Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

NEIZRAZITO, EVOLUCIJSKO I NEURO RAČUNARSTVO

IZVJEŠTAJ ZA 3. DIO PROJEKTNOG ZADATKA

Karlo Knežević
0036443568

Zagreb, siječanj 2013.

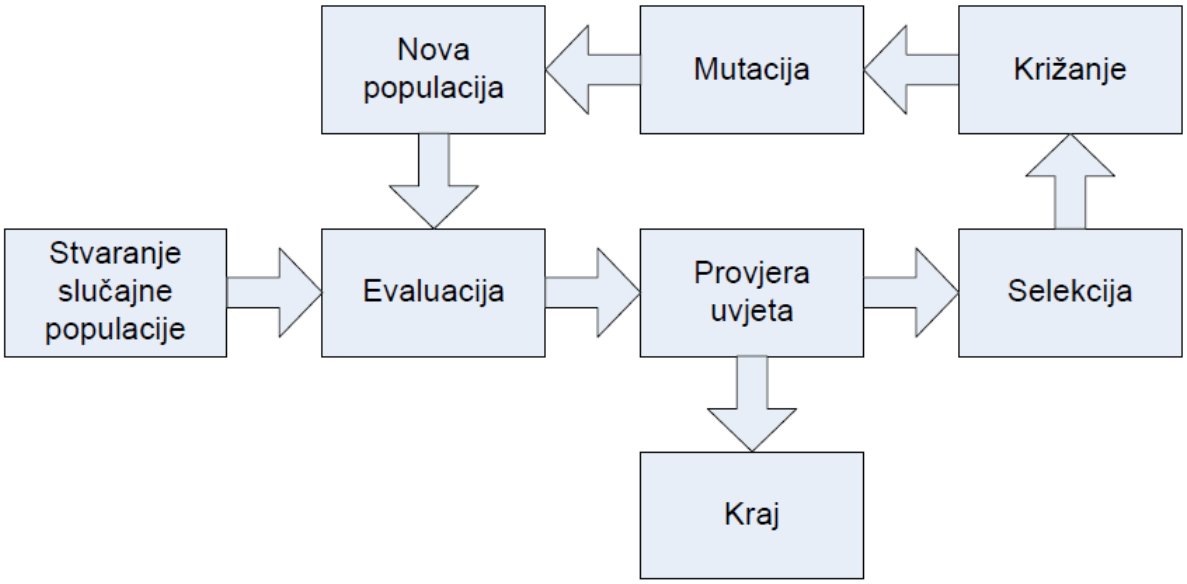
Sadržaj

0. Opći podaci i implementacija

Treći dio projektnog zadatka implementiran je u programskoj razvojnoj okolini *Eclipse*, u programskom jeziku *Java*.
Za potrebe vizualizacije korištena je razvojna okolina i jezik *Matlab*.

1. Zadatak

U sklopu 3. dijela projekta, implementiran je **genetski generacijski algoritam.** Slika 1. prikazuje shemu rada generacijskog genetskog algoritma.



Slika 1. Shema rada generacijskog genetskog algoritma[1].

Pseudokod generacijskog genetskog algoritma prikazan je slikom 2.

```
P = stvori_početnu_populaciju(veličina_populacije)
evaluiraj (P)
ponavljaj_dok_nije_promijenjen_određen_broj_generacija:
        nova_populacija P' = 0
        ponavljaj_dok_je veličina(P') < veličina_populacije:
                odaberi R1 i R2 iz P
                {D1, D2} = križaj(R1, R2)
                mutiraj D1
                mutiraj D2
                dodaj D1 i D2 u P'
        kraj
        P = P'
kraj
```

Slika 2. Pseudokod generacijskog genetskog algoritma.

[1] Čupić, Marko. Prirodom inspirirani optimizacijski algoritmi; Metaheuristike. Zagreb, 2012.

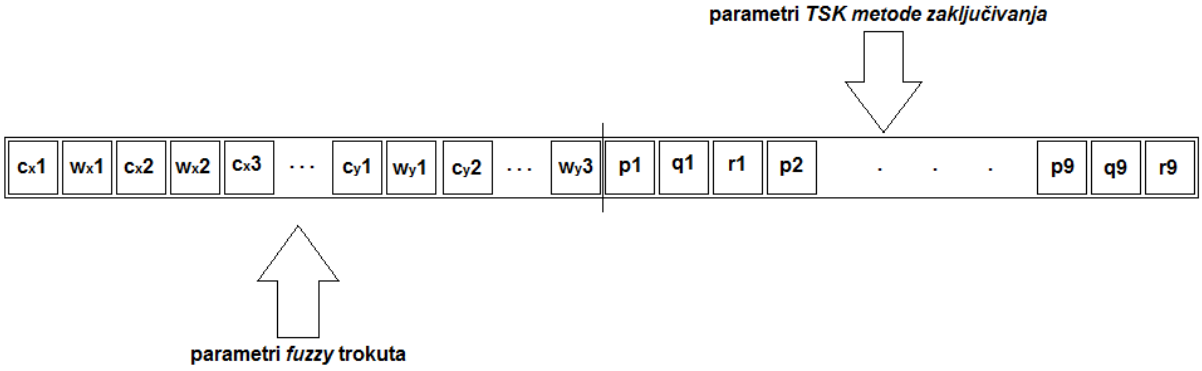
Prikaz rješenja (jedinke, kromosoma) napravljen je pomoću **vektora realnih brojeva**. Vektor se sastoji od 39 polja u kojima su zapisane vrijednosti:

- c (središte *fuzzy* trokuta),
- w (pola širine *fuzzy* trokuta),
- p, q i r (parametri konsekvensa, TSK metode zaključivanja).

