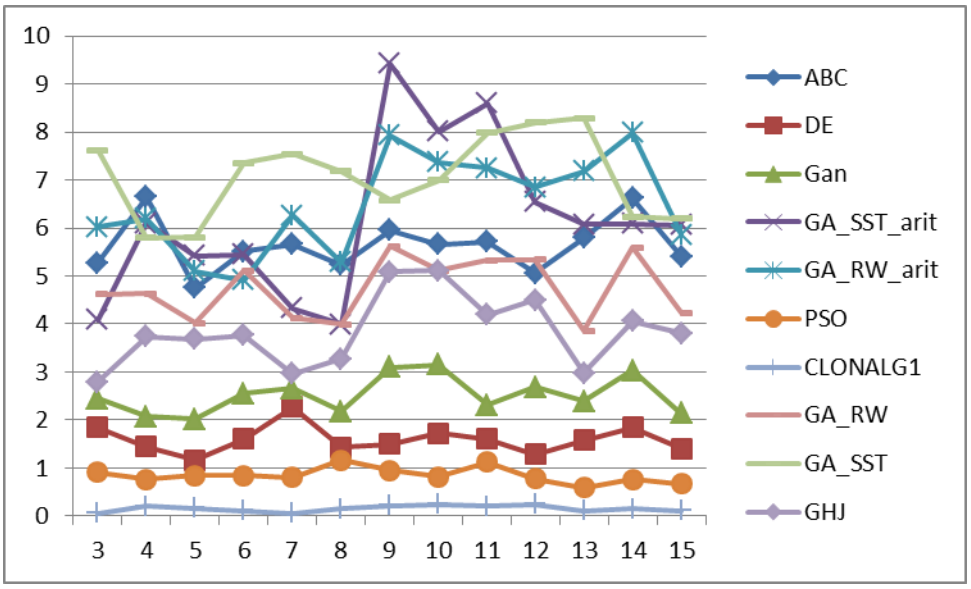
Slika 5-5 Graf ovisnosti pogreške o maksimalnom broju čvorova

Šesti parametar je postotak vjerojatnosti mutacije u nekom stablu, pri njegovom stvaranju nakon križanja njegovih roditelja. Na slici 5.6 može se vidjeti graf koji to opisuje a kao najbolja vrijednost odabrano je 0.17.



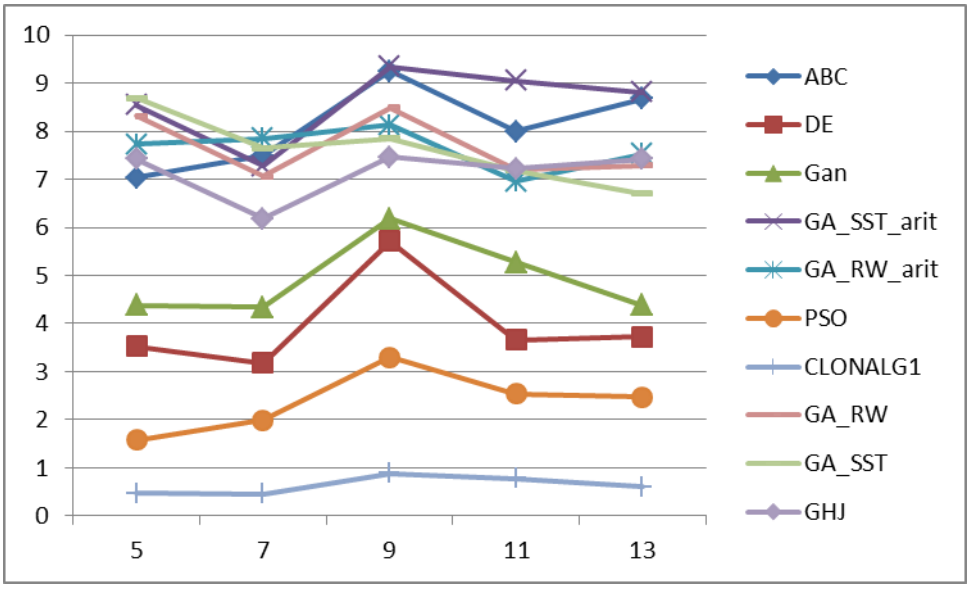
Slika 5-6 Graf ovisnosti pogreške o vjerojatnosti mutacije

Sedmi parametar je količina slučajno odabranih jedinki za turnir izbacivanja, to jest za turnir iz kojeg se bira najlošija jedinka i zamjenjuje novom stvorenom jedinkom. Graf je na slici 5.7, a kao najbolja vrijednost je uzeto 7 jedinki.



Slika 5-7 Graf ovisnosti pogreške o turniru izbacivanja

Zadnji, osmi ispitani parametar je bila maksimalna dubina stabla. Ovo je poprilično slično već ispitanom parametru maksimalnom broju čvorova, a slično je po tome što ako se dopusti prevelika dubina, može se dogoditi da će simbolička regresija biti pogodna samo za rješavanje skupa za učenje i ničega više. Graf se može vidjeti na slici 5.8, a kao najbolja vrijednost odabrano je da maksimalna dubina bude 7.

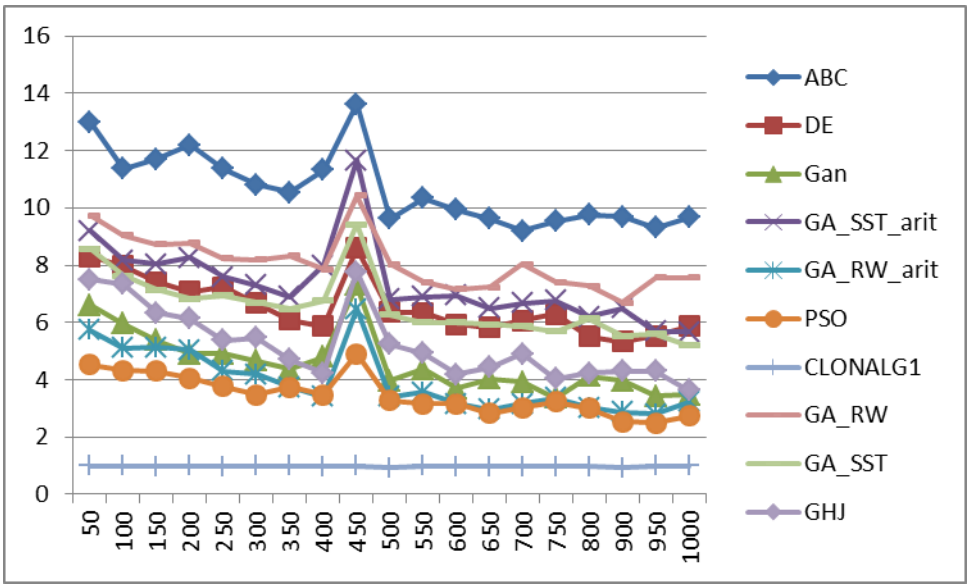


Slika 5-8 Graf ovisnosti pogreške o dubini stabla

5.2 Stablo odluke generirano genetskim programiranjem

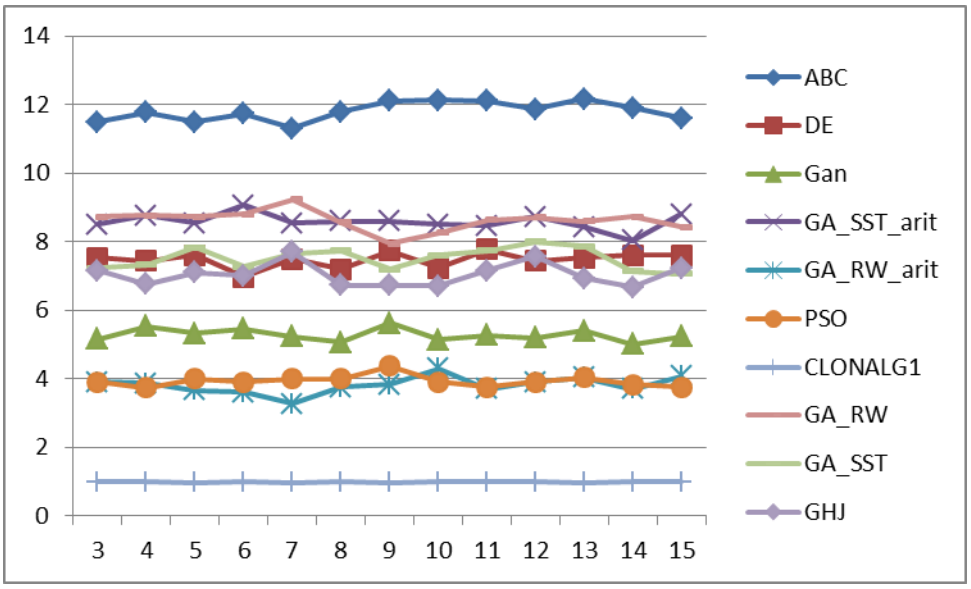
Za stablo odluke generirano genetskim programiranjem ispitani parametri su jako slični onima korištenim u simboličkoj regresiji genetskim programiranjem. Grafovi su rađeni na isti način, na vodoravnoj osi ispitane vrijednosti tog parametra, a na uspravnoj osi pogreška klasifikacije. Ispitano je šest parametara, a već opisani parametri nemaju ponovni opis nego su dane samo dobivene njihove najbolje vrijednosti.

Prvi parametar je broj jedinki populacije. Graf je na slici 5.9, a najbolja vrijednost je 950 jedinki. Kao i kod simboličke regresije i ovo je eliminacijski algoritam.



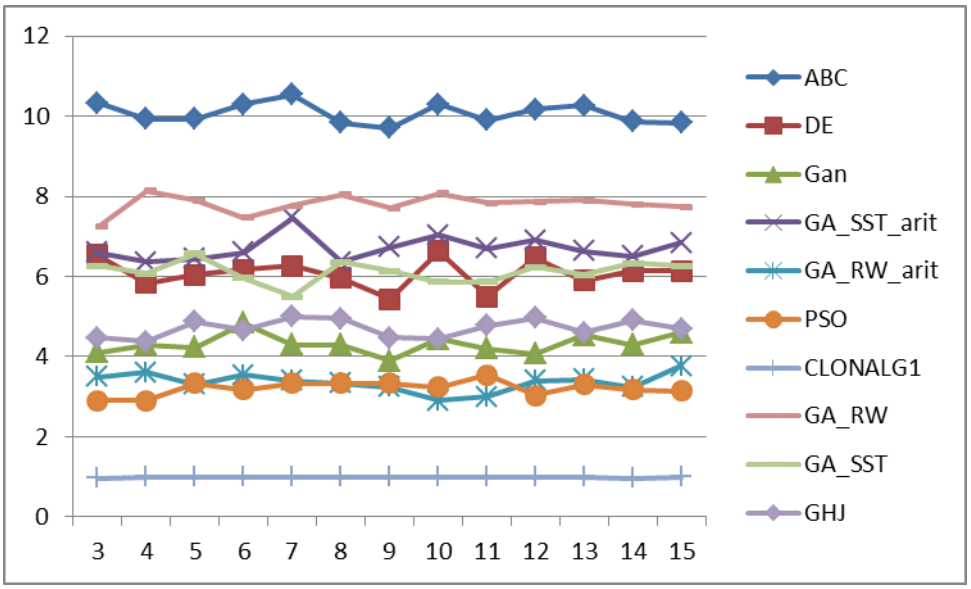
Slika 5-9 Graf ovisnosti pogreške o veličini populacije

Drugi parametar koji je bio ispitan je količina slučajno odabranih jedinki za turnirsku selekciju izbacivanja. Graf se može vidjeti na slici 5.10, a najbolja vrijednost je 14 jedinki.



