SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 651

Programsko ostvarenje algoritama rojeva čestica

Denis Ćutić

Zagreb, veljača 2014.

Sadržaj

1. Uvod

Sveprisutnost problema optimizacije razlog je velikom broju razvijenih pristupa za njihovo rješavanje. Iako bi željeli koristiti egzaktne metode koje nam uvijek daju optimalno rješenje njih je često nemoguće primijeniti, najčešće zbog težine problema što postavlja praktično ograničenje na vrijeme potrebno za pronalaženje optimalnog rješenja. Uz njih razvijeni su i pristupi koji ne garantiraju pronalaženje optimalnog rješenja već nastoje u što većoj mjeri zadovoljiti postavljene uvjete poput brzine pronalaženja rješenja, njegove kvalitete, primjenjivosti na široku klasu problema i druge.

Jednu veliku skupinu takvih pristupa predstavljaju prirodom inspirirani optimizacijski algoritmi od kojih će u ovom radu biti razmotren algoritam roja čestica i njegove inačice. Njegov rad se temelji na simulaciji kretanja jata ptica, a odabir ovog algoritma potaknut je njegovom popularnošću, koju zahvaljuje jednostavnosti implementacije i dobrim performansama.

U radu će biti prikazano na koji način radi osnovna inačica, o kojim parametrima i na koji način o njima ovisi, te koje su njene prednosti i mane. Nadalje, opisat ćemo skup naprednijih inačica koje nastoje ispraviti nedostatke osnovne inačice s ciljem poboljšavanja performansi. Razmotrit ćemo kako se osnovna i naprednije inačice ponašaju na različitim problemima i vidjeti koje strategije naprednih inačica pospješuju proces optimizacije općenito ili za određene probleme. Pritom će usporedba inačica biti provedena na skupu standardnih problema razvijenih za testiranje performansi optimizacijskih algoritama te na problemu učenja mreže ANFIS, koja će biti opisana u jednom od poglavlja.

Uzimajući u obzir rezultate iz prethodne analize bit će prikazan pokušaj stvaranja novih inačica koristeći komponente razmotrenih naprednih inačica za koje se pokazalo da značajno utječu na njihove performanse. Pokazat ćemo da u pravilu tako stvoreni algoritmi mogu pod cijenom veće kompleksnosti ostvariti bolje rezultate od algoritama od kojih su sastavljeni (čije komponente koriste).

2. Optimizacijski algoritmi

2.1 Općenito o optimizacijskim algoritmima

Problem optimizacije, odnosno pronalaženja najboljeg rješenja matematičkog modela u skupu svih mogućih rješenja, spada u skupinu problema čije je rješavanje omogućila pojava računala. Razlog tome je što su postupci na kojima se temelji njihovo rješavanje isključivo iterativne prirode. Odnosno, do rješenja se nastoji doći slijedom koraka kojima se nastoji poboljšati pronađeno rješenje. U postupku pronalaženja rješenja algoritam može proći kroz lokalne optimume koji predstavljaju najbolja rješenja u njihovoj okolici. Takva rješenja često predstavljaju prepreku u pronalaženju globalnog optimuma u slučajevima kad algoritam nije u stanju pomaknuti prostor pretraživanja izvan okoline tih rješenja.

Trivijalan pokušaj rješavanja optimizacijskog problema predstavlja metoda iscrpnog pretraživanja prostora mogućih rješenja. U tom slučaju algoritam sistematično „posjećuje“ sva valjana rješenja te odabire ono za koje funkcija, koja opisuje problem, poprima najveću ili najmanju vrijednost, ovisno o tome radi li se o problemu maksimizacije ili minimizacije. Takav pristup omogućuje rješavanje problema čiji je prostor rješenja unutar određenih granica. Naime, neovisno o impresivnoj jačini današnjih računala koja su u stanju izvesti stotine trilijuna operacija u sekundi, većina optimizacijskih problema ima prostor stanja za čije je iscrpno pretraživanje potrebno više vremena od starosti svemira, jer sadrže 10^{100} i više stanja. U tu skupinu problema dakako spadaju svi nama najzanimljiviji problemi poput problema trgovačkog putnika, problema izrade rasporeda te mnogih drugih. Rješavanje takvih problema je u mnogim područjima svakodnevnica, te se zahtijeva da se do rješenja dođe u razumnom, a ponekad i u realnom vremenu.

Kako bi se suočili s tim zahtjevima za ovu klasu problema razvijeni su razni pristupi, a među njima su najpoznatije heuristike odnosno metaheuristike koje funkcioniraju na način da definiraju strategiju pretraživanja prostora stanja. Iako te metode ne garantiraju

pronalazak optimalnog rješenja, omogućavaju pronalazak dovoljno dobrog rješenja u zadanim vremenskim ograničenjima. Iako se oba pristupa temelje na istoj ideji, razlikuju se u tome što se heuristike definiraju za određene probleme dok metaheuristike definiraju globalnu strategiju koju je moguće primijeniti na širok spektar različitih problema. U ovom će radu biti razmatrana metaheuristika roja čestica, no prije nego krenemo u njeno detaljno proučavanje razmotrit ćemo klasifikaciju metaheuristika i vidjeti kako se međusobno odnose.

2.2 Podjela metaheuristika

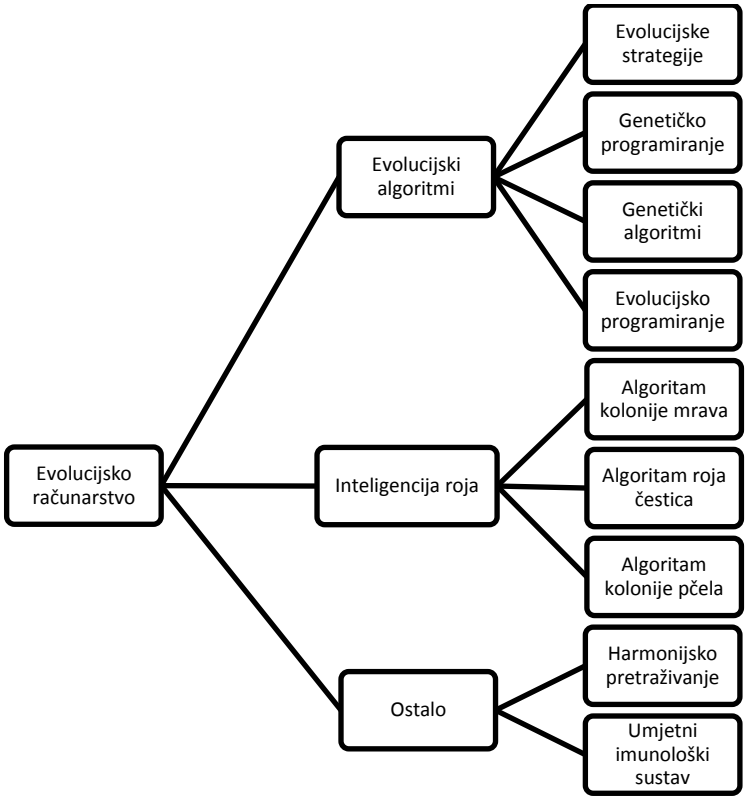
Razvijen je veliki broj metaheuristika te je podjelu moguće provesti po različitim karakteristikama, kao što su: broj rješenja s kojima algoritam radi, kompleksnost, vrsti podataka koju koriste (kontinuirani ili diskretni), ideja na temelju koje je algoritam nastao i raznim drugim. U nastavku je dan kratak pregled nekih podjela, s obzirom na to da iscrpan pregled nadilazi potrebe ovog rada.

Krenimo razmatrajući broj rješenja koje algoritam koristi. Po toj podjeli dijelimo ih na populacijske i algoritme s jednim rješenjem, ovisno održavaju li skup rješenja te do konačnog rješenja dolaze interakcijom jedinki iz populacije rješenja ili koriste jedno rješenje nad kojim provode izmjene. Kako se i iz samog imena može zaključiti, algoritam roja čestica, spada u prvu skupinu te ga možemo još specifičnije svrstati u algoritme koji koriste inteligenciju roja, a karakterizira ih iskorištavanje kolektivnog ponašanje decentraliziranih samostalnih jedinki iz populacije.