Kromosom u konstruktoru prima:

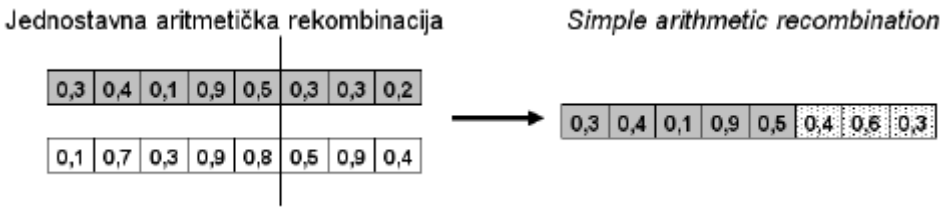
- dekoder kromosoma
    - iz vektora realnih vrijednosti stvara pravila formata koji prepoznaje *fuzzy-neuronska* mreža,
- informaciju o tipu zaključka (konstantan ili funkcija od ulaznih vrijednosti),
- generator pseudoslučajnih brojeva,
- intervale određenih gena kromosoma.

Pomoću generatora pseudoslučajnih brojeva i intervala (koje zadaje korisnik), inicijalizira se realni vektor. Slika 3. prikazuje kromosom i semantiku gena kromosoma.



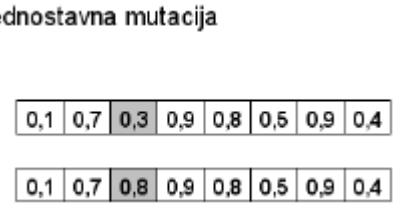
Slika 3. Prikaz kromosoma genetskog algoritma.

Genetski operator **križanja** ostvaren je jednostavnom aritmetičkom rekombinacijom (hijazma u jednoj točki). Hijazma se odabire na slučajan način u intervalu veličine kromosoma. Slika 4. prikazuje jednostavnu aritmetičku rekombinaciju.



Slika 4. Jednostavna aritmetička rekombinacija.

Genetski operator **mutacije** ostvaren je jednostavnom mutacijom, gdje se mutirani gen stvara slučajnim odabirom realnog broja u intervalu zadanom u konstruktoru. Slika 5. prikazuje jednostavnu mutaciju.



Slika 5. Jednostavna mutacija

Genetski operator **selekcije** ostvaren je proporcionalnom selekcijom („roulette wheel"). Vjerojatnost odabira svake jednike proporcionalan je njezinij dobroti. Da bi se riješio problem „sličnih" dobrota, koristi se relativna dobrota, odnosno kao efektivna dobrota svake jedinke uzima se razlika između dobrote jednike i dobrote najgore jednike.

1.1 Vrijednosti inicijalnih parametara genetskog algoritma

Višestrukim pokretanjem algoritma te uzimanjem u obzir prihvatljivih kriterija za odabir parametara, odabrani su sljedeće vrijednosti:

- vjerojatnost križanja, $\chi = 0.7$
- vjerojatnost mutacije, $\mu = 0.016$
- dinamičko osvježavanje vjerojatnosti: NE
- veličina populacije, N = 30
- broj evaluacija, E = 500000 – 1000000
    - broj generacija = broj evaluacija / veličina populacije
- elitizam: DA (2 jednike)
- intervali za parametre pravila
    - c → [-5, 15]
    - w → [1, 5]
    - p → [-10, 10]
    - q → [-10, 10]
    - r → { [-65, 65], [-150, 150] }
- paralelna obrada *fitnessa*: DA

U okviru eksperimentiranja, implementirana je dinamička promjena vjerojatnosti križanja i mutacije. Ideja je bila da u početku vjerojatnost križanja bude 100%, a mutacije relativno velika (≈20%). Cilj je bio da u početku algoritma jednike jako diverzificiraju, odnosno pohlepno pretražuju prostor domene, a na pola promijenjenig generacija da počnu lokalno istraživati prostor. Vjerojatnost križanja i mutacije eksponencijalnom brzinom pada prema vjerojatnosti postavljenoj na početku programa. Slika 6. prikazuje odsječak Java koda.

```
 * Exponential decreasing.
 * @param generation
 */
private void updateμ(int generation) {

    double treshold = 0.05;

    //NOTE: update only if initial mutation less then treshold
    //otherwise, no updating.
    if (μ0 <= treshold) {
        //fall rate => e^(-FALL_RATE*x)
        μ = μ0 + (treshold-μ0)*(Math.exp(-10.0*generation/generations));

    }

}

/**
 * Dynamicly updated crossing.
 * Exponential decreasing.
 * @param generation
 */
private void updateχ(int generation) {

    //fall rate => e^(-FALL_RATE*x)
    χ = χ0 + (1-χ0)*(Math.exp(-10.0*generation/generations));

}
```

Slika 6. Dinamičko osvježavanje vjerojatnosti mutacije i križanja.