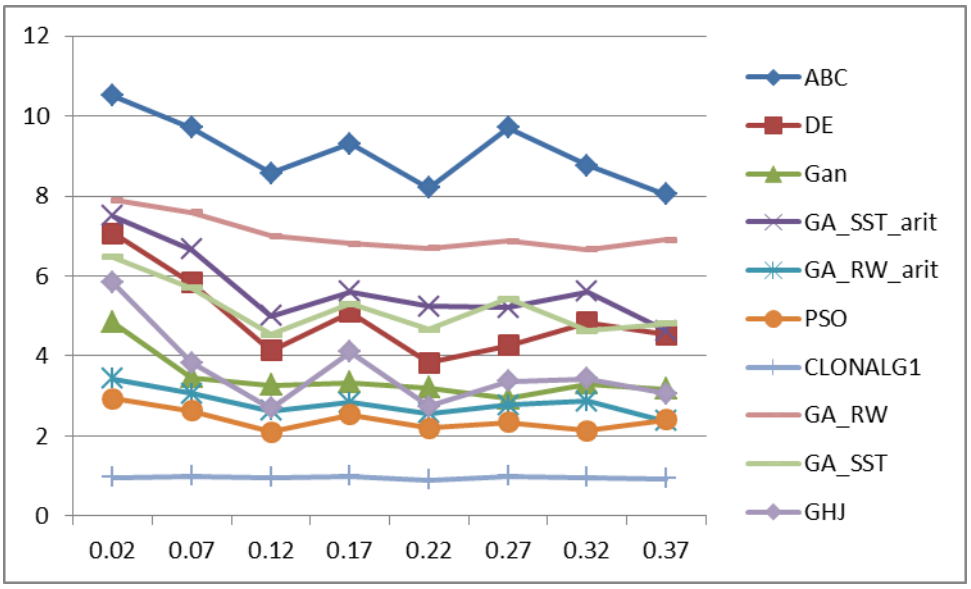
Slika 5-10 Graf ovisnosti pogreške o turniru izbacivanja

Treći parameter koji je bio ispitan je turnirska selekcija za izbor roditelja. Graf se može vidjeti na slici 5.11, a najbolja vrijednost je 9 jedinki.



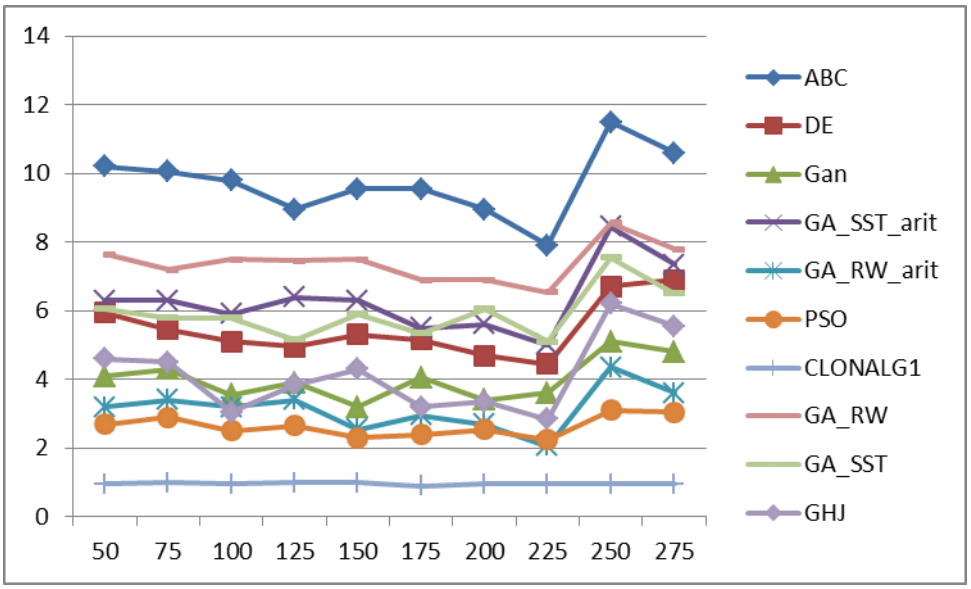
Slika 5-11 Graf ovisnosti pogreške o turnirskoj selekciji

Četvrti ispitni parametar je vjerojatnost mutacije. Graf se može vidjeti na slici 5.12, a najbolja vrijednost je 0.22.



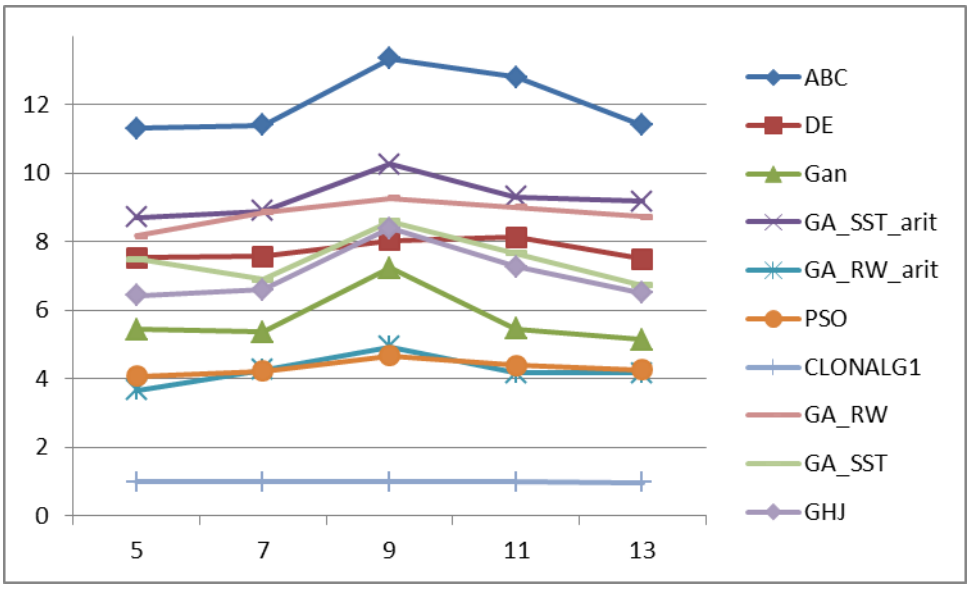
Slika 5-12 Graf ovisnosti pogreške o vjerojatnosti mutacije

Peti testni parametar je broj maksimalno dopuštenih čvorova u stablu. Graf se može vidjeti na slici 5.13, a najbolja vrijednost je 225 čvorova.



Slika 5-13 Graf ovisnosti pogreške o maksimalnom broju čvorova

Zadnji, šesti ispitni parametar je maksimalna dubina stabla. Graf se može vidjeti na slici 5.14, najbolja vrijednost dubine je 7.

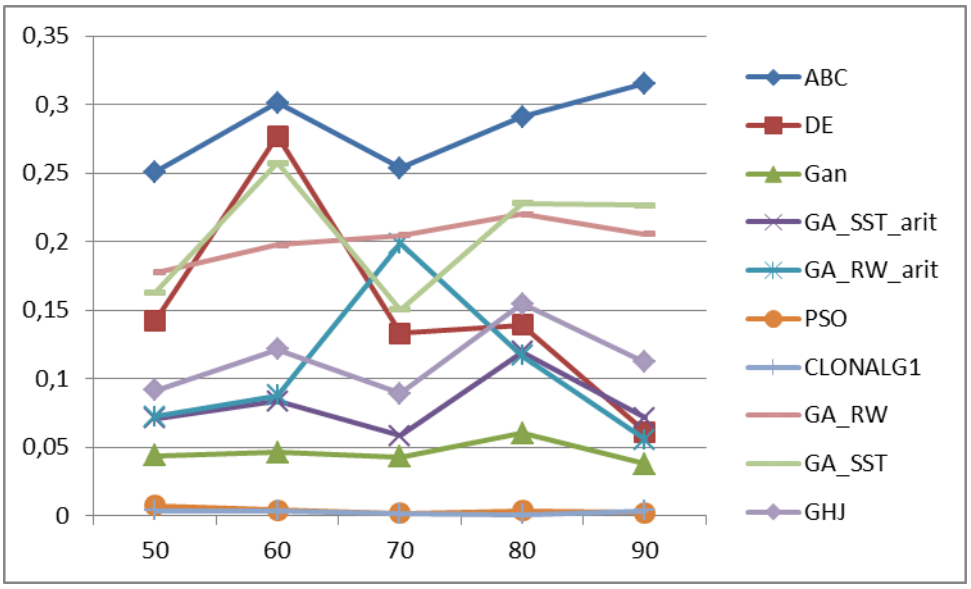


Slika 5-14 Graf ovisnosti pogreške o dubini stabla

5.3 Učenje neuronske mreže diferencijskom evolucijom

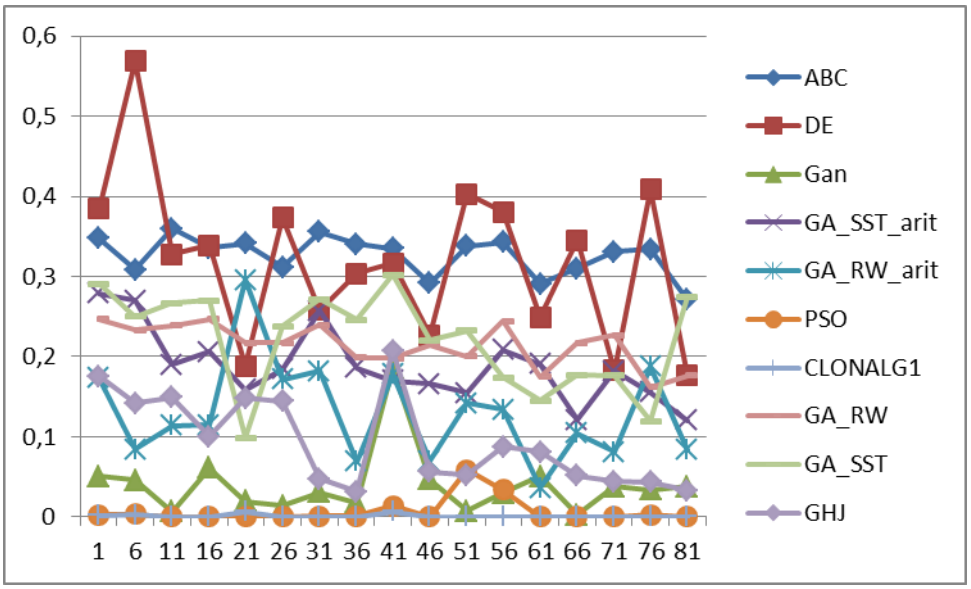
Za diferencijsku evoluciju ispitana su 4 različita parametra. Uz to ispitana je i količina skrivenih čvorova neuronske mreže. Kao i u prethodnim grafovima za genetsko programiranje, na uspravnoj osi su vrijednosti ispitanog parametra, a na vodoravnoj osi je pogreška klasifikacije. Treba napomenuti da se u ovom koraku ispitivanja pogreška računa drugačije u odnosu na stabla dobivena genetskim programiranjem. Pogreška je suma pogreški izlaza neuronske mreže za sve funkcije podijeljena s brojem funkcija.