Jednu veliku skupinu čine algoritmi evolucijskog računanja. Oni su nastali promatranjem prirode i načina na koji živa bića međusobnom, direktnom ili indirektnom, interakcijom rješavaju probleme na koje nailaze ili doprinose napretku vrste [1]. Na taj način su nastali algoritmi inspirirani Darwinovom teorijom evolucije, ponašanjem kolonije mrava i osa, imunološkim sustavom, inteligencijom roja (koja će biti objašnjena u nastavku) [2] i nizom drugih primjera. Neki od algoritama evolucijskog programiranja su: genetski algoritam (*GA*) (Holland, 1975) (Čupić, 2010), algoritam kolonije mrava (*ACO*) (Dorigo, 1992) (Čupić, 2010), algoritam roja čestica (*PSO*) koji je obrađen u sljedećem poglavlju (Eberhart, 1995) (Clerc, 2006) (Čupić, 2010) [3], umjetni imunološki sustav (*IA*)

(Farmer, 1986) (Čupić, 2010), algoritam roja pčela (*BCO*) (Teodorović, 2005), stohastičko difuzno pretraživanje (*SDS*) (Bishop, 1989), harmonijsko pretraživanje (*HS*) (Geem, 2001) i mnogi drugi. Na slici 1 možemo vidjeti njihovu podjelu.

Još jednu karakterističnu skupinu metaheuristika čine algoritmi lokalnog pretraživanja. Oni do rješenja nastoje doći lokalnim izmjenama te ih karakteriziraju brzina i jednostavnost. Nedostatak je velika vjerojatnost da u procesu zaglave u lokalnim optimumima. Često se ovi algoritmi koriste u kombinaciji s kompleksnijim metaheuristikama kako bi pobliže istražili potencijalno zanimljiva pronađena područja u kasnijim fazama pretraživanja.



Slika 1 - Podjela algoritama evolucijskog računanja

Uz kombiniranje metaheuristika s lokalnim pretraživanjem, sve su popularnije razne kombinacije kompleksnijih metaheuristika u kojima se nastoje iskoristiti prednosti jedne metaheuristike kako bi se smanjili nedostaci druge. Osnovni problem koji se tim pristupom nastoji riješiti je vezan za faze pretraživanja (eksploracije) i iskorištavanja (eksploatacije). Odnosno nastojanja algoritma da pretraži što veći dio prostora rješenja i da detaljno istraži zanimljiva područja. Nekoliko implementacija hibridnih metaheuristika će biti prikazane u nastavku rada.

Sad kad smo razmotrili problem optimizacije i napravili kratki pregled različitih pristupa za njegovo rješavanje krenut ćemo u detaljno razmatranje algoritma roja čestica, njegovu osnovnu inačicu i naprednije pristupe koji nastoje riješiti njene nedostatke.

3. Algoritam roja četica

U ovom ćemo poglavlju predstaviti osnovnu ideju algoritma roja čestica te detaljno razmotriti njegov rad i parametre o kojima ovisi. Velik dio poglavlja će bit posvećen razmatranju parametara s obzirom na njihov utjecaj na performanse algoritma. Vidjet ćemo o kojim sve parametrima ovisi osnovna inačica algoritma, te objasniti na koji način oni utječu na ponašanje populacije čestica u procesu pretrage prostora stanja. Također, razmotrit ćemo prednosti i nedostatke osnovne inačice i vidjeti na koji su se način nedostatci pokušali otkloniti. Posebnu pažnju ćemo posvetiti problemu balansiranja eksploracije i eksploatacije. Navest ćemo neke konkretne inačice algoritma roja čestica koji će biti detaljnije razmotreni u nastavku rada.

3.1 Ideja

Algoritam je nastao nadovezivanjem R. C. Eberharta i J. Kennedya na rad C. W. Reynoldsa, koji je skupom jednostavnih pravila želio simulirati kretanja jata ptica na računalu (Čupić, 2010). Prilikom kretanja populacije dolazi do razmijene informacije među susjednim jedinkama, što dovodi do pojave inteligencije roja. Tako će, na primjer, osa kad pronađe lokaciju bogatu peludom i nektarom informirati ostatak roja o novopronađenoj lokaciji. Pritom će prenesena informacija sadržavati određenu razinu netočnosti te postoji vjerojatnost da će jedinke iz roja, na putu ili na krivo prenesenim koordinatama, pronaći bolju lokaciju i o tome informirati susjedne jedinke koje će dalje propagirati informaciju (Clerc, 2006). Upravo ta interakcija među jedinkama, kroz koju one dobivaju informacije o susjednim povoljnim lokacijama i njihovo kombiniranje sa svojim znanjem o do tad pronađenoj najboljoj lokaciji, vodi jedinke u pronalasku novih, resursima bogatijih, područja.

Uz Reynoldsov rad, inspiraciju su pružala i istraživanja na području socijalne psihologije, naročito dinamička teorija socijalnog utjecaja. Pravila koja određuju kretanje čestica u prostoru rješenja se mogu poistovjetiti s modelima ljudskog socijalnog ponašanja u kojima pojedinci usuglašavaju svoja vjerovanja i stavove, s onima svojih vršnjaka [5].

Za popularnost ovog algoritma je najviše zaslužna njegova jednostavnost implementacije i relativno malen broj parametara, a koju potvrđuje i rastući broj akademskih članaka o njemu. Unatoč jednostavnosti algoritam je pokazao dobre rezultate na širokom skupu problema. Za to je zaslužna i činjenica da algoritam ne postavlja nikakva ograničenja na karakteristike optimizacijskog problema te se može koristiti za probleme koje karakterizira nediferencijabilnost, određena razina nepravilnosti, prisutnost šuma ili vremenska promjenjivost. Također, iako je osnovna inačica zamišljena za rješavanje problema s kontinuiranim varijablama nerijetko se algoritam koristi i za kombinatoričke probleme. Stoga ne čudi činjenica da su analizom velikog broja članaka pronađene primjene algoritma na 26 kategorija problema (Poli, 2008). Neke od najpopularnijih primjena vezane su za:

- dizajn sklopovlja, uređaja i antena,
- kontrolne aplikacije za tokove, uređaje i sustave,
- distribucijske mreže,
- elektroniku,
- upravljanjem energetskim sustavima i postrojenjima,
- izradu rasporeda te
- analizu slike i videa, koja je ujedno i najpopularnija primjena.

Slijedi detaljna analiza osnovne inačice kako bi vidjeli na koji način algoritam roja čestica uspijeva ostvariti dobre rezultate na tako širokom skupu optimizacijskih problema. Krenut ćemo s kratkim formalnim opisom algoritma te opisati korak inicijalizacije i glavni korak algoritma koji se ponavlja do zadovoljenja postavljenih uvjeta.

3.2 Opis algoritma

Osnovna inačica algoritma roja čestica je namijenjena za globalnu optimizaciju u kojoj se rješenje može predstaviti kao točka u *n*-dimenzijskom prostoru [3]. Jedinke se modeliraju česticama koje se opisuju položajem, brzinom, susjedstvom te sadrže informaciju o svojoj najboljoj do tad pronađenoj lokaciji. Položaj čestice je jedno potencijalno rješenje i predstavlja točku promatranog *n*-dimenzijskog prostora, dok je

brzina vektor u tom istom prostoru. Skup čestica čini populaciju te one svojim kretanjem, koje ovisi o svom i iskustvu svojih susjeda, pretražuju prostor stanja.

Zbog mogućnosti da funkcija koju se optimizira nije definirana za sve vrijednosti potrebno je ograničiti prostor kretanja čestica. Desi li se da čestica napusti prostor stanja ona treba biti vraćena u njega. To se najčešće obavlja tako da se za položaj postavi najbliža joj točka iz prostora stanja. Dodatno, da bi se izbjeglo moguće višestruko izlaženje i ponovo vraćanje u prostor stanja, potrebno je osigurati promjenu vektora smjera (Clerc, 2006).