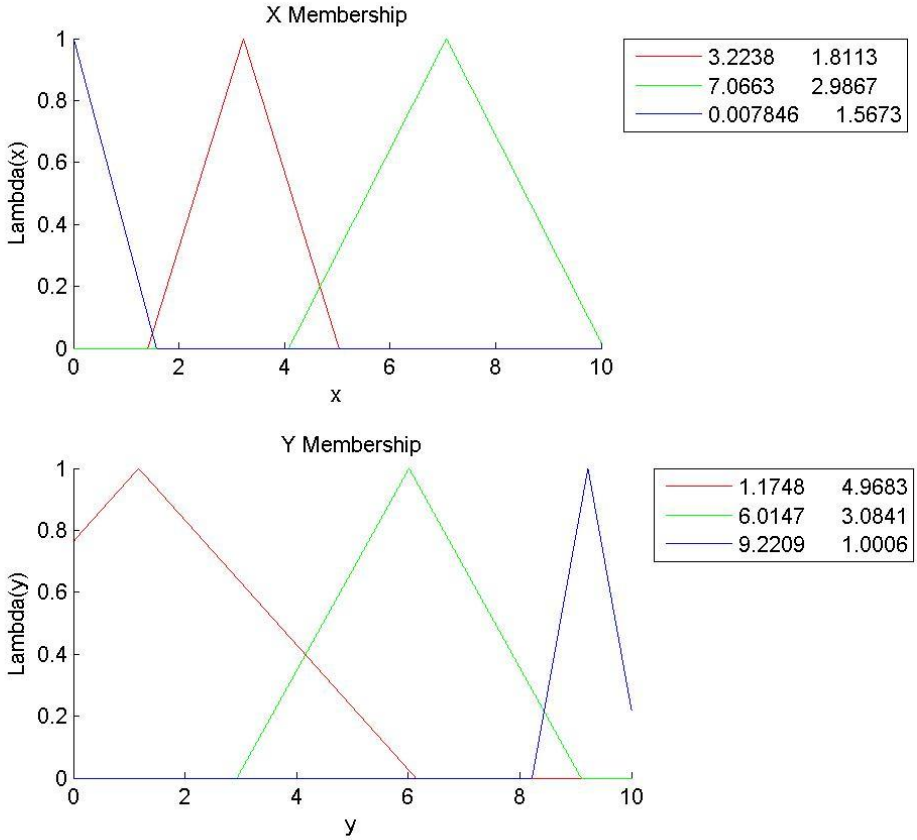
Od ovakvog pristupa se odustalo jer se pokazao manje efikasnim od statičkih vrijednosti. Razlog loših rezultata je što se jakom diverzifikacijom često pronalaze lokalni optimumi (iz kojih se ujedno i brzo iskače), te kada vjerojatnost križanja i mutacije padne na početno zadanu vrijednost, kromosom najčešće ostaje u lokalnom optimumu u koji ga je jaka diverzifikacija dovela (naravno, ovo se ne događa u 100% slučajeva, ali dovoljno često).

Također, pošto je fitness funkcija usko grlo rada sustava, obrada se paralelizirala tako da se fitness funkcija evaluira na onolikom broju jezgara procesora koliko računalo na kojem se izvodi algoritam ima (u mom slučaju, 2 jezgre, ubrzanje ≈ 2x).