Prvi parametar predstavlja količinu jedinki u populaciji. Graf se može vidjeti na slici 5.15, a najbolja vrijednost je 50 jedinki.



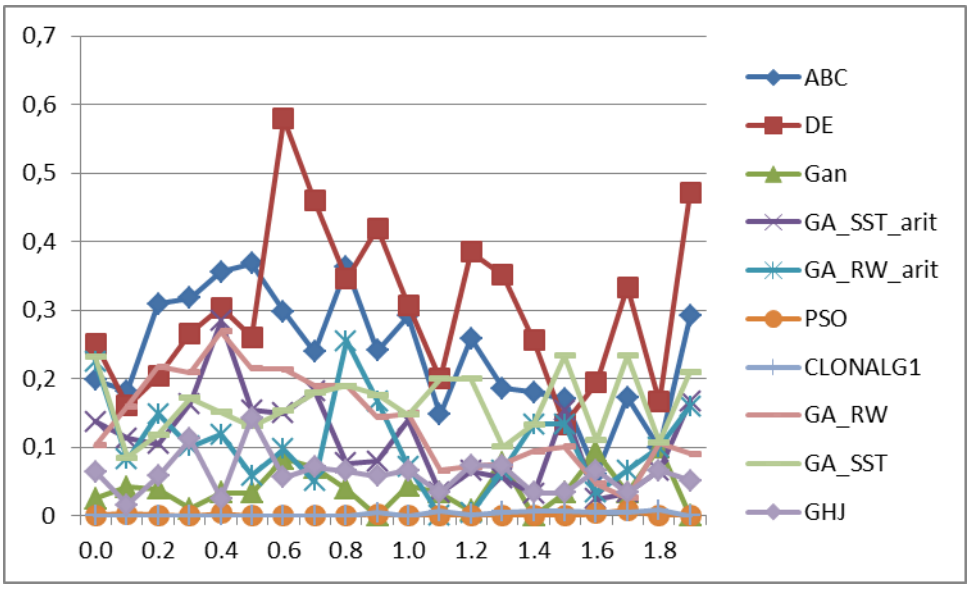
Slika 5-15 Graf ovisnosti pogreške o veličini generacije

Drugi parametar predstavlja radijus iz kojeg se biraju elementi vektora težina neuronske mreže. Kako je genotip svake jedinke baš taj vektor težina, ovaj parametar se koristi pri generiranju početne populacije. Graf se može vidjeti na slici 5.16 a najbolja vrijednost je ±81.



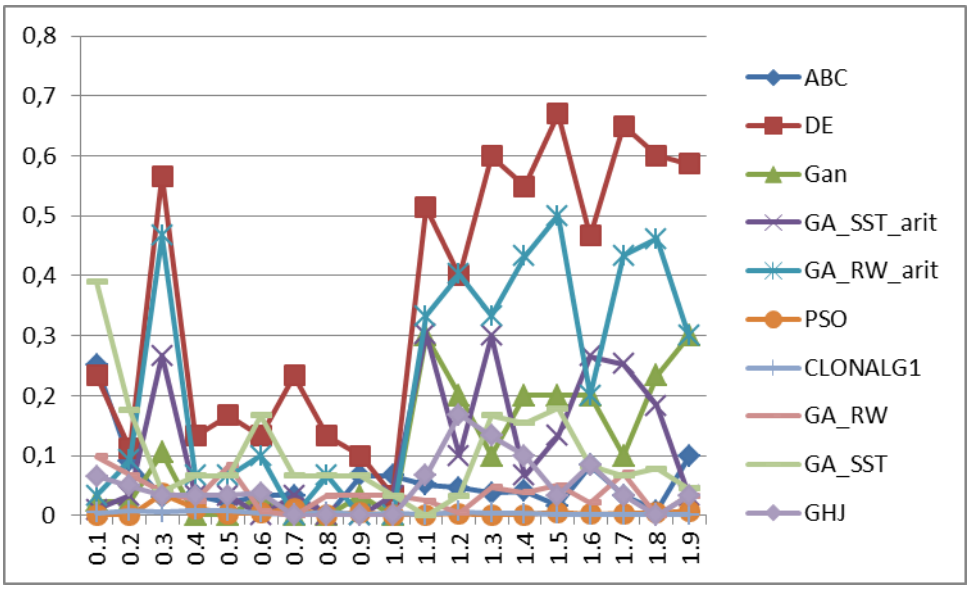
Slika 5-16 Graf ovisnosti pogreške o radijusu

Treći parametar je skalar F koji se množi s razlikom vektora r0 i r1 u postupku mutacije. Graf se može vidjeti na slici 5.17, a za najbolju vrijednost je uzeto 1.6.



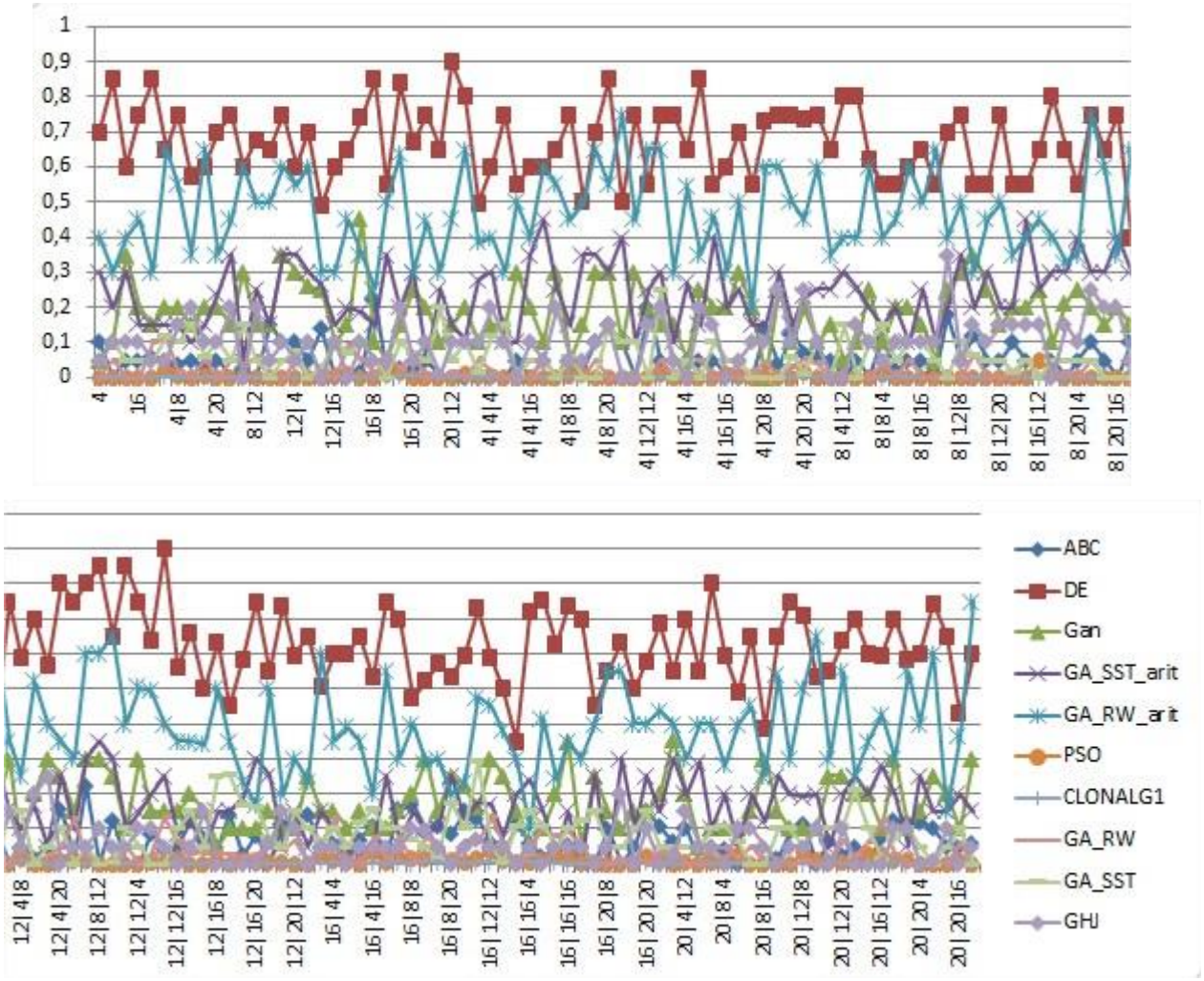
Slika 5-17 Graf ovisnosti pogreške o skalaru F

Četvrti parametar predstavlja vjerojatnost križanja Cr. Graf se može vidjeti na slici 5.18, a za najbolju vrijednost je uzeto 1.



Slika 5-18 Graf ovisnosti pogreške o vjerojatnosti križanja Cr

Zadnje ispitano u okviru diferencijske evolucije je količina skrivenih čvorova neuronske mreže. Ovdje je ispitano puno više mogučnosti što se može vidjeti na grafu na slici 5.19. Na uspravnoj osi nalazi se količina neurona po slojevima skrivenih slojeva. Slojevi su odvojeni sa |. Razmatranjem se odlučilo za skrivene slojeve uzeti 4|20|4 jer je to najbolji omjer manjeg broja čvorova i male pogreške klasifikacije.