Važnu stavku u radu algoritma imaju parametri koji su u osnovnoj inačici vremenski nepromjenjivi. Njima se određuje omjer između eksploracije i eksploatacije koju će algoritam provoditi u pretraživanju prostora stanja, kao i način na koji će se informacija o pronađenim dobrim rješenjima prenositi među česticama. Međutim, prije nego možemo razmotriti točan utjecaj parametara trebamo formalno opisati algoritam i njegove korake.

3.2.1 Inicijalizacija i korak algoritma

Na početku rada algoritma inicijaliziraju se čestice, pri čemu im se dodjeljuju slučajne vrijednosti za položaj i brzinu te se definiraju susjedstva. Pritom, svaka čestica može pripadati različitim susjedstvima čime se omogućava tok informacija među susjedstvima. Samo izvođenje algoritma provodi se u diskretnim koracima. Pritom se u svakom koraku za sve čestice računa:

- *novi vektor brzine* zbrajanjem:
    - vektora trenutne brzine,
    - vektora određenog razlikom svojeg najboljeg i trenutnog položaja te
    - vektora određenog razlikom lokalno najboljeg (u susjedstvu najboljeg pronađenog položaja) i trenutnog položaja;

  pomnoženih faktorima povjerenja kako je prikazano izrazom (1),
- *novi položaj* dodavanjem novog vektora brzine trenutnom položaju kao u izrazu (2) te

- *vrijednost funkcije* za novi položaj; ako je novi položaj bolji od prethodnog najboljeg položaja za tu česticu, vrijednost potonjeg se zamjenjuje novim položajem.

$$v_i^{t+1} = c_1 * v_i^t + c_2 * r_p * (p_i - x_i^t) + c_3 * r_l * (l_i - x_i^t) \quad (1)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2)$$

Pritom su:

$\vec{v}_i^t$ – *i*-ta komponenta trenutnog vektora brzine,

$\vec{v}_i^{t+1}$ - *i*-ta komponenta novog vektora brzine,

$\vec{x}_i^t$ – *i*-ta komponenta radij-vektora trenutnog položaja čestice,

$\vec{x}_i^{t+1}$ – *i*-ta komponenta radij-vektora novog položaja čestice,

r_p – slučajna vrijednost iz uniformne distribucije [0, 1],

r_l – slučajna vrijednost iz uniformne distribucije [0, 1],

$\vec{p}_i$ – radij-vektor osobnog najbolje položaja,

$\vec{l}_i$ – radij-vektor osobnog najboljeg položaja te

c_1, c_2, c_3 – faktori povjerenja.

Postupak završava kad je izvršen definirani maksimalni broj iteracija ili kad je pronađeno zadovoljavajuće rješenje, odnosno vrijednost funkcije je manja ili veća od neke definirane vrijednosti. U nekim slučajevima se kao uvjet zaustavljanja koristi i broj poziva funkcije dobrote umjesto broja iteracija. Rad algoritma opisan je u pseudokodu 1. Na temelju pseudokoda uočavamo jednostavnost implementacije o kojoj je bilo riječ u uvodnom poglavlju.

Pseudokod 1 - Osnovna inačica algoritma roja čestica

1. Inicijaliziraj čestice
2. Stvori susjedstva
3. **Ponavljaj**
4. **Ponavljaj**
5. Izračunaj dobrotu čestice
6. Izračunaj novu brzinu čestice
7. Izračunaj novi položaj čestice
8. **Za svaku česticu**
9. **Dok nije zadovoljen uvjet zaustavljanja**

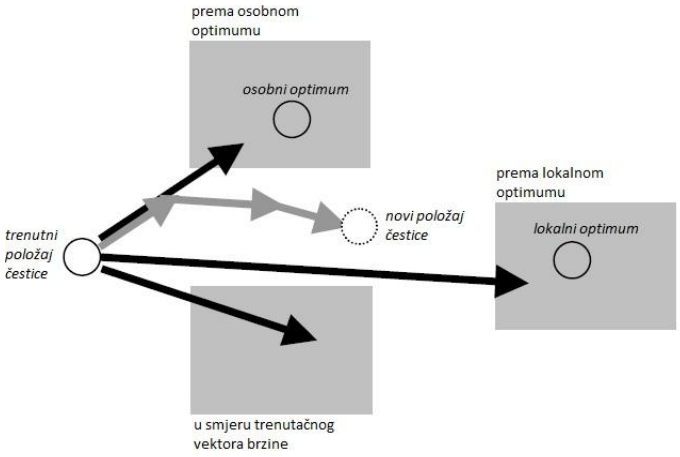
3.3 Parametri

U prethodnom tekstu smo mogli vidjeti nekoliko eksplicitno korištenih parametara poput faktora povjerenja i broja čestica te jednog implicitnog, topologiju susjedstva. Sada ćemo analizirati način na koji oni utječu na rad i performanse algoritma, dok ćemo na kraju razmotriti problematiku odabira njihovih vrijednosti.

Prvo ćemo razmotriti faktore povjerenja, koji određuju mjeru u kojoj se čestica kreće prema poznatim optimumima (parametri c_2 i c_3), odnosno u kojoj mjeri se nastavlja kretati u svom dotadašnjem smjeru (parametar c_1) (Clerc, 2006). Prikaz navedenog možemo vidjeti na slici 2, a njihov pojedinačni utjecaj možemo vidjeti razmotrimo li granične slučajeve.

- Ako je $c_1 > 0$, $c_2 = c_3 = 0$, čestica se nastavlja kretati u smjeru vektora brzine prethodnog koraka te ju stoga još nazivamo i individualnom komponentom, iako se češće naziva inercijom i označava s ω.
- Ako je $c_2 > 0$, $c_1 = c_3 = 0$, čestica se giba u smjeru svojeg najboljeg rješenja i zbog tog nastojanja da se čestica vrati u svoje najbolje pronađeno rješenje nazivamo ju kognitivnom komponentom.
- Ako je $c_3 > 0$, $c_1 = c_2 = 0$, čestica se giba prema lokalnom najboljem rješenju te zbog uzimanja u obzir rješenja pronađenog od drugih čestica nazivamo još i socijalnom komponentnom.

Njihove vrijednosti moraju biti pažljivo odabrane jer će upravo o njima ovisiti brzina pronalaska i kvaliteta konačnog rješenja. Odabirom malih vrijednosti, naročito za C_1, pomaci u prostoru rješenja biti će mali te će se povećati vjerojatnost da algoritam zaglavi u lokalnom optimumu. Razlog tome je što će se čestice kretati pretežito u smjeru najbolje čestice bez tendencije pretraživanja novih područja. S druge strane velike vrijednosti parametara će imati za posljedicu velike pomake u prostoru rješenja te je velika vjerojatnost da će čestice preletjet preko potencijalno dobrih rješenja.



Slika 2 - Prikaz određivanja novog položaja čestice u koraku algoritma

Također, važan je omjer između njihovih vrijednosti. Veća vrijednost individualne komponente će potencirati eksploraciju, dok će veće vrijednosti kognitivne i socijalne komponente potencirati eksploativno ponašanje. Razlog tome je što zanemarivanjem kognitivne i socijalne komponente čestica će samo u manjoj mjeri odstupati od svoje (početne) putanje, dok će zanemarivanje individualne komponente biti snažno privučena lokalnim optimumima, a time se smanjuje vjerojatnost nailaženja na druga kvalitetna područja u prostoru rješenja. Često inačice koriste vremenski promjenjivu individualnu komponentu, koja se linearno smanjuje s brojem iteracija. Na taj način se nastoji potencirati eksplorativno ponašanje na početku, a eksploativno ponašanje na kraju pretrage.