2. Zadatak

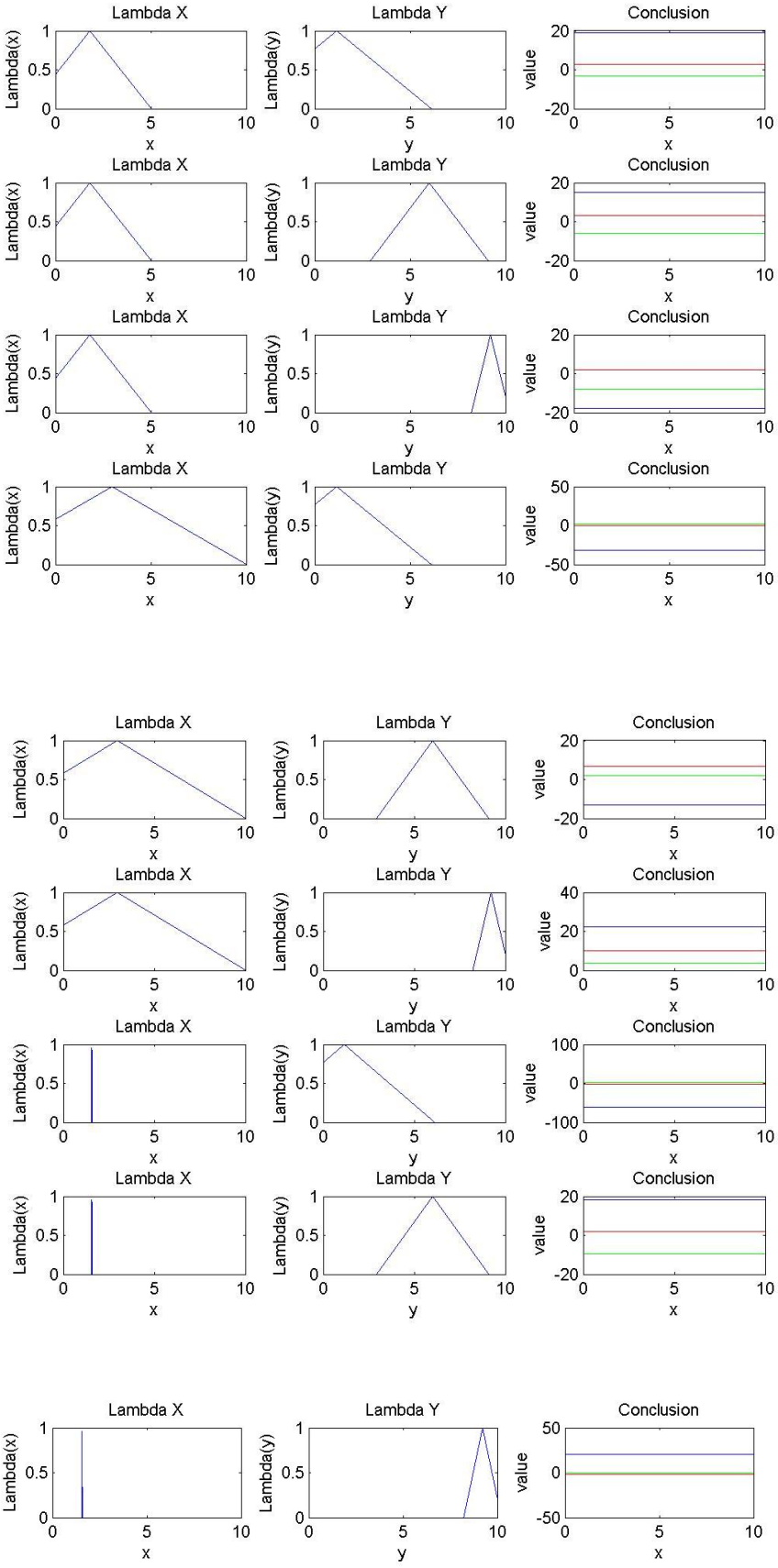
2.1 TSK zaključak: px+qy+r

Slika 7. prikazuje naučene funkcije pripadnosti pravila.



Slika 7. Naučene funkcije pripadnosti za domenu x i y.

Slika 8. prikazuje svih 9 naučenih pravila.



Slika 8. 9 naučenih pravila za TSK oblika **px+qy+r**

Nazivi neizrazitih skupova prikazani su tablicom 1.

Tablica 1. Nazivi neizrazitih skupova.

X	Y
Broj više-manje oko 3	Broj oko 5
Broj oko 7	Broj oko 3
Broj blizu 0	Broj više-manje oko 1

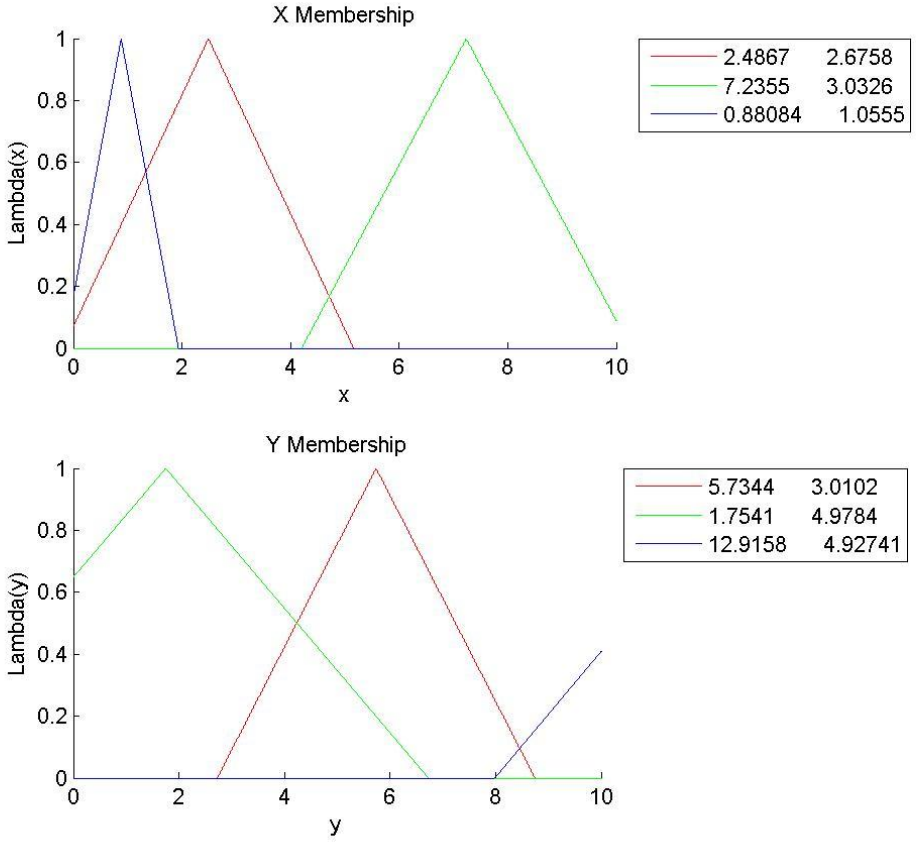
Svih 9 pravila oblika: **Ako x je X i y je Y tada Konsekvens(p,q,r)**, prikazani su tablicom 2.

Tablica 2. Svih 9 pravila opisanih riječima.

X	Y	Konsekvens
Broj više-manje oko 3	Broj oko 5	2.942966875957435 -3.1075692475948724 18.899910464798893
Broj više-manje oko 3	Broj oko 3	3.0791002771195473 -5.822440306891332 15.053272399641273
Broj više-manje oko 3	Broj više-manje oko 1	1.984920942606534 -7.7300896056305906 -17.960192236633496
Broj oko 7	Broj oko 5	0.29432700185862615 1.8038885459998815 -32.162749308339386
Broj oko 7	Broj oko 3	6.546072454501888 2.1672354078282385 -13.016320797116911
Broj oko 7	Broj više-manje oko 1	9.997498080499003 3.82295168937293 22.36083437549577
Broj blizu 0	Broj oko 5	-2.930728338597789 1.5183406776409147 -60.692286868494875
Broj blizu 0	Broj oko 3	1.9094177890863229 -9.675290434341285 18.372472826751654
Broj blizu 0	Broj više-manje oko 1	-1.6029902274603443 0.160917173883279 20.456056652615153