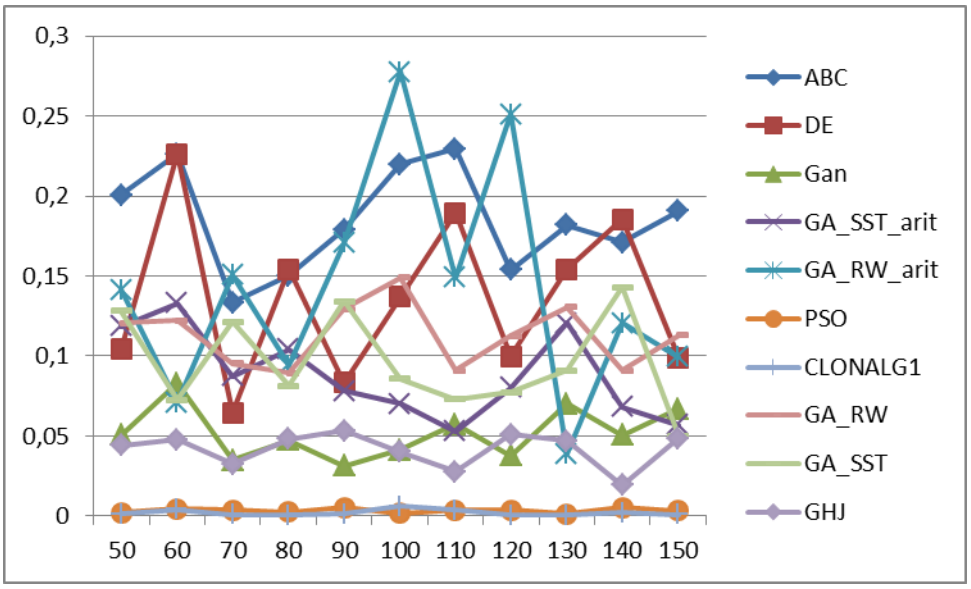


Slika 5-19 Graf ovisnosti pogreške o skrivenim slojevima NN.

5.4 Učenje neuronske mreže algoritmom selekcije klona

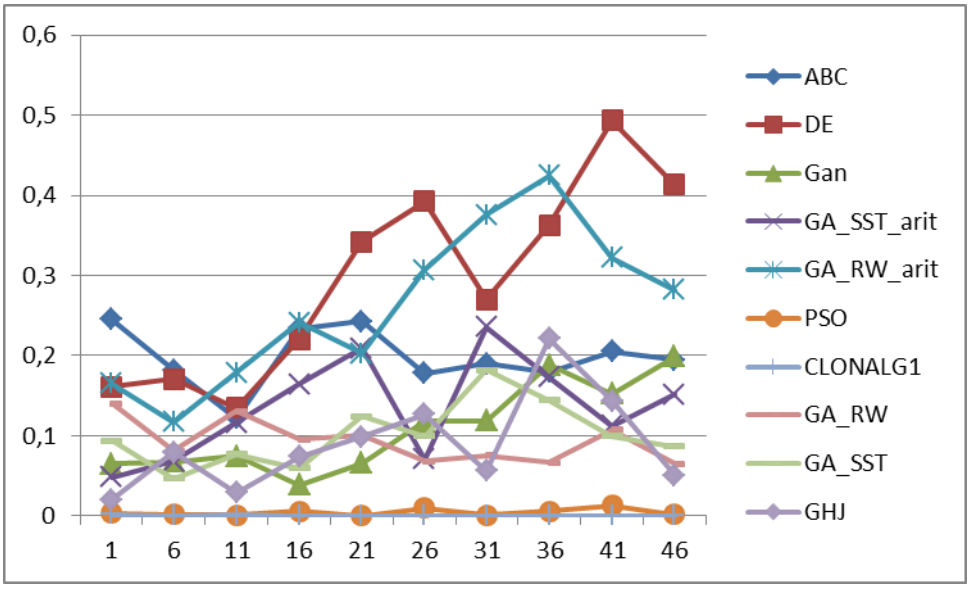
Za algoritam selekcije klona ispitano je 5 parametara. Ovdje se za razliku od diferencijske evolucije nije tražio broj čvorova u skrivenom sloju, nego je u drugoj fazi ispitivanja korišten broj dobiven sa diferencijskom evolucijom. Kao i u prethodnim grafovima, tako se i ovdje na uspravnoj osi nalaze vrijednosti parametara koji se ispituju, dok se na vodoravnoj osi nalazi greška klasifikacije.

Prvi parametar koji je ispitan predstavlja količinu jedinki u populaciji. Graf se može vidjeti na slici 5.20, a za najbolju vrijednost je uzeto 70 jedinki.



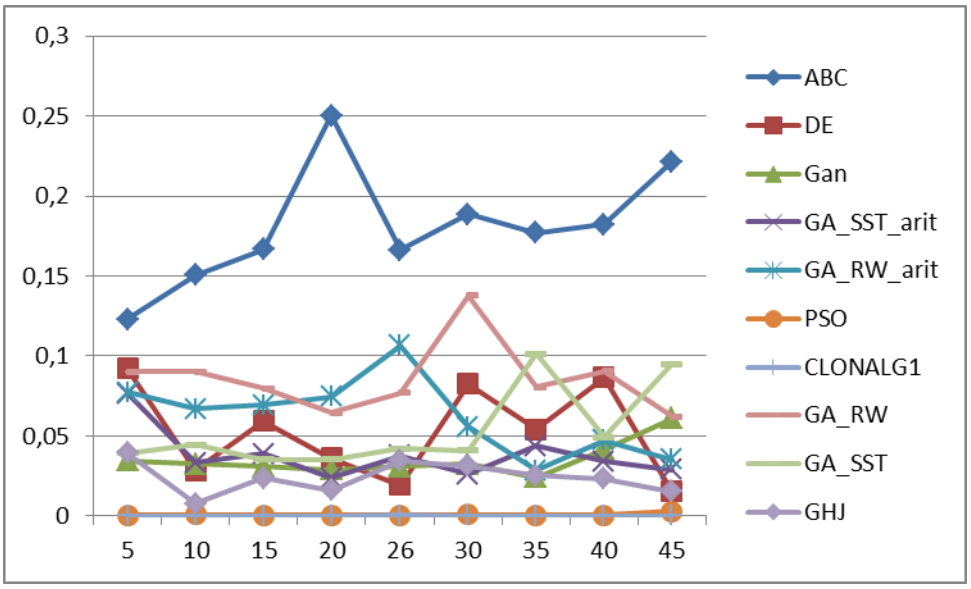
Slika 5-20 Graf ovisnosti pogreške o veličini generacije

Drugi parametar je radijus iz kojega se biraju elementi vektora težina neuronske mreže. Isto kao i kod diferencijske evolucije genotip predstavlja taj vektor težina. Graf se može vidjeti na slici 5.21 a za najbolju vrijednost je uzet radijus ±6.



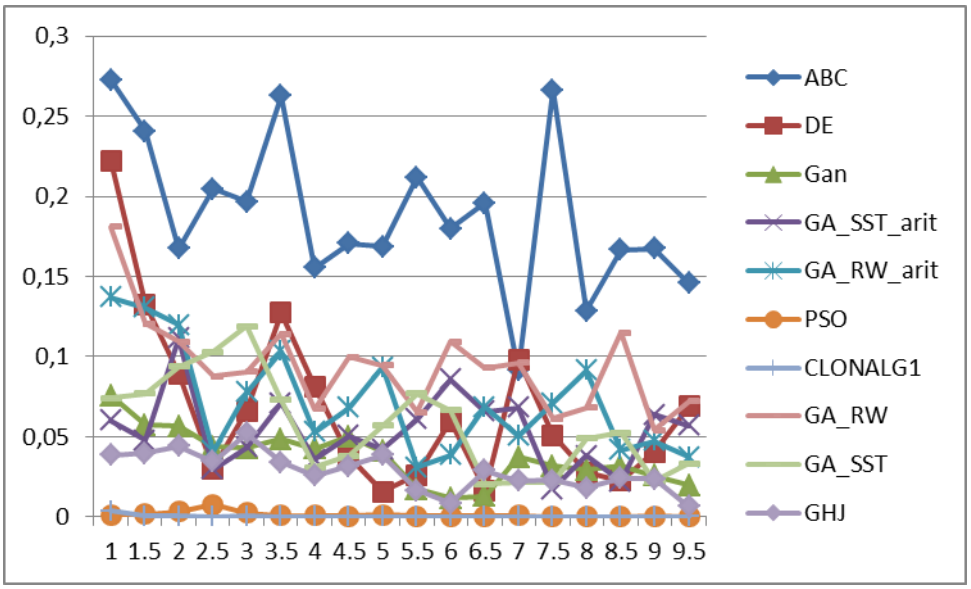
Slika 5-21 Graf ovisnosti pogreške o radijusu

Treći ispitni parametar je parametar D, broj koliko će se novih slučajno stvorenih jedinki dodati u svaku generaciju. Graf se može vidjeti na slici 5.22, a za najbolju vrijednost je uzeto 10.



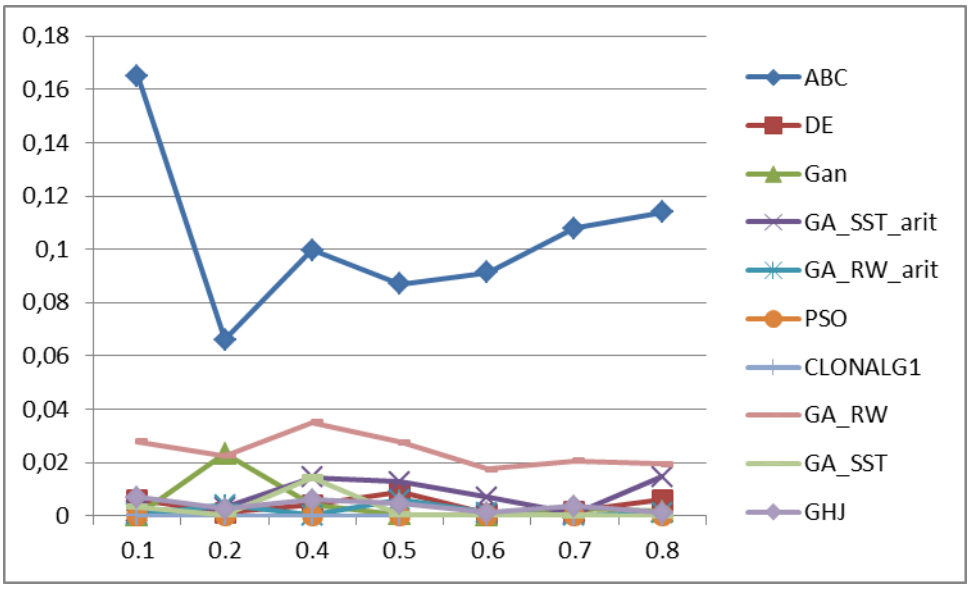
Slika 5-22 Graf ovisnosti pogreške o parametru D

Četvrti testni parametar je parametar β, parametar koji određuje veličinu populacije klonova. Graf se može vidjeti na slici 5.23, a za najbolju vrijednost je uzeto 7.



Slika 5-23 Graf ovisnosti pogreške o parametru β

Zadnji, peti testni parametar, je parametar ρ, parametar hipermutacije. Graf se može vidjeti na slici 5.24, a za najbolju vrijednost je uzeto 0.6.