Sljedeći parametar čiji je utjecaj na rad algoritma potrebno razmotriti je broj čestica. Najjednostavnija inačica algoritma koristi fiksno definirani broj koji ostaje nepromijenjen

do kraja izvođenja. Složenije inačice povećavaju ili smanjuju broj čestica ili ga pak održavaju konstantnim, ali uz korištenje selekcije, pri čemu na mjesto izbačenih ubacuju nove, slučajno generirane, čestice. Postavlja se pitanje koliko je čestica potrebno za dobar rad algoritma? Iako rojevi u životinjskom svijetu mogu biti veliki i sadržavati više desetaka tisuća jedinki samo mali broj jedinki doprinosi roju s novim, kvalitetnim informacijama, dok ostatak roja uglavnom samo iskorištava već poznate informacije. Stoga veliki broj čestica koje stvaraju nepotrebne ili redundantne informacije i uvelike usporavaju rad algoritma nisu od velike koristi. U pravilu se koriste rojevi početne veličine od 20 do 40 čestica, za koje se eksperimentalno pokazalo da daju najbolje rezultate (Clerc, 2006). Još valja napomenuti da za razliku od primjera iz prirode gdje jedinke u pravilu kreću s istog mjesta, algoritam, u nedostatku informacija, nasumično generira početnu populaciju.

Preostalo nam jer razmotriti zadnji parametar, a to je susjedstvo. Njime je definiran način prijenosa informacija, a opisujemo topologijom povezanosti čestica. U nastavku su navedene neke najčešće korištene topologije.

- *Potpuno povezana* – u kojoj je svaka čestica povezana sa svim ostalim česticama, čime je informacija o najboljem rješenju globalna i u svakom trenutku poznata svim česticama. Grafički prikaz se može vidjeti na slici 3 a).
- *Prstenasta* – svaka čestica je povezana s dvije čestice tvoreći zatvoreni krug kao što je grafički prikazano na slici 3 b).
- *von Neumannova* – svaka čestica ima četiri susjeda, kako je prikazano na slici 3 c), tvoreći matricu u kojoj je svaka čestica povezana s neposrednim susjedima, pritom su rubne čestice povezane s česticama na suprotnom kraju.

Najbolji zapamćeni položaj svih čestica predstavlja lokalno najbolje rješenje u tom susjedstvu, a u slučaju potpuno povezane topologije ono je ujedno i globalno najbolje rješenje. Lokalni optimum se kroz iteracije propagira i u slučaju da nisu pronađena nova bolja rješenja ono će nakon nekog vremena postati globalno. Propagacija je moguća jer svaka čestica pripada u više različitih susjedstva, čime su susjedstva povezana. Što su susjedstva manja i brojnija propagacija je sporija, a što je socijalna komponenta veća to je propagacija brža. Stoga bi se na prvi pogled činilo da bi informacija o najboljem rješenju na globalnoj razini, korištenjem potpuno povezane topologije, bila najprikladnija, pošto u

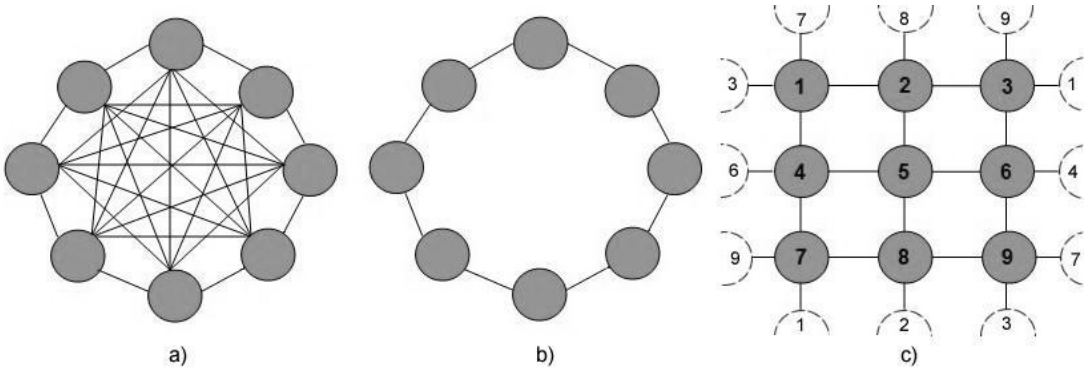
svakom koraku sve čestice imaju „svježu“ informaciju. Međutim, takva povezanost ima za posljedicu smanjenje raznolikosti među česticama, zbog čega algoritam može zaglaviti u lokalnom optimumu. Iz tog razloga je potrebno odabrati topologiju koja će osigurati potrebnu raznolikosti rješenja i prikladnu brzinu propagacije informacije, što u velikoj mjeri ovisi o karakteristikama problema. Nadalje postoje varijante algoritma u kojima su susjedstva dinamička te se u njima može mijenjati broj susjednih čestica, čime se mijenja topologija susjedstva i veze među česticama unutar iste topologije (na primjer kada želimo da susjedstvo čine čestice s najmanjom udaljenošću) [4].

3.3.1 Problem odabira parametara

Osnovni problem odabira parametara je nepostojanje analitičkog pristupa za pronalaženje optimalnih vrijednosti. U suštini, suočeni smo s još jednim optimizacijskim problemom. I u ovom slučaju *n*-torka vrijednosti parametara čini točku u prostoru rješenja, s time da je dimenzija ovog prostora u pravilu znatno manja od dimenzije prostora rješenja problema. Također, pretpostavljamo određenu razinu neosjetljivosti algoritma na vrijednosti parametara, odnosno da se male razlike u vrijednostima parametara neće imati značajan utjecaj na rad algoritma.

Pretraživanje prostora vrijednosti parametara se u pravilu obavlja ručno, procesom pokušaja i pogreške, pri čemu se koriste teoretska razmatranja kako bi se prostor pretraživanja smanjio. U jednostavnim je inačicama, onima s malim brojem parametara, često moguće i analitički doći do raspona ili odnosa vrijednosti parametara za koje se može garantirati konvergencija algoritma.

Nadalje, sama optimalnost skupa parametara ovisi i o optimizacijskom problemu. Skup parametara koji je optimalan za jedan optimizacijski problem, nije optimalan, a moguće ni prikladan, za drugi problem ili isti problem sa značajno različitim skupom podataka. Ovo nas navodi na pitanje: je li moguće razviti prilagodljiv algoritam za koji neće biti potrebno svaki put tražiti zadovoljavajuće vrijednosti parametara? Odgovor je potvrdan i tu skupinu čine adaptivne inačice koje koriste razne strategije za ugađanje vrijednosti parametara za vrijeme rada algoritma (Chen, 2006). Adaptivne inačice uvode dodatnu kompleksnost te se uglavnom koriste za probleme u realnom vremenu gdje je potrebna brza prilagodba na promjenjiv skup ulaznih podataka.



Slika 3 - Topologije: a) potpuno povezana, b) prstenasta i c) von Neumannova

3.4 Inačice

S obzirom na uspjeh algoritma roja čestica, vremenom su se razvile brojne inačice. Razlikuju se po:

- vrsti problema koje rješavaju, ovisno jesu li kontinuirani ili kombinatorički,
- trebaju li imati predefinirane parametre (parametarski) ili ne (adaptivni) (Chen, 2006) (Zhan, 2009),
- vrsti susjedstva, odnosno je su li statička ili dinamička, korištena topologija i način raspoređivanja čestica u njoj (važno kod hijerarhijske inačice [4]) (Janson, 2004),
- načinu širenja informacije,
- načinu izračunavanja brzine (na primjer, drže li vrijednost inercije konstantnom ili se mijenja) (Montes, 2009) (Clerc, 2006),
- rade li samostalno ili se kombiniraju se nekom drugom metaheuristikom (Banks, 2008) (Lin, 2002) (Lin, 2008) (Lin, 2009),
- broju korištenih rojeva (Multi-swarm PSO (Blackwell, 2004)),
- odnosom među rojevima (jesu li svi rojevi jednaki ili postoje povlašteni rojevi) (Niu, 2007) (Niu, 2008) (Li, 2012),
- koriste li neki model populacijske dinamike (na primjer, Lotka-Volterrin model (Kang, 2008)) te
- optimiziraju li statičku ili dinamičku funkciju koja se mijenja tijekom postupka pretrage (Du, 2008).