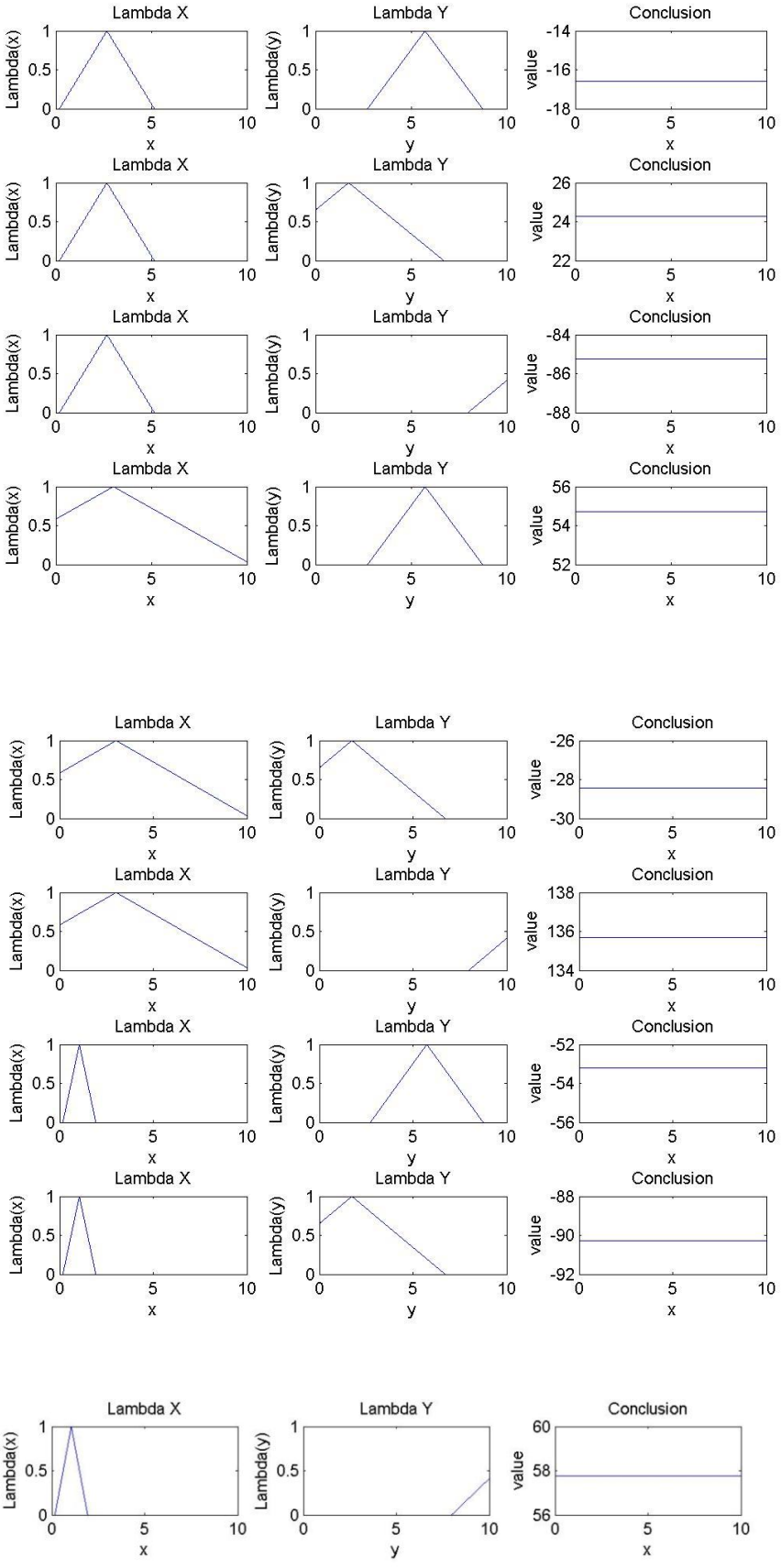
2.2 TSK zaključak: r (const.)

Slika 9. prikazuje naučene funkcije pripadnosti pravila.



Slika 9. Naučene funkcije pripadnosti za domenu x i y.

Slika 10. prikazuje svih 9 naučenih pravila.



Slika 10. 9 naučenih pravila za TSK oblika **r (const.)**

Nazivi neizrazitih skupova prikazani su tablicom 3.

Tablica 3. Nazivi neizrazitih skupova.

X	Y
Broj više-manje oko 2.5	Broj relativno više-manje oko 5.75
Broj relativno više-manje oko 7.25	Broj jako više-manje oko 1.75
Broj oko 1	Broj jako više-manje oko 13

Svih 9 pravila oblika: **Ako x je X i y je Y tada Konsekvens(r)**, prikazani su tablicom 4.

Tablica 4. Svih 9 pravila opisanih riječima.

X	Y	Konsekvens
Broj više-manje oko 2.5	Broj relativno više-manje oko 5.75	-16.586557457083018
Broj više-manje oko 2.5	Broj jako više-manje oko 1.75	24.266443612056506
Broj više-manje oko 2.5	Broj jako više-manje oko 13	-85.25848050219535
Broj relativno više-manje oko 7.25	Broj relativno više-manje oko 5.75	54.70194974377765
Broj relativno više-manje oko 7.25	Broj jako više-manje oko 1.75	-28.432584629370368
Broj relativno više-manje oko 7.25	Broj jako više-manje oko 13	135.67451547485734
Broj oko 1	Broj relativno više-manje oko 5.75	-53.21600533628036
Broj oko 1	Broj jako više-manje oko 1.75	-90.27411795488837
Broj oko 1	Broj jako više-manje oko 13	57.751096196870115

3. Zadatak

Kod najboljeg rješenja pogreška učenja (po primjeru) je:

- Error(TSK→p,q,r) = **1.83**
- Error(TSK→r) = **31.5**

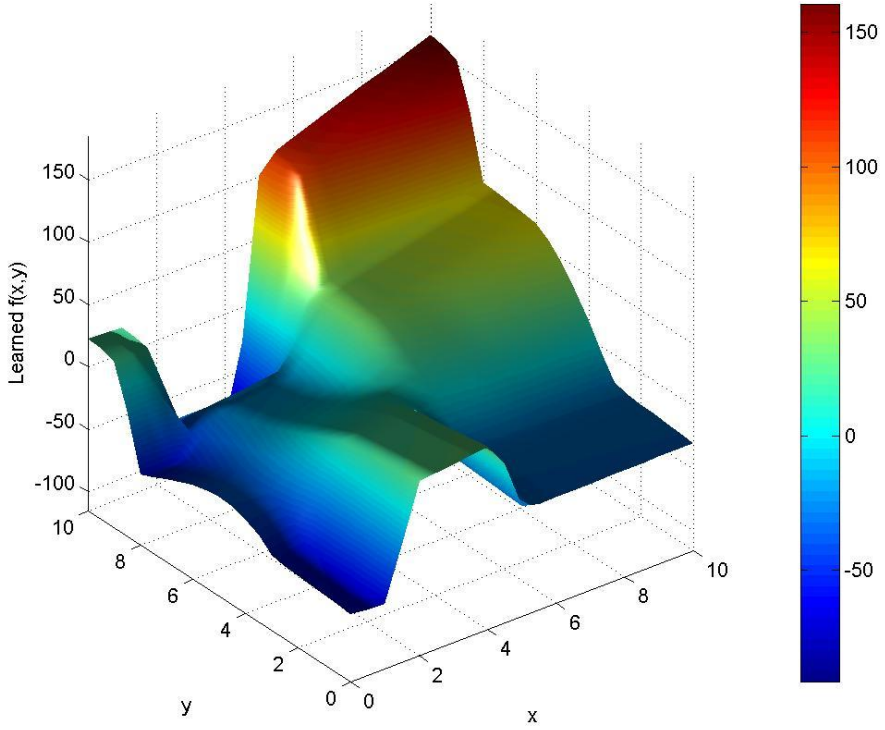
Pogreška učenja kod metode TSK→r (const) veća je nego kod TSK→p,q,r zbog puno manje ekspresivnosti nego kod druge metode (zaključak je konstanta u prvom slučaju, a u drugom je funkcija ulaza u neuronsku mrežu).

Za pronalazak prihvatljivog rješenja bilo je potrebno 1.000.000 evaluacija (33.333 generacija).
OPASKA: jedini prihvatljivi kriterij usporedbe evolucijskih algoritama (genetskog algoritma) jest broj evaluacija (ne generacije, niti vrijeme izvođenja) da bi usporedba bila relevantna[2].

3.1 TSK zaključak: px+qy+r

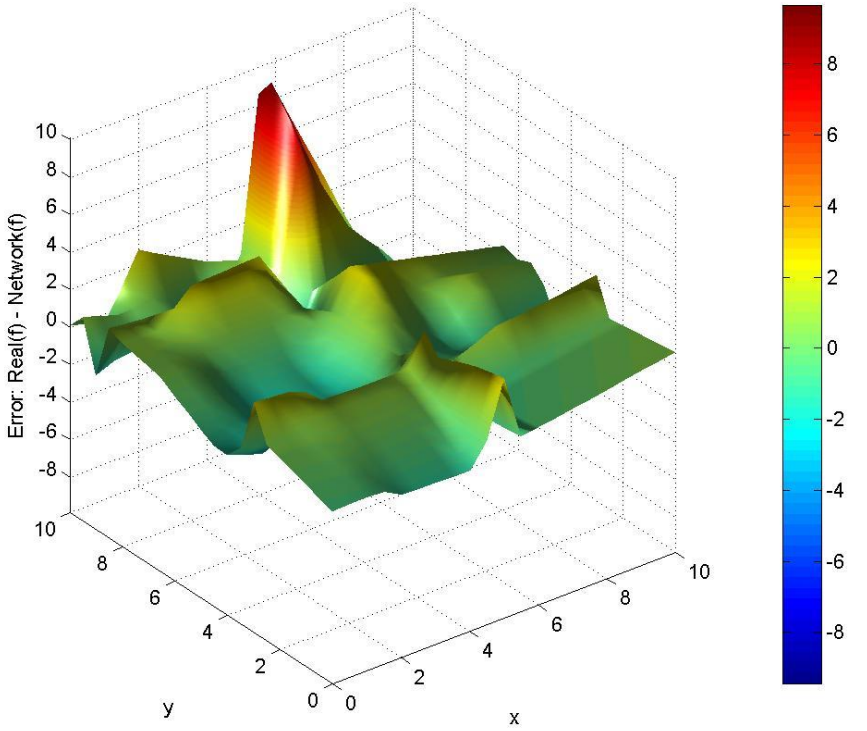
Slika 11. prikazuje naučen sustav (ovisnost izlaza o ulazu).

[2] Mentor: prof.dr.sc. Domagoj Jakobović



Slika 11. Ovisnost izlaza sustava o obje ulazne varijable.