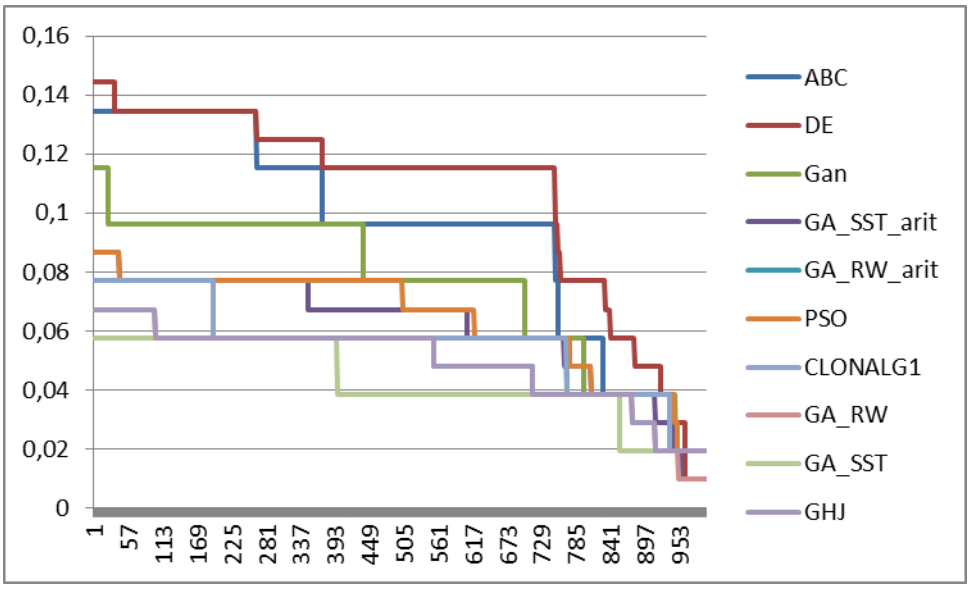
Slika 5-24 Graf ovisnosti pogreške o parametru ρ

6. Usporedba modela i određivanje kriterija zaustavljanja

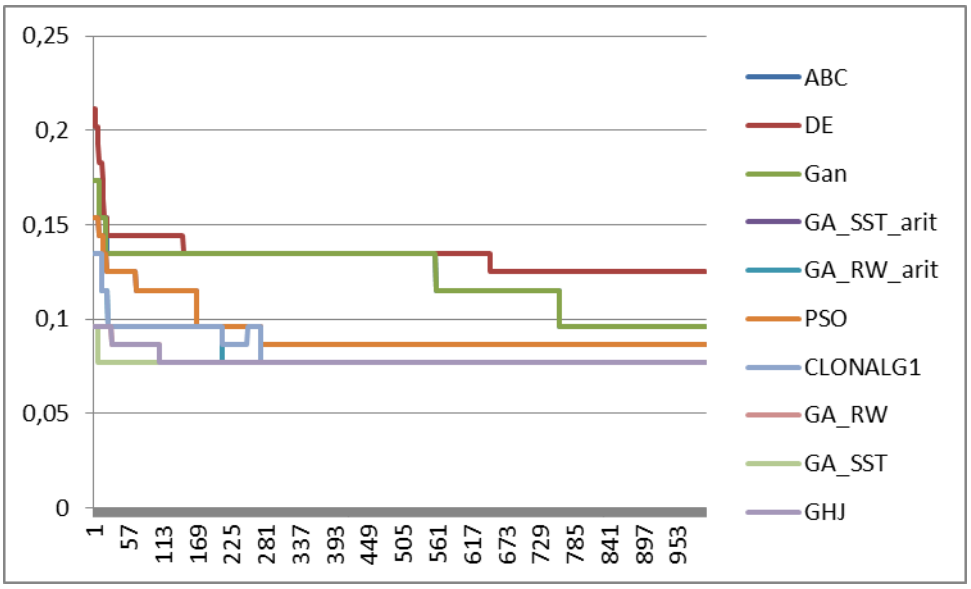
U prethodnoj fazi ispitivanja bilo je bitno odraditi najbolje parametre za sva 4 modela učenja. Tada nije bilo potrebe gledati na pogrešku unakrsne provjere, nego je bilo potrebno gledati pogrešku klasifikacije na skupu za učenje. U ovoj fazi ispitivanja određeno je koliko je svaki od 4 modela učenja zaista dobar za predviđanje klasifikacije po krajoliku dobrote jer se promatra pogreška na skupu za unakrsnu provjeru. Inicijalni skup od 52 funkcije [18, 19] podijeljen je na 2 skupa, jedan za učenje (75% početnog) i jedan za unakrsnu provjeru (25% početnog).

Za sva 4 modela učenja definira se jednaki način izračuna pogreške. Pogreška klasifikacije za zadanu jedinku je broj pogrešno klasificiranih ispitnih funkcija [18,19] podijeljen sa ukupnom količinom ispitnih funkcija. To znači ako određena jedinka pogodi klasifikaciju za sve funkcije iz skupa za unakrsnu provjeru pogreška će biti 0, a ako ne pogodi ni jednu pogreška će biti 1.

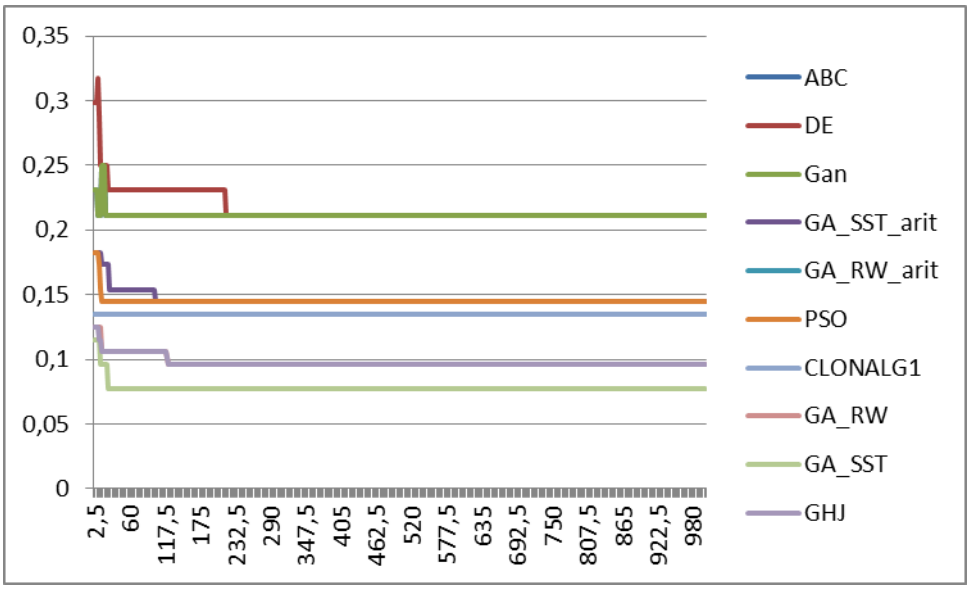
Usporedbe radi dani su grafovi (slike NNNN) koji prokazuju grešku klasifikacije po evaluacijama (radi lakšeg čitanja broj evaluacije je podijeljen sa 100) na skupu za učenje.



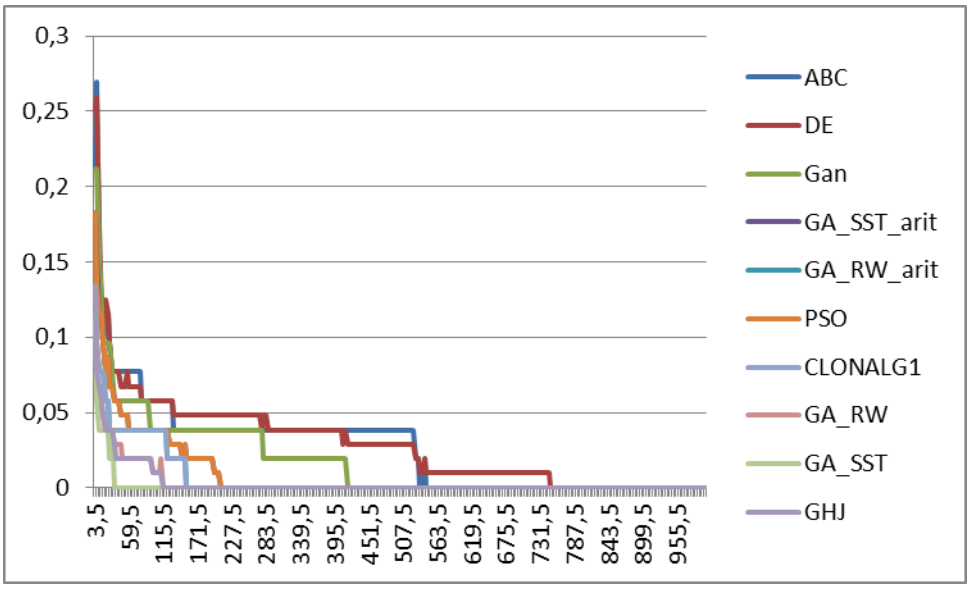
Slika 6-1 Graf greške simboličke regresije na skupu za učenje



Slika 6-2 Graf greške stabla odluke na skupu za učenje



Slika 6-3 Graf greške diferencijske evolucije na skupu za učenje



Slika 6-4 Graf greške algoritma selekcije klona na skupu za učenje

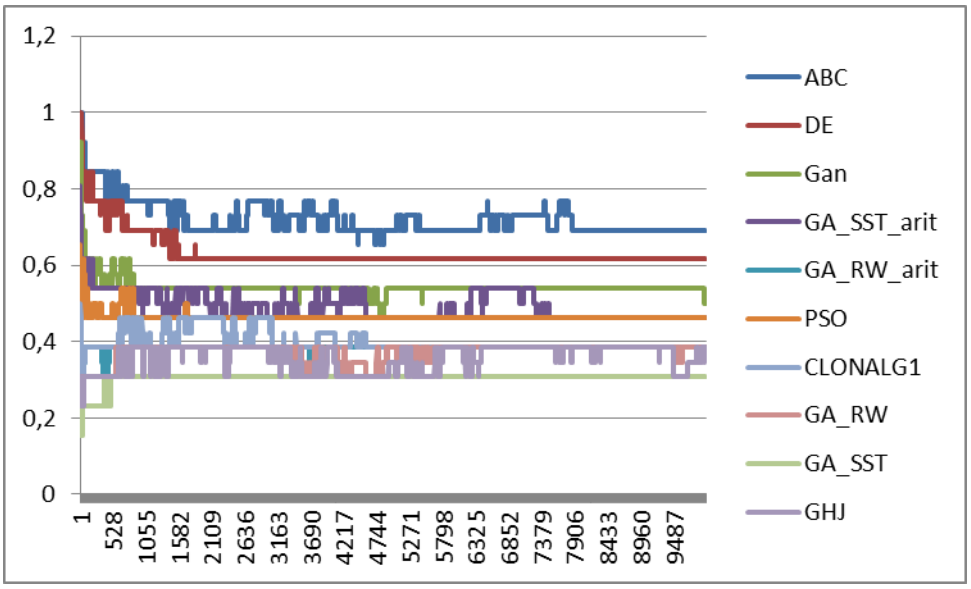
U nastavku su za sve 4 metode učenja prikazani grafovi ocjene klasifikacije. Kako su prikazani stohastički algoritmi, svaka metoda je bila pokrenuta 30 puta, za određeni broj evaluacija. Greška je bilježena svakih nekoliko evaluacija, ovisno o modelu. Za vrijednost svakog ispisa nakon određenog broja evaluacija iz ovih 30 pokretanja, prikazan je medijan od 30 vrijednosti tog zapisa.