U pravilu sve inačice nastoje optimizirati faze eksploracije i eksploatacije te definirati strategiju koja će ih pravovremeno provoditi. Time se nastoji izbjeći prerana

konvergencija i zaglavljivanje u lokalnom optimumu, kao i potpuno nasumična pretraga prostora stanja.

Jedna od osnovnih strategija je prvo provesti fazu eksploracije i pretražiti što veći dio prostora rješenja, a tek onda krenuti u fazu eksploatacije „zanimljivih“ područja. U tako definiranoj strategiji je važno odrediti u kojem će trenutku prestati jedna faza, a započeti druga. Osnovna strategija ovog tipa postepeno prelazi iz jedne faze u drugu mijenjanjem vrijednosti parametara po određenom izrazu, najčešće linearnoj funkciji. Međutim, ispravna strategija ne postoji s obzirom na to da trebaju biti prilagođene problemu.

Osim osnovne razvijen je veliki broj složenijih strategija. Neke od karakteristika tih strategija su:

- naizmjenično provode spomenutih faza,
- uvode odbojne sile među bliskim česticama/podrojevima kako bi se spriječila prerana konvergencija,
- koriste proces prirodne selekcije ili neki drugi prirodom inspirirani proces (kompetitivnog ili kooperativnog) među česticama/podrojevima ili
- koriste algoritam roja čestica samo u jednoj od faza, dok za drugu koriste neku drugu metaheuristiku.

Naravno, performanse svih inačica ovise o konkretnom problemu kojeg se nastoji riješiti te kao što je dokazano „No Free Lunch“ teoremom (Wolpert, 1997), ne postoji algoritam koji će biti bolji od svih drugih na svim problemima. Štoviše, za svaku inačicu je potrebno pronaći optimalni skup vrijednosti parametara kako bi na njemu ostvarili željene performanse. Taj problem je sve izraženiji što inačica ima više parametara koji upravljaju njenim radom. U pravilu se broj parametara povećava s kompleksnošću algoritma. Tako će hibridne inačice sadržavati skup parametara za svaki korišteni optimizacijski algoritam što uvelike otežava pronalaženje dobrog skupa parametara.

4. Implementirane inačice

U ovom poglavlju ćemo razmotriti neke složenije inačice algoritma roja čestica. Vidjet ćemo ideje na kojima se temelje te detaljan opis njihovog rada. Ukratko ćemo razmotriti na koji način komponente algoritma utječu na njihov rad te iskoristiti najznačajnije u izgradnji novog algoritma roja čestica. Pristup se temelji na ideji po kojoj je nastala Frankenstein's PSO inačica, koja kombinira komponente jednostavnih inačica osnovnog algoritma, a za koje se pokazalo da značajno doprinose brzini i pouzdanosti pronalaženja rješenja. Pokazalo se da tako izgrađena inačica daje bolje rezultate od inačica koje ih koriste pojedinačno (Montes, 2009).

Na CD-u priloženom uz ovaj rad moguće je pronaći izvorni kod svih algoritama navedenih u ovom poglavlju te upute za njihovo pokretanje.

4.1 GAPSO[1]

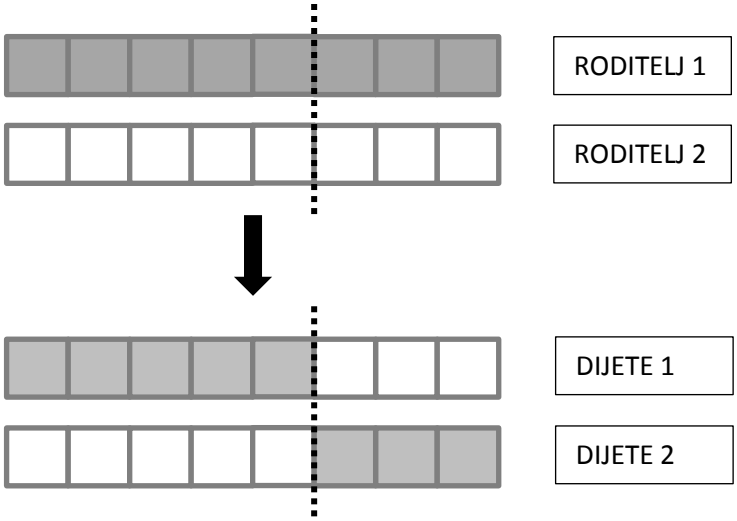
GAPSO (Genetic Algorithm Particle Swarm Optimization) inačica (Ghomsheh, 2007), kao što i samo ime kaže, uvodi elemente genetskog algoritma u rad algoritma roja čestica. Sam genetski algoritam jedna je od najpoznatijih metaheuristika koja se temelji na procesu prirodne selekcije. Rješenja genetskog algoritma predstavljaju kromosomi koji operatorima križanja i mutacije stvaraju kromosome sljedeće generacije, pritom se operatori primjenjuju samo na odabranim kromosomima (selekcija).

U ovoj se inačici algoritma roja čestica nakon svakog koraka iz roja izbacuje najgora čestica te se ona nadomješćuje novom česticom, koja se stvara križanjem dvaju čestica iz roja. Čestice roditelji se biraju nasumično te križanjem nastaju dvije nove čestice, djeca. Odabire se bolje dijete te se ono dodaje u roj kao zamjena za izbačenu česticu, tako da broj čestica ostaje isti.

U izvornom je članku ovaj algoritam korišten za učenje mreže ANFIS[2], problem koji se svodi na pronalaženje optimalnog skupa vrijednost parametara mreže. Parametri su

[1] U izvornom članu autori ne koriste poseban naziv/skraćenicu za ovu inačicu algoritma već ju nazivaju „izmijenjeni PSO“. Radi lakšeg razlikovanja s ostalim inačicama korišten je naziv GAPSO koji upućuje na prisustvo elemenata genetskog algoritma.

kodirani u čestice na način da jedan parametar predstavlja jednu dimenziju čestice. S obzirom na to da mreža ANFIS razlikuje nekoliko grupa parametara ova inačica algoritma u jednoj iteraciji osvježava samo jednu skupinu parametara, odnosno primjenjuje izraze (1) i (2) samo za određene dimenzije čestice. Ova strategija računanja novih pozicija čestica se pokazala veoma značajnom u procesu učenja mreže ANFIS tako što je algoritam pronalazio značajno bolje rezultate od slučaja kada su se nove pozicije čestica računale na temelju svih dimenzija.



Slika 4 - Križanje

Uz standardne parametre algoritma roja čestica za ovu inačicu je potrebno definirati i broj točaka križanja. Taj parametar određuje broj dijelova kromosoma/čestica koji će se naizmjenično prepisati iz roditelja da bi se stvorila djeca (slika 4). U slučaju jedne točke križanja, dio čestice od početka do nasumično odabrane točke križanja se uzima od jednog roditelja, a drugi dio čestice od drugog roditelja. Na isti način se provodi križanje s većim brojem točaka samo što se razmjenjuje veći broj segmenata. Vrijednost ovog parametra ovisi o problemu kojeg se nastoji riješiti jer problemi s malim brojem dimenzija dopuštaju i malen broj točaka križanja. Također, problemi sa strukturiranim rješenjima poput učenje mreže ANFIS, mogu se postići bolje performanse ukoliko se točke križanja podudaraju s grupama parametara.

[2] Mreže ANFIS će biti obrađene u sljedećem poglavlju.