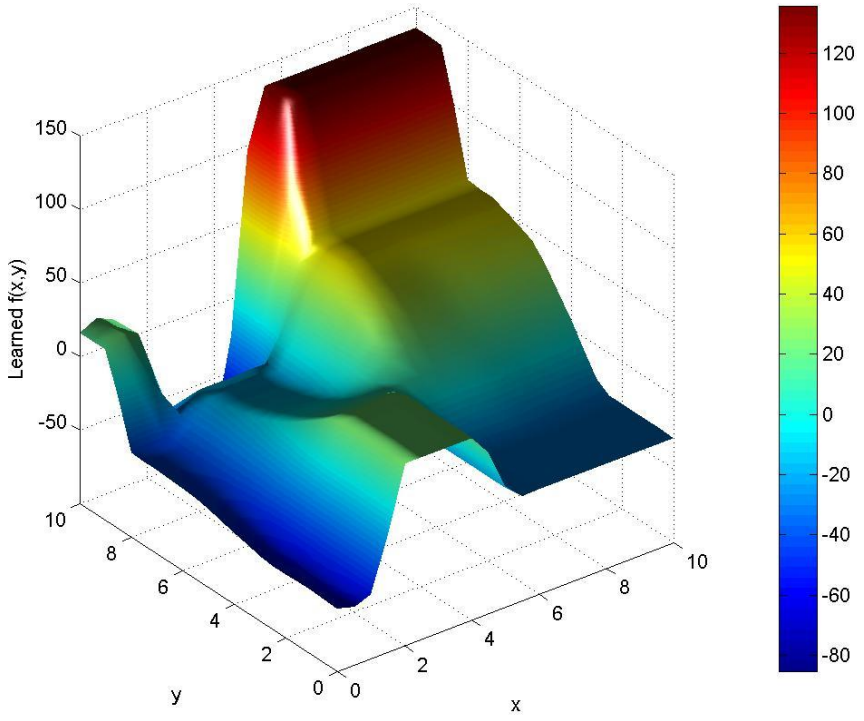
Slika 12. prikazuje funkciju pogreške (definiranu 2. dijelom projekta).



Slika 12. Funkcija pogreške.

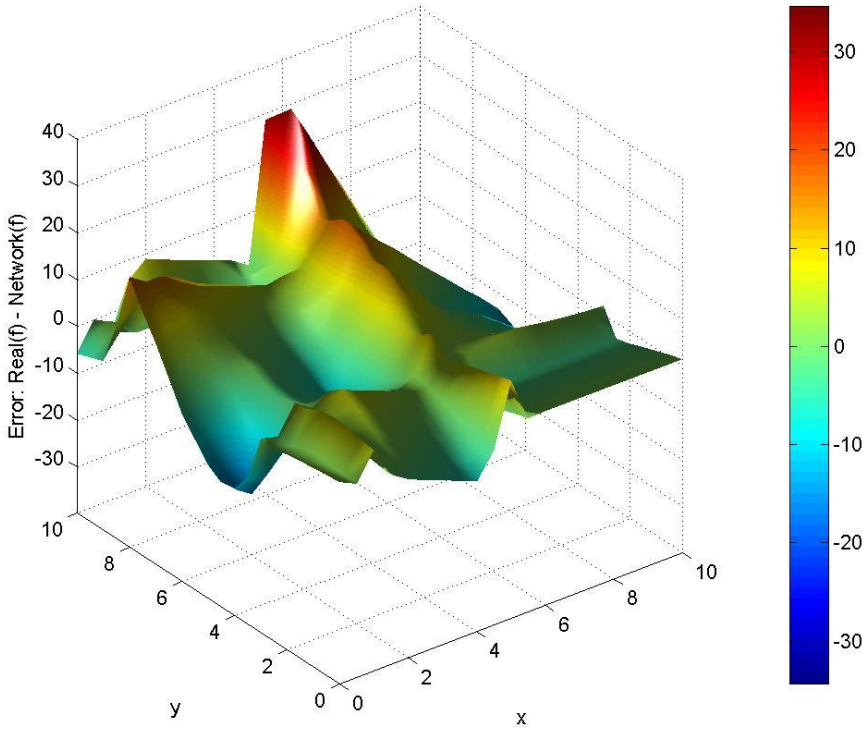
3.2 TSK zaključak: r (const.)

Slika 13. prikazuje naučen sustav (ovisnost izlaza o ulazu).



Slika 13. Ovisnost izlaza sustava o obje ulazne varijable.

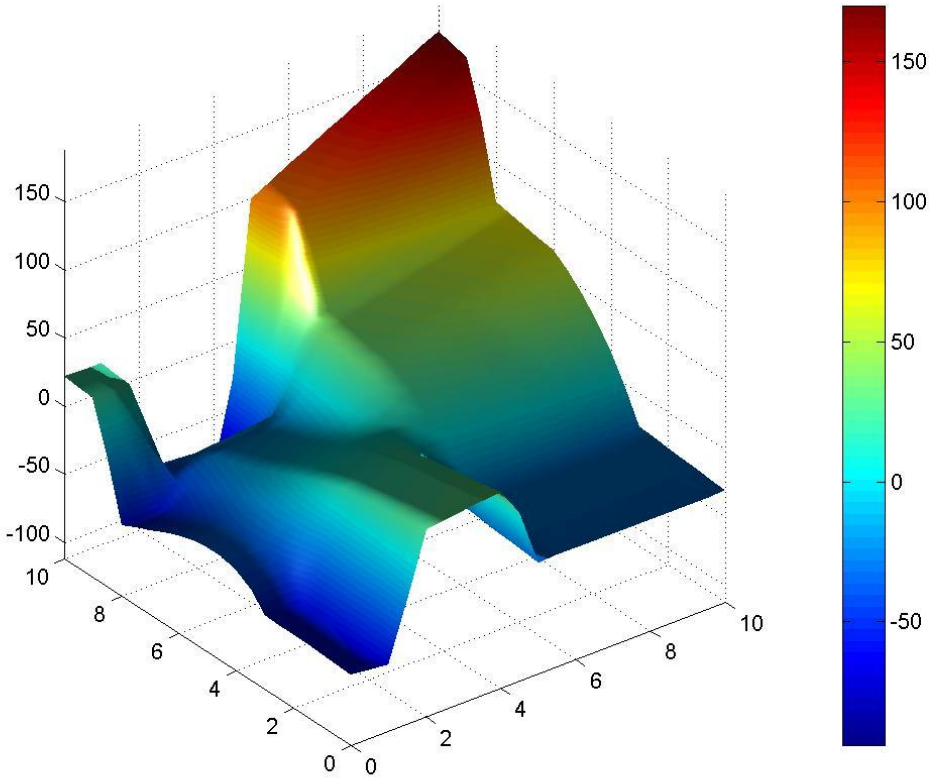
Slika 14. prikazuje funkciju pogreške (definiranu 2. dijelom projekta).