Uz pogrešku klasifikacije na skupu za unakrsnu provjeru bilježio se i broj evaluacija bez nađene nove najbolje jedinke. Svrha ovog ispisa je pronalaženje najboljeg kriterija zaustavljanja za svaku od metoda učenja posebno. Navedeni podatci ispisuju se u datoteku zajedno s greškom unakrsne provjere.

Radi jednostavnijeg čitanja grafova na slijedećim grafovima broj evaluacija koji se nalazi na vodoravnoj osi je podiljen sa 100. Broj evaluacija bez promjene koji se u nekim grafovima nalazi na horizontalnoj osi nije dijeljen sa 100.

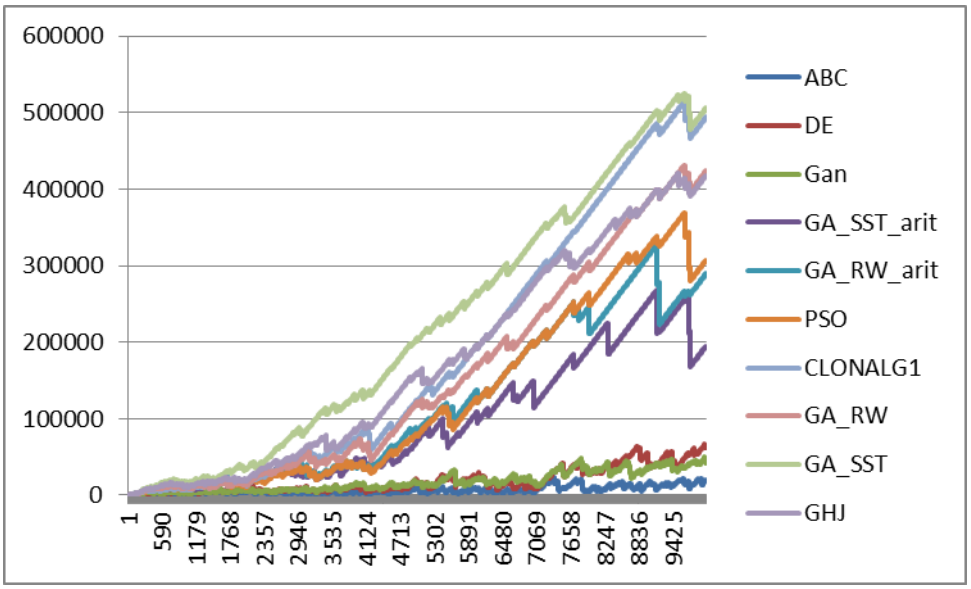
6.1 Simbolička regresija genetskim programiranjem

Graf na slici 6.5 prikazuje grešku klasifikacije na skupu za unakrsnu provjeru. Na grafu se vidi da pogreška u početku varira, a onda se stabilizira i postane otprilike konstantna, pa je zato 85000-ta evaluacija uzeta za mjesto gdje bi se trebalo zaustaviti.



Slika 6-5 Graf greške klasifikacije simboličkom regresijom

Na slici 6.6 se može vidjeti kako se povećava broj evaluacija bez pronalaska nove najbolje jedinke. Krivulje nalikuju na neku slabiju eksponencijalnu funkciju. Ali za kriterij zaustavljanja uzeto je 3000 evaluacija bez promjene jer to odgovara 85000-toj evaluaciji.



Slika 6-6 Graf evaluacija bez pronalaska nove najbolje jedinke simboličke regresije

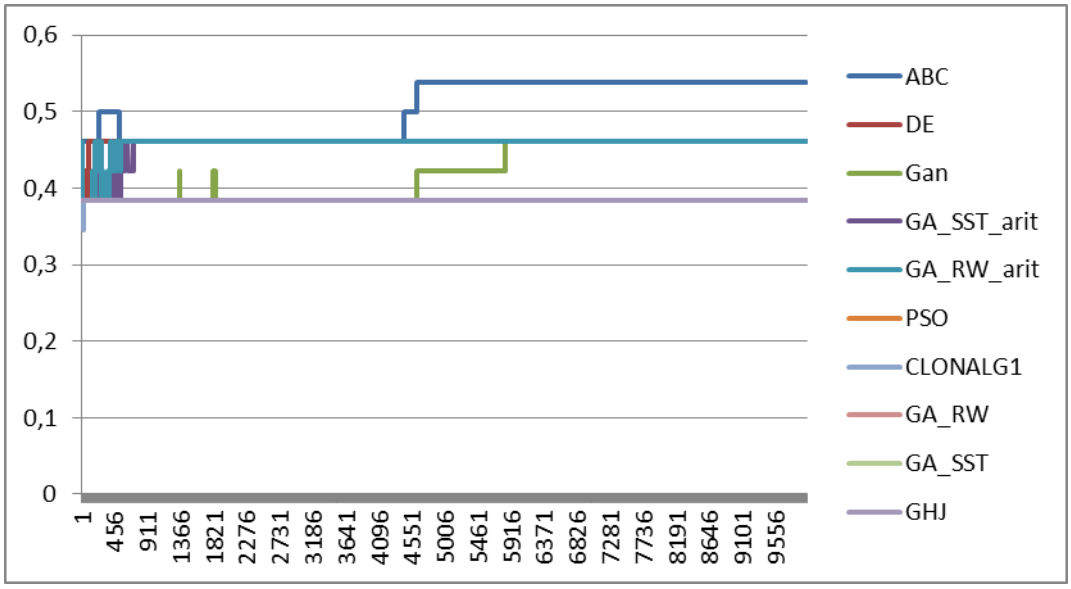
Tablica 6.1 prikazuje najmanju, prosječnu i najveću pogrešku klasifikacije za sve algoritme.

Tablica 6-1 Tablica pogreške klasifikacije simboličkom regresijom

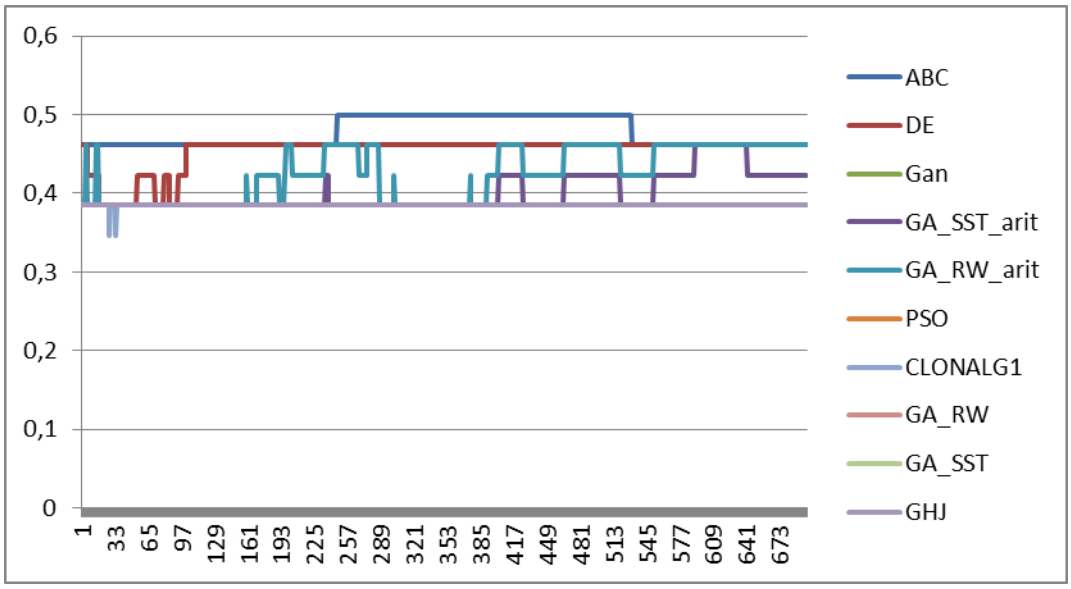
	ABC	DE	Gan	GA_SST_arit	GA_RW_arit	PSO	CLONALG1	GA_RW	GA_SST	GHJ	Ukupno
min	0,653846	0,615385	0,461538	0,461538	0,307692	0,461538	0,307692	0,192308	0,153846	0,230769	0,153846
avg	0,722109	0,632509	0,541662	0,492202	0,384311	0,464335	0,397431	0,371646	0,303814	0,357488	0,466751
max	1	1	0,923077	0,807692	0,538462	0,653846	0,5	0,384615	0,307692	0,384615	1

6.2 Stablo odluke generirano genetskim programiranjem

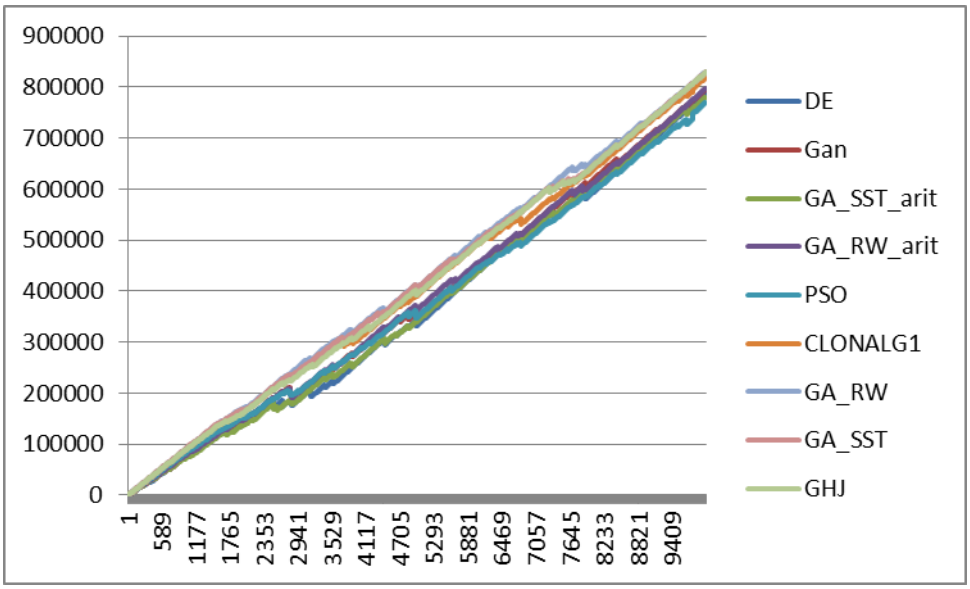
Graf koji prikazuje grešku klasifikacije stablom odluke može se vidjeti na slici 6.7. Iz tog grafa može se lako zaključiti da greška počinje rasti većim brojem evaluacija. Graf na slici 6.8 prikazuje područje od prve do 70000-e evaluacije iz grafa na slici 6.7, s ciljem da se bolje prikažu vrijednosti pogreške. Na grafu 6.9 stavljen je prikaz provedenog vremena, to jest koliko je evaluacija prošlo bez da je nađena nova najbolja jedinka. Taj graf je poprilično linearan.