Pseudokod 2 - GAPSO

1. Inicijaliziraj čestice
2. Stvori susjedstva
3. Izračunaj dobrote svih čestica
4. **Ponavljaj**
5. **Ponavljaj**
6. Izračunaj novu brzinu čestice za jednu grupu parametara
7. Izračunaj novi položaj čestice za jednu grupu parametara
8. Izračunaj dobrotu čestice
9. **Za svaku česticu**
10. Izbaci najgoru česticu iz generacije
11. Nasumično odaberi dvije čestice iz roja
12. Stvori dvije nove čestice križanjem odabranih čestica
13. Ubaci u roj bolju od dvaju tako dobivenih čestica
14. **Dok nije zadovoljen uvjet zaustavljanja**

4.2 SAPSO

Kao i prethodna inačica, SAPSO (Symbiotic Adaptive Particle Swarm Optimization) inačica je razvijena za rješavanje problema učenja mreže ANFIS (Lin, 2011). Razvijena je imajući na umu pad performansi optimizacijskih algoritama s porastom dimenzionalnosti prostora rješenja, uslijed kojeg se smanjuje vjerojatnost pronalaženja optimalne regije, koja u pravilu čini mali dio ukupnog prostora. Stoga se kao jedno moguće rješenje postavlja particioniranje prostora pretraživanja u potprostore nižih dimenzionalnosti, ali pod uvjetom da će algoritam i dalje biti u mogućnosti pretražiti sve regije originalnog prostora rješenja. U tu svrhu ova inačica koristi čestice koje predstavljaju dio rješenja[3] i to na način da za svaki dio rješenja postoji podroj čestica koje provode optimizaciju tog dijela.

Dok su u drugim inačicama čestice predstavljale cjelovita rješenja kojima je bilo moguće izračunati dobrotu u skladu s njihovim učinkom, u ovom slučaju to neće biti moguće. Za određivanje dobrote čestice biti će potrebno razmotriti njen učinak u kombinaciji s česticama iz drugih podrojeva. U idealnom slučaju željeli bi izračunati učinke svih mogućih kombinacija podčestica[4], međutim to bi značajno usporilo rad algoritma, a u

[3] Svaka čestica predstavlja jedno pravilo mreže ANFIS.
[4] Podčesticama nazivamo čestice podroja.

nekim slučajevima bi moglo biti i praktično neostvarivo. Stoga se za računanje dobrote podčestica nasumično kombiniraju čestice iz različitih podrojeva sve dok sve podčestice nisu odabrane dovoljan broj puta, podatak koji se zadaje kao parametar algoritma. Nakon toga se za svaku podčesticu zbroje sve dobrote čestica u kojima je podčestica sudjelovala te se zbroj podijeli s brojem čestica u kojima je podčestica sudjelovala.

Druga izmjena koju su autori uveli potaknuta je analizom stabilnosti adaptivne inačice algoritma, odnosno istraživanjem njenog pri pretraživanju prostora rješenja i strategije koju koristi za ugađanje svojih parametara. Malom izmjenom izraza za računanje novog vektora brzine čestice omogućeno je analitičko određivanje vrijednosti parametara koji garantiraju željenu dinamiku ponašanja čestica. Izmijenjen izraz je prikazan pod (3).

$$\begin{aligned} v_i^{t+1} = \omega * v_i^t + 3 * \alpha * (\\ \kappa_1 * r_l * (l_i - x_i^t) \\ + \kappa_2 * r_p * (p_i - x_i^t) \\ + \kappa_3 * r_c * (c_i - x_i^t) \end{aligned} \tag{3}$$

Pritom je $\alpha = \omega + 1 - 2\sqrt{\omega} + 4\delta\sqrt{\omega}$, a $0 < \delta < 1$ dodatni parametar algoritma. Parametri ω, κ_1 i κ_2 imaju istu interpretaciju kao i C_1, C_2 i C_3 iz osnovne inačice, dok dodatni član c_i predstavlja i-tu komponentu najboljeg pronađenog kooperativnog rješenja, a κ_3 je pripadni faktor povjerenja. Po izrazu (3) kvalitativno ponašanje algoritma ovisi o parametrima na sljedeći način:

- za $0 \leq \omega < 1$ i ω koja se približava nuli, sustav će težiti konvergenciji, dok će u suprotnom težiti divergenciji,
- za $0 < \alpha < 1$ i α koji se približava jedinici, dinamika sustava postaje titrajuća te
- za $0 < \kappa_1 + \kappa_2 + \kappa_3 < 1$ i za bilo koji od parametara κ da teži u 1 sustav se kreće ka pripadnom optimumu.

Posljednja izmjena dodana u algoritam odnosi se na način određivanja susjedstva. Susjedstva se izračunavaju u svakom koraku, a čine ih sve čestice relativno udaljene od

razmatrane čestice za manje od granične vrijednosti. Sama granična vrijednost se mijenja u svakoj iteraciji, tako što linearno raste po izrazu danom u izrazu (4).

$$l = 0.5 + 0.5 * \left(\frac{trenutačna\ iteracija}{\max broj\ iteracija}\right). \quad (4)$$

Dvije čestice su susjedi ako je $l > 0.8$ ili ako je relativna udaljenost dvaju čestica, odnosno omjer njihove udaljenosti i maksimalne udaljenosti dvaju čestica u roju, manja od l. Povećanjem vrijednost ovog parametra povećava se i radijus susjedstva, te će nakon određene iteracije čestice biti potpuno povezane.

Pseudokod 3 - SAPSO

```
1. Inicijaliziraj podčestice
2. Organiziraj čestice u podrojeve
3. Odredi susjedstvo za svaku podčesticu
4. Izračunaj dobrotu podčestica
5. Ponavljaj
6.   Izračunaj novi položaj svih podčestica koristeći izraze 3) i 2)
7.   Odredi novo susjedstvo za svaku podčesticu
8.   Izračunaj dobrotu podčestica
9. Dok nije zadovoljen uvjet zaustavljanja
```

Računanje dobrote:

```
1. Ponavljaj
2.   Iz svakog podroja nasumično odaberi jednu podčesticu
3.   Izračunaj dobrotu čestice sastavljene od odabranih podčestica
4.   Dobroti svake podčestice u čestici dodaj dobrotu čestice
5. Dok svaka podčestica nije odabrana dovoljan broj puta
6. Podijeli dobrotu svake podčestice s brojem čestica u kojima je
   podčestica sudjelovala
```

4.3 IPSO

Ranije smo u tekstu spominjali postojanje hibridnih optimizacijskih algoritama te smo na tragu te ideje u GAPSO-u vidjeli korištenje operator križanja iz genetskog algoritma u postupku algoritma roja čestica. Na primjeru inačice IPSO (Immune-based Particle Swarm Optimization) (Lin, 2008) vidjet ćemo jedan punokrvni hibridni algoritam koji svoj rad temelji na imunološkom algoritmu, a jedan od njegovih koraka koristi algoritam roja čestica.

Imunološki algoritam još je jedan popularan optimizacijski algoritam. Nastao je istraživanjem umjetnih imunoloških sustava, koji predstavljaju skupinu inteligentnih sustava inspiriranih principima i procesima imunološkog sustava kralješnjaka. Kad u tijelo uđe nametnička stanica, kojeg nazivamo antigenom, stanice imunološkog sustava (B-stanice) započnu stvarati antitijela. Antitijela imaju receptore kojima se povezuju s antigenima kako bi iste uništili. Hoće li se i kojom jačinom antitijelo i antigen povezati određuje afinitet određenog antitijela spram određenog antigena. Uspješnost spajanja antitijela na antigen potiče B-stanicu koja je proizvela antitijelo na diobu, a time i stvaranjem (nesavršenih) kopija same sebe. Nove B-stanice i njihova antitijela mogu biti bolje ili lošije prilagođena trenutačnom antigenu, odnosno potaknuti ili zaustaviti diobu pojedine B-stanice. Za problem optimizacije, imunološki algoritam koristi populaciju potencijalnih rješenja, antitijela, kako bi pronašao optimum funkcije koja opisuje problem, antigen, nastojeći maksimizirati dobrotu te funkcije, afinitet.