Slika 14. Funkcija pogreške.

3.3 Slika originalne funkcije za učenje

Slika 15. prikazuje originalnu funkciju za učenje (usporedbe radi).



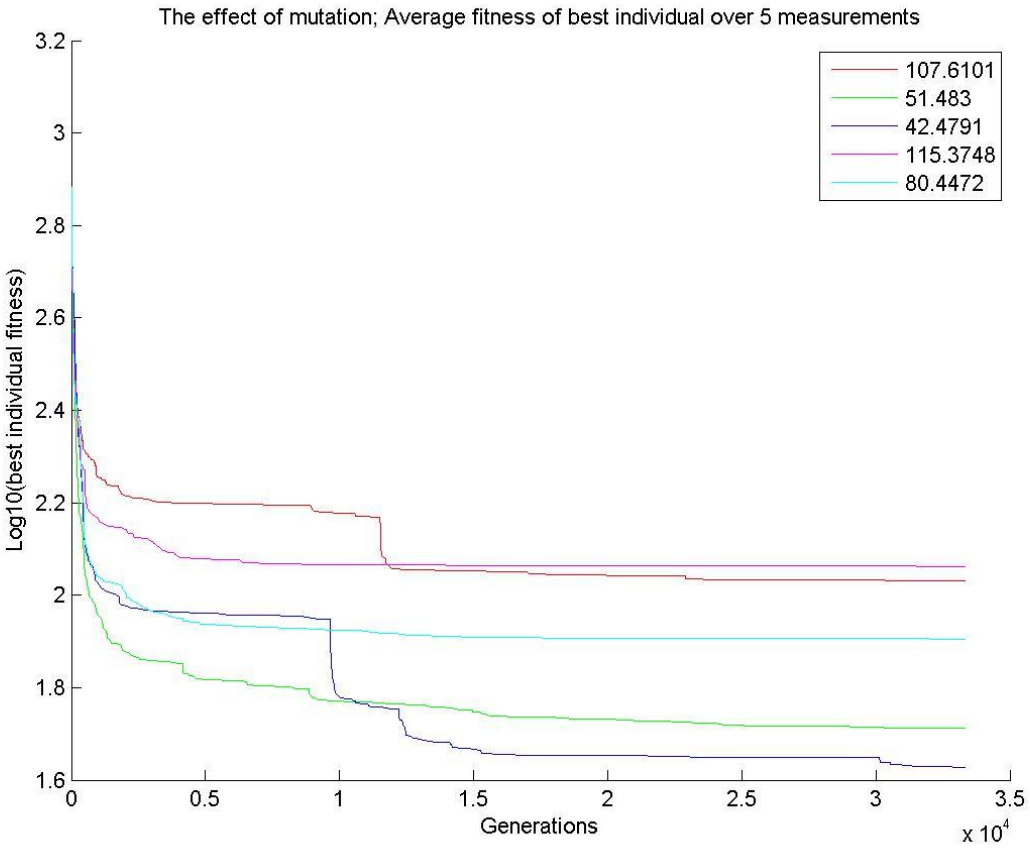
Slika 15. Originalna funkcija za učenje.

4. Zadatak

Slika 16. prikazuje prosječnu dobrotu najbolje jedinke kroz 5 mjerenja i 1.000.000 evaluacija. Mjerenja su izvedena za sljedeće vrijednosti:

- μ/50
- μ/10
- μ = 0.016
- μ*10
- μ*50

Na slici, u legendi, prikazana je prosječna dobrota na kraju postupka, a svaka linija u legendi predstavlja mutaciju upravo onu koja je iznad, za tu liniju napisana.



Slika 16. Prosječna dobrota najbolje jednike kroz generacije, u 5 mjerenja. Svaki graf izrađen je različitom stopom mutacije (navedenom u odlomku iznad slike).

Iz slike se može vidjeti da je odabrana stupa mutacije, μ, upravo i postigla najbolju vrijednost.

Ukoliko je vjerojatnost mutacije premala, konvergencija je izrazito spora (crvena linija); visoka stopa intenzifikacije i često zapinjanje i ostajanje u lokalnim optimumima.

Ukoliko je vjerojatnost mutacije prevelika, u **početku** je konvergencija relativno brza, ali nakon određenog broja evaluacija, konvergencija postaje izrazito spora. Razlog takvom ponašanju je što je ovakav način pretrage sličan „random search-u“ (slučajnoj pretrazi) jer postupak izrazito diverzificira. U početku se relativno brzo pogađaju lokalni optimumi, ali nakon određenog broja evaluacija, postupak staje u najboljem lokalnom optimumu jer „ne može pogoditi“ optimum bolji od trenutnog.

Iz priloženg grafa najbolje vrijednosti postigle su niske vjerojatnosti mutacije, ali ne preniske:

- $\mu = 0.0016$,
- $\mu = 0.016$.

Vjerojatnosti za križanje i mutaciju moraju biti izabrane vrlo pažljivo jer drastično utječu na konačan rezultat.