Slika 6-7 Graf greške klasifikacije stablom odluke



Slika 6-8 Uvećani dio grafa sa slike 6.3



Slika 6-9 Graf evaluacija bez pronalaska nove najbolje jedinke stabla odluke

Kako se iz grafa na slici 6.9 može vidjeti da je greška klasifikacije najmanja za 45000 evaluacija. Pri toj evaluaciji broj evaluacije bez pronalaska nove najbolje jedinke iznosi 3000. Tako da se za kriterij zaustavljanja učenja stabla odluke genetskim programiranjem uzima 3000 evaluacija bez pronalaska nove najbolje jedinke.

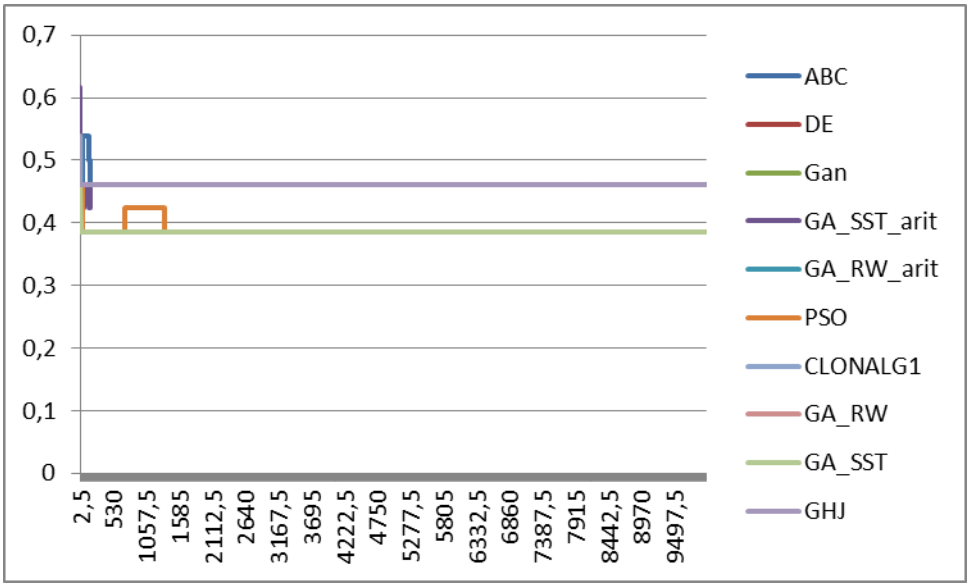
Tablica 6.2 prikazuje najmanju, prosječnu i najveću pogrešku klasifikacije za sve algoritme.

Tablica 6-2 Tablica pogreške klasifikacije stabla odluke

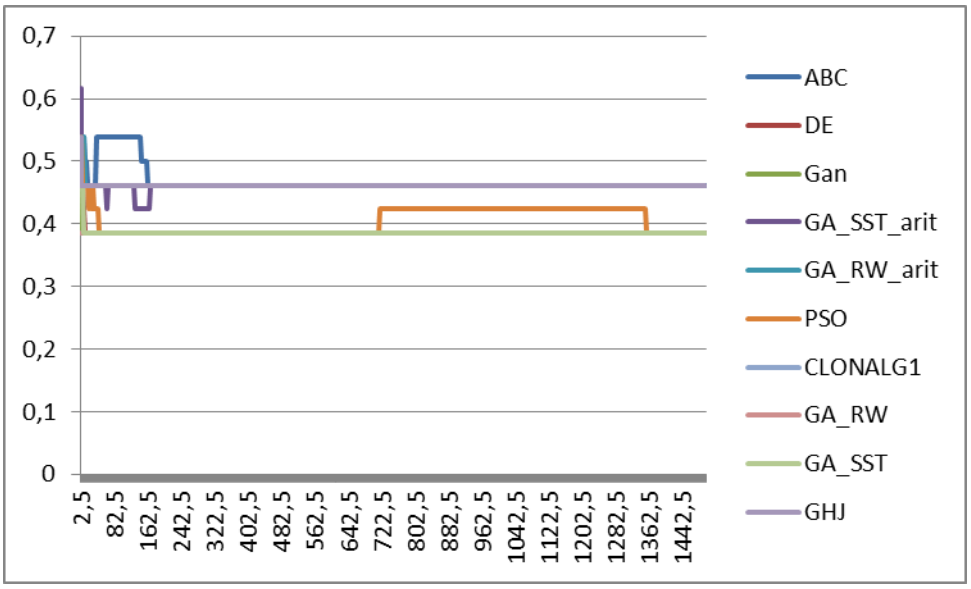
	ABC	DE	Gan	GA_SST_arit	GA_RW_arit	PSO	CLONALG1	GA_RW	GA_SST	GHJ	Ukupno
min	0,461538	0,384615	0,384615	0,384615385	0,384615385	0,384615	0,346153846	0,384615	0,384615	0,384615	0,346154
avg	0,50466	0,460973	0,421415	0,457159349	0,45887944	0,384615	0,384607688	0,384615	0,384615	0,384615	0,422616
max	0,538462	0,461538	0,461538	0,461538462	0,461538462	0,384615	0,384615385	0,384615	0,384615	0,384615	0,538462

6.3 Učenje neuronske mreže diferencijskom evolucijom

Graf na slici 6.10 prikazuje grešku klasifikacije neuronske mreže učene diferencijalnom evolucijom. Iz razloga što je greška promjenjiva do 300000-te evaluacije, a kasnije nema promjena, na grafu na slici 6.11 vidi se dio grafa 6.10 do 150000-te evaluacije. Iz tog grafa se može vidjeti da greška zapravo malo i raste nakon 25000-te evaluacije.

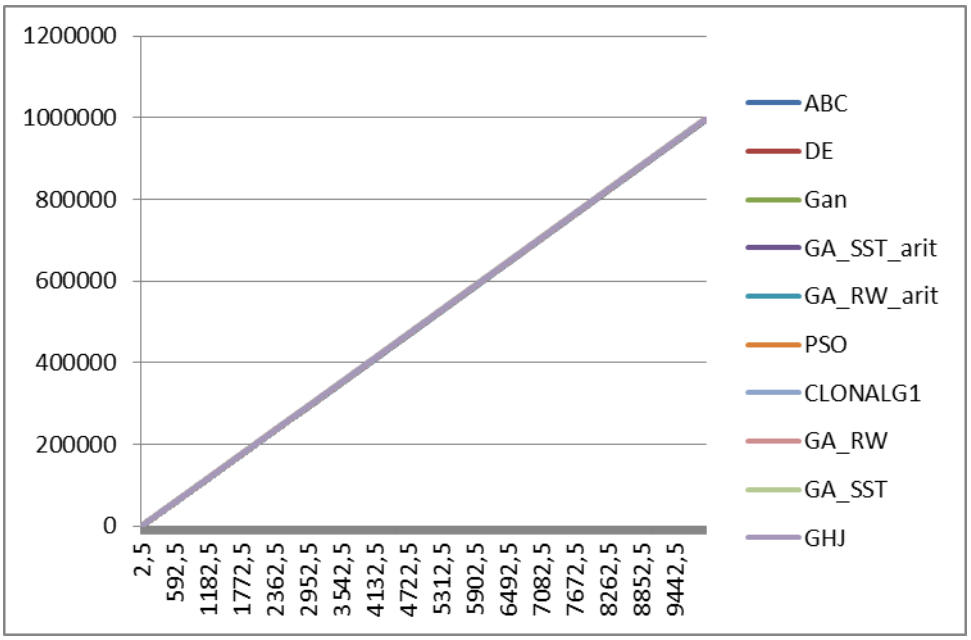


Slika 6-10 Graf greške klasifikacije diferencijske evolucije



Slika 6-11 Uvećani dio grafa sa slike 6.6

Iz grafa na slici 6.12 može se vidjeti porast evaluacija bez pronalaska nove najbolje jedinke. Taj graf je linearan, a vrijednost za 25000 evaluacija je 20000 evaluacija bez promjene. Stoga je za kriterij zaustavljanja diferencijske evolucije uzeto 20000 evaluacija bez promjene.



Slika 6-12 Graf evaluacija bez pronalaska nove najbolje jedinke diferencijske evolucije

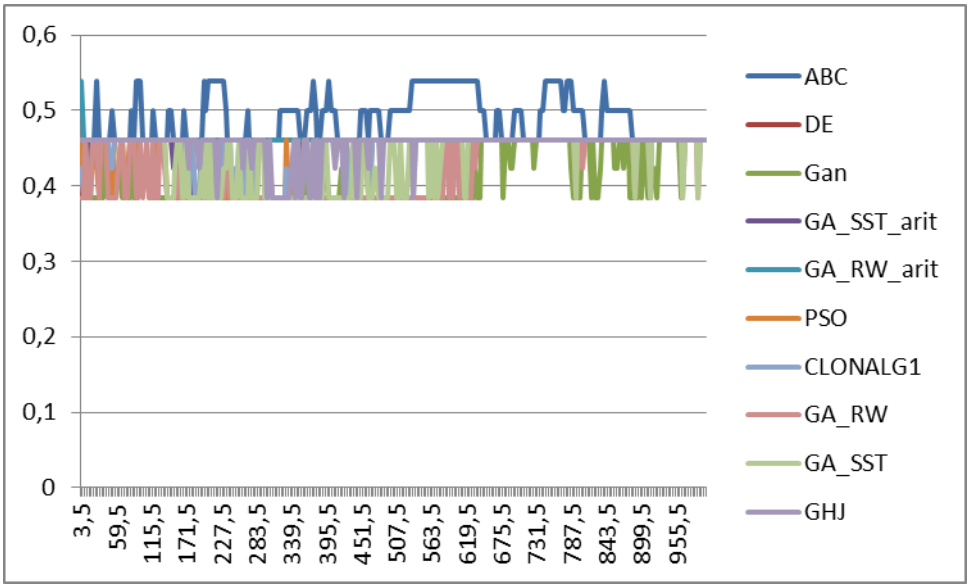
Tablica 6.3 prikazuje najmanju, prosječnu i najveću pogrešku klasifikacije za sve algoritme.