IPSO inačica u svakom koraku stvara nesavršene klonove trenutačnih antitijela te nad svakom skupinom klonova provodi mutaciju pokretanjem standardne inačice algoritma roja čestica. Korištenjem algoritma roja čestica postiže se veću raznolikost kako bi se povećao kapacitet globalnog pretraživanja. Dodatno, u svakom koraku se uspoređuju antitijela u populaciji na način da se uspoređuju njihov međusobni afinitet koji se računa pomoću izraza (7). U slučaju da je afinitet između dva antitijela manji od zadane granice, antitijelo s manjim afinitetom se mutira.

$$IE_l(N) = \sum_{i=1}^{N} -P_{il} \log P_{il} \tag{5}$$

$$IE(N) = \frac{1}{L} \sum_{l=1}^{L} IE_l(N) \tag{6}$$

$$Afinitet\_Antitijela_{jk} = \frac{1}{1 + IE(2)} \tag{7}$$

Pritom su:

- l – l-ta dimenzija antitijela/rješenja,
- L – dužina antitijela,

- N – broj antitijela te
- P_{il} – vjerojatnost da je l-ta dimenzija poprimila trenutnu vrijednost.

S obzirom na to da računanje entropije, odnosno vjerojatnosti da su komponente antitijela poprimile određene vrijednosti, nije izvedivo za realne vrijednosti antitijela treba diskretizirati[5]. Diskretizacija se provodi podjelom intervala na D dijelova, nakon čega se za svakoj komponenti antitijela dodijeli indeks podintervala kojem ona pripada, koji predstavlja njenu diskretnu vrijednost. Kako bi se na koncu izračunala vjerojatnost pojave određene vrijednosti za l-tu komponentu, podijeli se broj pojave te vrijednosti u l-toj komponenti za sva antitijela s brojem antitijela.

Kako se ne bi desilo da dobro rješenje bude izbačeno iz populacije algoritam u svakom koraku zadržava najbolje antitijelo, zbog čega kažemo da je algoritam elitistički.

Pseudokod 4 - IPSO

```
1. Inicijaliziraj antitijela
2. Izračunaj afinitete antitijela
3. Ponavljaj
4.   Zapamti najbolje antitijelo
5.   Kloniraj antitijela uz male mutacije
6.   Mutiraj klonove pomoću osnovne inačice algoritma roja čestica
7.   Izračunaj afinitete klonova
8.   Zamijeni antitijela najboljim klonovima
9.   Za svaki par antitijela
10.    Izračunaj međusobni afinitet
11.    Ako je afinitet veći od zadane granice mutiraj lošije antitijelo
12.  Ponavljaj
13. Dok nije zadovoljen uvjet zaustavljanja
```

4.4 MCPSO

U ranije opisanim algoritmima smo imali prilike vidjeti korištenje podrojeva, međutim njihovo je kretanje bilo međusobno neovisno, odnosno nisu utjecali jedno na drugo. U MCPSO (Multi-swarm Cooperative Particle Swarm Optimization) inačici (Niu, 2008) vidjet ćemo jedan drugačiji pristup inspiriran simbiozom u ekosustavima.

Algoritam se sastoji od podrojeva organiziranih po modelu gospodar-podanik, u kojem jedan podroj predstavlja glavni roj, dok su ostali podanički rojevi. Svaki podanički

[5] Vjerojatnost odabira realnog broja u bilo kojem realnom intervalu, nenulte dužine, je 0.

roj samostalno izvodi jednu inačicu algoritma roja čestica kako bi održavali raznolikost čestica, dok glavni roj evoluira na temelju svojeg i znanja prikupljenog od podaničkih rojeva. Također, s obzirom na moguće odnose između roja gospodara i podaničkih rojeva razvijene su dvije inačice algoritma, ovisno temelji li glavni roj svoju evoluciju na antagonističkom ili sinergističkom scenariju. U prvom slučaju glavni roj evoluira svoje čestice na temelju izravnog natjecanja s podaničkim rojevima, odnosno najbolja čestica iz svih rojeva (glavnog i podaničkih) ima priliku voditi pretraživanje prostora rješenja, te ga nazivamo kompetitivni MCPSO (izrazi (8), (9) i (10)). S druge strane, u sinergističkom scenariju, pretraživanje vode dvije čestice, najbolja čestica glavnog roja i najbolja čestica iz podaničkih rojeva, te ga nazivamo združeni MCPSO (izrazi (9) i (11)).

$$\begin{aligned} v_{Mi}^{t+1} = {} & \omega * v_{Mi}^{t} + c_1 * r_1 * (p_{Mi} - x_{Mi}^{t}) \\ & + \phi * c_2 * r_2 * (g_{Mi} - x_{Mi}^{t}) \\ & + (1 - \phi) * c_3 * r_3 * (g_{Si} - x_{Mi}^{t}) \end{aligned} \tag{8}$$

$$x_{Mi}^{t+1} = x_{Mi}^{t} + v_{Mi}^{t+1} \tag{9}$$

$$\phi = \begin{cases} 0 & Gbest^{S} < Gbest^{M} \\ 0.5 & Gbest^{S} = Gbest^{M} \\ 1 & Gbest^{S} > Gbest^{M} \end{cases} \tag{10}$$

$$\begin{aligned} v_{Mi}^{t+1} = {} & \omega * v_{Mi}^{t} + c_1 * r_1 * (p_{Mi} - x_{Mi}^{t}) \\ & + c_2 * r_2 * (g_{Mi} - x_{Mi}^{t}) \\ & + c_3 * r_3 * (g_{Si} - x_{Mi}^{t}) \end{aligned} \tag{11}$$

Tablica 1 - Oznake korištene u izrazima za MCPSO

M	Roj gospodar
S	Roj podanik
g	Najbolja čestica pripadnog roja
p	Najbolje individualno rješenje
ϕ	Migracijskih faktor
r	Nasumičan broj iz intervala [0, 1]
ω	Faktor inercije

Na rad ove inačice možemo promatrati kao na opetovano izvođenje osnovne inačice algoritma roja čestica s razlikom što postoji roji koji evoluira na temelju informacija prikupljenih tijekom njihovog izvođenja. Drugim riječima, podrojevi osiguravaju raznolikost informacija i osiguravaju pretragu većeg dijela prostora rješenja uz odvojeni roj koji vrši istraživanje obećavajućih regija. S obzirom na činjenicu da podanički rojevi evoluiraju međusobno neovisno jedan o drugom što omogućuje njihovo paralelno izvođenje.

Pseudokod 5 - MCPSO

```
1. Inicijaliziraj čestice glavnog roja
2. Stvori podaničke rojeve i inicijaliziraj njihove čestice
3. Ponavljaj
4.   Izračunaj dobrote svih čestica
5.   Paralelno
6.     Pokreni osnovnu inačicu algoritma roja čestica
7.   Za sve rojeve podanike
8.   Čekaj završetak svih rojeva
9.   Dohvati najbolju česticu iz svih podaničkih rojeva
10.  Evoluiraj glavni roj
11. Dok nije zadovoljen uvjet zaustavljanja
```

4.5 MRPSO

Zadnja inačica (Vanneschi, 2010) algoritma roja čestica koju ćemo razmotriti, MRPSO (Multi-swarm Repulsive Particle Swarm Optimization), kao i prethodna koristi veći broj rojeva s razlikom da su svi podrojevi jednaki, ali dolazi do razmjene čestica među rojevima te postoji proces koji nastoji udaljavati rojeve kako im se područja pretraživanja ne bi preklapala.