Tablica 6-3 Tablica pogreške klasifikacije diferencijske evolucje

	ABC	DE	Gan	GA_SST_arit	GA_RW_arit	PSO	CLONALG1	GA_RW	GA_SST	GHJ	Ukupno
min	0,538462	0,384615	0,384615	0,461538	0,461538	0,384615	0,384615	0,384615	0,384615	0,461538	0,384615
avg	0,53849	0,458856	0,384712	0,461654	0,461663	0,391856	0,384644	0,384615	0,385087	0,461558	0,431313
max	0,576923	0,461538	0,5	0,615385	0,615385	0,538462	0,461538	0,384615	0,538462	0,538462	0,615385

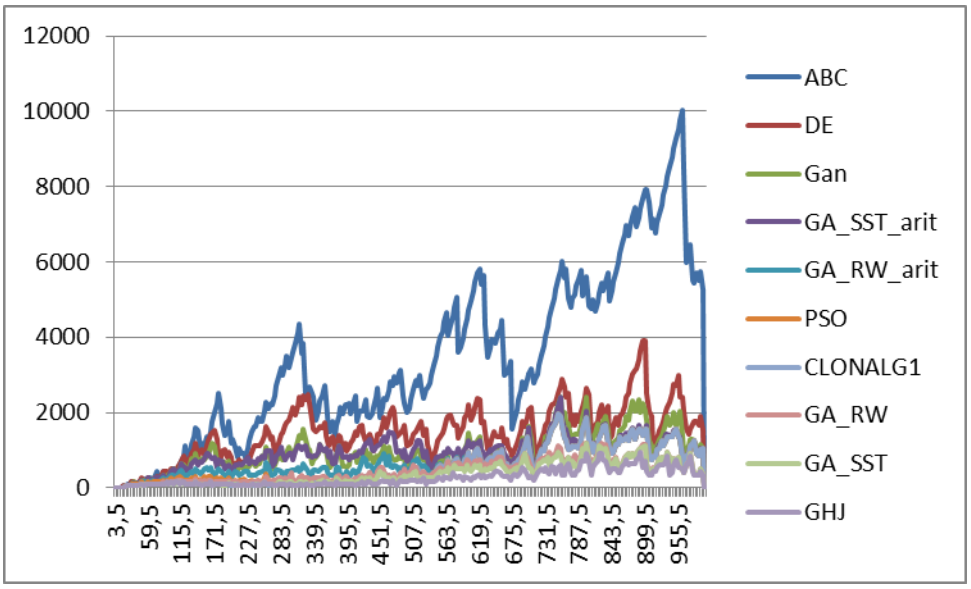
6.4 Učenje neuronske mreže algoritmom selekcije klona

Graf na slici 6.13 prikazuje grešku klasifikacije neuronske mreže učene od strane algoritma selekcije klona. Greška varira veliki dio grafa, ali izgleda da se stabilizira nakon 94500 evaluacija.



Slika 6-13 Graf greške klasifikacije algoritma selekcije klona

Graf na slici 6.14 prikazuje porast broja evaluacija bez pronalaska nove najbolje jedinke. Zbog velike varijacije za zaustavljanje je uzeto 700 evaluacija bez pronalaska nove najbolje jedinke prema vrijednostima grafova 6.9 i 6.10.



Slika 6-14 Graf evaluacija bez pronalaska nove najbolje jedinke algoritma selekcije klona

Tablica 6.4 prikazuje najmanju, prosječnu i najveću pogrešku klasifikacije za sve algoritme.

Tablica 6-4 Tablica pogreške klasifikacije algoritma selekcije klona

	ABC	DE	Gan	GA_SST_arit	GA_RW_arit	PSO	CLONALG1	GA_RW	GA_SST	GHJ	Ukupno
min	0,461538	0,384615	0,384615	0,384615	0,461538	0,384615	0,384615	0,384615	0,384615	0,384615	0,384615
avg	0,487493	0,461001	0,409763	0,457504	0,461807	0,441904	0,447956	0,419715	0,43518	0,45468	0,4477
max	0,538462	0,461538	0,461538	0,461538	0,538462	0,461538	0,461538	0,461538	0,461538	0,461538	0,538462

6.5 Usporedba modela na algoritmima

Za zaključivanje je li ABC dobar algoritam za optimizaciju neke funkcije najbolje određuje neuronska mreža učena algoritmom klonske selekcije uz prosječnu točnost od 52%. Za zaključivanje je li DE dobar za optimizaciju neke funkcije njabolje određuje neuronska mreža učena diferencijalnom evolucijom uz prosječnu točnost od 55%. Za zaključivanje je li Gan dobar algoritam za optimizaciju neke funkcije najbolje određuje neuronska mreža učena diferencijalnom evolucijom uz prosječnu točnost od 62%. Za zaključivanje je li GS_SST_arit dobar algoritam za optimizaciju neke funkcije najbolje određuje stablo odluke generirano genetskim programiranjem uz prosječnu točnost od 65%. Za zaključivanje je li GS_RW_arit dobar algoritam za optimizaciju neke funkcije najbolje određuje simbolička regresija nastala genetskim programiranjem uz prosječnu točnost od 62%. Za zaključivanje je li PSO dobar algoritam za optimizaciju neke funkcije najbolje određuje stablo odluke generirano

genetskim programiranjem uz prosječnu točnost od 62%. Za zaključivanje je li CLONALG dobar algoritam za optimizaciju neke funkcije najbolje određuje stablo odluke generirano genetskim programiranjem uz prosječnu točnost od 62%. Za zaključivanje je li GA_RW dobar algoritam za optimizaciju neke funkcije najbolje određuje simbolička regresija nastala genetskim programiranjem uz prosječnu točnost od 62%. Za zaključivanje je li GS_SST dobar algoritam za optimizaciju neke funkcije najbolje određuje simbolička regresija nastala genetskim programiranjem uz prosječnu točnost od 70%. Za zaključivanje je li GHJ dobar algoritam za optimizaciju neke funkcije najbolje određuje simbolička regresija nastala genetskim programiranjem uz prosječnu točnost od 65%.

7. Zaključak

Ovaj se rad bavi istraživanjem automatske klasifikacije koja daje odgovor na pitanje koliko je neki optimizacijski algoritam dobar za određene funkcije cilja. Korištene su različite metode učenja nad skupom od 52 ispitne funkcije. Uspješnosti ostvarenih metoda učenja mogu se vidjeti u prethodnom poglavlju.

Po brzini, to jest po najmanjem broju evaluacija do zaustavljanja**,** najbrže je učenje neuronske mreže diferencijskom evolucijom (25000 evaluacija). Dugo je stablo odlike generirano genetskim programiranjem (70000 evaluacija). Treći i četvrti su simbolička regresija (85000 evaluacija) i učenje neuronske mreže algoritmom selekcije klona (94500 evaluacija).

Kao najkonzistentnija metoda učenja i metoda učenja sa najmanjom pogreškom izdvaja se stablo odluke generirano genetskim programiranjem, a kao najmanje konzistentan algoritam sa vrijednostima iz skoro cijele kodomene (0-1) javlja se simbolička regresija nastala genetskim programiranjem.

U daljem istraživanju smatram da bi se ponajprije trebalo prikupiti veći skup podataka za učenje / unakrsnu provjeru. Trebao bi se odrediti bolji algoritam klasifikacije algoritama iz skupa za učenje, koji bi za usporedbu uz pogrešku koristio i logaritamsku pogrešku. Simbolička regresija genetskim programiranjem i neuronska mreža učena algoritmom selekcije klona bi se mogle izbaciti kao modeli učenja zbog velike varijacije pogreške i velikog broja potrebnih evaluacija do konvergencije.

8. Literatura

[1] E.G. Talbi: Metaheuristics – From Design to Implementation, Wiley, 2009.

[2] D. W. Wolpert and W. G. Macready: No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[3] E. Pitzer and M. A Enzeller: A comprehensive survey on fitness landscape analysis. In J. Fodor, R. Klempous, and C. Suarez Araujo, editors, Recent Advances in Intelligent Engineering Systems, 2012.

[4] http://en.wikipedia.org/wiki/Artificial_neural_network

[5] Marko Čupić: Prirodom inspirirani optimizacijski algoritmi. Metaheuristike, 2013.

[6] http://en.wikipedia.org/wiki/*Sigmoid_function*

[7] http://en.wikipedia.org/wiki/*Differential_evolution*

[8] http://en.wikipedia.org/wiki/*Clonal_Selection_Algorithm*

[9] Heder S. Bernardino, Leonardo G. Fonseca and Helio J. C. Barbosa: Surrogate-Assisted Artificial Immune Systems for Expensive Optimization Problems, ISBN 978-953-307-008-7, CC BY-NC-SA 3. License, 2009.

[10] http://en.wikipedia.org/wiki/*Genetic_programming*

[11] Koza, J. R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: The MIT Press. 1992.

[12] http://www.geneticprogramming.com/Tutorial/

[13] http://www.zemris.fer.hr/~yeti/studenti/Uvod_u_genetsko_programiranje.pdf

[14] http://en.wikipedia.org/wiki/*Decision_tree*

[15] http://www.mafy.lut.fi/EcmiNL/older/ecmi35/node70.html

[16] Brownlee J: Clonal Selection Algorithms, Technical Report 070209A Complex Intelligent Systems Laboratory, Centre for Information Technology Research, Faculty of Information Communication Technology, Swinburne University of Technology Melbourne, Australia, CIS Technical Report 070209A, 2007.

[17] Picek, S: From Fitness Landscape to Crossover Operator Choice, 2014.

[18] http://coco.gforge.inria.fr/doku.php

[19] http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm

[20] en.wikipedia.org/wiki/*Boolean_satisfiability_problem*

[21] http://en.wikipedia.org/wiki/*K-means_clustering*

[22] http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions

Učinkovitost optimizacijskih algoritama u ovisnosti o značajkama problema

Sažetak

Ovaj rad se bavi ispitivanjem dali se mogu klasificirati optimizacijski problemi i onda odrediti dali je neki algoritam optimizacije dobar za optimiranje određene klase problema, sa ciljem smanjenja vremena potrebnog za određivanjem dobrog optimizacijskog algoritma na novom optimizacijskom problemu. Algoritmi se pokušavaju klasificirati pomoću krajolika dobrote tih problema. Testne funkcije korištene u radu grupirane su u klase pomoću K-means algoritma. U ovom radu korištena su 4 modela klasifikacije strojnim učenjem: stablo odluke generirano genetskim programiranjem, simbolička regresija generirana genetskim programiranjem, učenje neuronske mreže diferencijskom evolucijom i učenje neuronske mreže algoritmom selekcije klona. Ispitivanje se izvodi u 2 faze, u prvoj fazi se određuju najbolji parametri za svaki od modela učenja, dok se u drugoj fazi uspoređuju modeli i procjenjuje se njihova uspješnost klasifikacije.

Ključne riječi

Krajolik dobrote, algoritmi optimizacije, evolucijski algoritmi, strojno učenje, neuronska mreža, genetsko programiranje, diferencijska evolucija, algoritam selekcije klona, simbolička regresija, stablo odluke

Optimization efficiency based on fitness landscape

Summary

This paper is concerned with examining whether you can classify an optimization problem and then determine is some optimization algorithm good for optimizing that problem with goal of reducing time required for determining that. Algorithms where classified using fitness landscape values. Fitness landscape values were calculated from optimisation problems. Test functions where classified using k-means algorithm. In this paper 4 classification models were used: symbolic regression generated by genetic programming, decision three generated by genetic programming, neural network trained by differential evolution and finaly neural network trained by clonal selection algorithm. Testing was done in 2 faze. In first faze best parameters for all modes where determent, while in the second phase models were compared and evaluated by its performance in classification.

Key words

Fitness landscape, optimization algorithms, evolution algorithms, machine learning, neural network, genetic programming, differential evolution, clonal selection algorithm, symbolic regression, decision tree