Algoritam radi na način da se čestice razmjenjuju uvijek između istih rojeva, tako što se na početku formira prstenasta struktura u kojoj svaki roj prosljeđuje k najboljih čestica sljedećem roju u kojem se one zamijene s k najlošijih čestica u njemu. Tako definirana struktura se također koristi za međusobno udaljavanje rojeva tako što se rojevi kreću prema svojem globalno najboljem rješenju, a odmiču od globalno najboljeg rješenja prethodnog roja u strukturi. Odbijanje se postiže dodatnim članom u izrazu za računanje nove brzine čestica (12), pritom je sa g označeno globalno najbolje rješenje a sa fg

globalno najbolje rješenje prethodnog roja. Svaka čestica u roju se nastoji odmaknuti od globalno najbolje čestice roja koji prethodi u strukturi (najbolje strano rješenje), osim u slučaju kad se najbolje strano rješenje nalazi između čestice čija se brzina izračunava i globalno najboljeg rješenja tog roja. U tom se slučaju čestica dodatno ubrza u smjeru globalno najboljeg rješenja roja. Utjecaj najboljeg stranog rješenja je time veći što je čestica čija se brzina računa bliža njemu, kako je prikazano u izrazu (14).

Kao i u prethodnoj inačici podrojevi omogućavaju pretragu većeg dijela prostora stanja, a s odbojnom komponentom omogućava da ne dođe do preklapanja područja istraživanja susjednih rojeva s obzirom na to da razmjenjuju čestice. Time je omogućena razmjena informacija među svim podrojevima iako je brzina propagacije iste uvjetovana postavljenom topologijom povezanosti podrojeva.

$$\begin{gathered} v_i^{t+1} = \omega * v_i^t + \\ C_1 * r_p * (p_i - x_i^t) + \\ C_2 * r_g * (g_i - x_i^t) + \\ C_3 * r_{fg} * f(fg_i,\ x_i^t,\ g_i) \end{gathered} \tag{12}$$

$$f(fg, x, g) = \begin{cases} -\phi(x, fg) & fg\ između\ x\ i\ g \\ \phi(x, fg) & inače \end{cases} \tag{13}$$

$$\phi(x, fg) = sig(dis) * \left(1 - \left|\frac{udaljenost\ čestica}{maksimalna\ udaljenost}\right|\right) \tag{14}$$

Pseudokod 6 - MRPSO

```
1. Inicijaliziraj podrojeve i njihove čestice
2. Organiziraj podrojeve u prstenastu strukturu
3. Ponavljaj
4.   Izračunaj dobrotu čestica u svim podrojevima
5.   Sve parne podrojeve udalji od susjeda s nižim indeksom
6.   Pokreni algoritam roja čestica nad svim podrojevima
7.   Iz svakog podroja odaberi k najboljih čestica te njima zamijeni k
     najgorih čestica iz podroja s većim indeksom
8. Dok nije zadovoljen uvjet zaustavljanja
```

5. Opis problema za usporedbu

Kako bi usporedili opisane inačice provest ćemo niz eksperimenata u kojima ćemo iskoristiti inačice za rješavanje skupa različitih problema. Međutim, prije nego krenemo s usporedbom navest ćemo korišteni skup problema i opisati njihove karakteristike. Vidjet ćemo najčešće korišteni skup problema za usporedbu optimizacijskih algoritama i problem učenja mreže ANFIS, koju ćemo usput i opisati.

5.1 Standardni skup problema

Ovaj skup problema (Niu, 2007) (Vesterstrom, 2004) (Eberhart, 2000) (Shi, 2001) čine nelinearne funkcije koje spadaju u probleme minimizacije, a karakterizira ih postojanje jednog minimuma s vrijednošću 0 i to u ishodištu koordinatnog sustava (osim za Rosenbrock funkciju čiji se minimum nalazi u točki [1, 1, ...]). Nadalje, osim funkcija Sphere i Rosenbrock za male dimenzije, sve funkcije su multimodalne, odnosno imaju više lokalnih optimuma. Važno je napomenuti i da su sve funkcije simetrične što ih razlikuje od praktičnih problema koje karakterizira visok stupanj nepravilnosti. Dimenzionalnost opisanih problema je proizvoljna, a porastom dimenzionalnosti raste i težina pronalaženja minimuma. Također, svaka funkcija predstavlja drugačiju prepreku u pronalaženju globalnog optimuma, tako je za funkciju Rosenbrock lako naći dolinu u kojoj leži globalni optimum, ali je sam optimum teško pronaći, dok funkciju Griewangk karakterizira visok stupanj multimodalnosti.

$$Sphere(\vec{x}) = \sum_{i}^{n} x_i^2 \tag{15}$$

$$Schwefel\_2.22(\vec{x}) = \sum_{i}^{n} |x_i| + \prod_{i}^{n} |x_i| \tag{16}$$

$$Rosenbrock(\vec{x}) = \sum_{i}^{n-1} 100 * \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2 \tag{17}$$

$$Quartic(\vec{x}) = \sum_{i}^{n} i * x_i^4 + rand[0,1) \tag{18}$$

$$Rastrigrin(\vec{x}) = \sum_{i}^{n} (x_i^2 - 10 * \cos(2\pi x_i) + 10) \tag{19}$$

$$Griewangk(\vec{x}) = \frac{1}{4000}\sum_{i}^{n} x_i^2 - \prod_{i}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{20}$$

5.2 Učenje mreže ANFIS

5.2.1 Općenito o neizrazitim neuronskim mrežama

ANFIS (Adaptive Neuro Fuzzy Inference System) (Jang, 1993) je hibridan pristup kombiniranja umjetnih neuronskih mreža i sustava neizrazitog zaključivanja, koji se u pravilu koriste za aproksimaciju nepoznatih funkcija. Motivacija za njihovo kombiniranje leži u dokazu da su neuronske mreže i sustavi neizrazitog zaključivanja ekvivalentni i jednako ekspresivni pristupi, zbog čega bi njihovom kombinacijom nastoje poništiti njihove pojedinačne mane, zadržavajući prednosti (Dalbelo Bašić, 2012).

Rezultat je sustav koji omogućava korištenje dobro poznatih algoritama za učenje umjetnih neuronskih mreža, koji se ne mogu koristiti u sustavima za neizrazito zaključivanje, dok se istovremeno zadržava mogućnost korištenja neizrazite logike i neizrazitog zaključivanja. Postoje razne inačice mreža ANFIS, dok ćemo mi u ovom radu razmotriti TSK[6] mrežu (Takagi, 1985) koja se temelji na neizrazitom modelu u kojem konzekvens pravila koristi linearnu kombinaciju njenih ulaznih varijabli sljedećeg oblika:

$$\begin{aligned} &Ako\ x_1\ je\ A_{1j}(m_{1j}, \sigma_{1j})\ i\ x_2\ je\ A_{2j}(m_{2j}, \sigma_{2j}) \ldots i\ x_n\ je\ A_{nj}(m_{nj}, \sigma_{nj}) \\ &onda\ y = w_{0j} + w_{1j}x_1 + \cdots + w_{nj}x_n. \end{aligned} \tag{21}$$

Mrežu čini 5 slojeva, sačinjenih od neurona koji nad ulaznim vrijednostima primjenjuju različite funkcije, u ovisnosti o sloju, te ne postoje povratne veze. Prvi sloj je ulazni sloj i on sadrži n neurona, po jedan neuron za svaku od n dimenzija ulaza, a funkcija im je preslikavanje ulaznih vrijednosti u sljedeći sloj. Ako ulazne varijable označimo s x_i te izlaze prvog sloja s $u_i^{(1)}$ izraz za prvi sloj glasi

$$u_i^{(1)} = x_i. \tag{22}$$

[6] Inačica je dobila ime po svojim izumiteljima: Takagi, Sugeno i Kang.

U drugom sloju se nalaze neuroni koji primjenjuju funkcije pripadnosti i time svaku ulaznu vrijednost preslikavaju u razinu pripadnosti neizrazitim skupovima pri čemu se uglavnom koristi Gaussova funkcija pripadnosti. Izlaz $j$te funkcije pripadnosti za $i$tu varijablu je dan izrazom (23).

$$u_{ij}^{(2)} = exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \tag{23}$$

Pritom su m_{ij} i σ_{ij} redom očekivanje i varijanca $j$te funkcije pripadnosti $i$tog ulaznog neurona (ulazne varijable). U trećem se sloju računaju jačine odziva pojedinih pravila korištenjem neizrazitog *I* operatora. Izlaz je dan izrazom (24).

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{24}$$

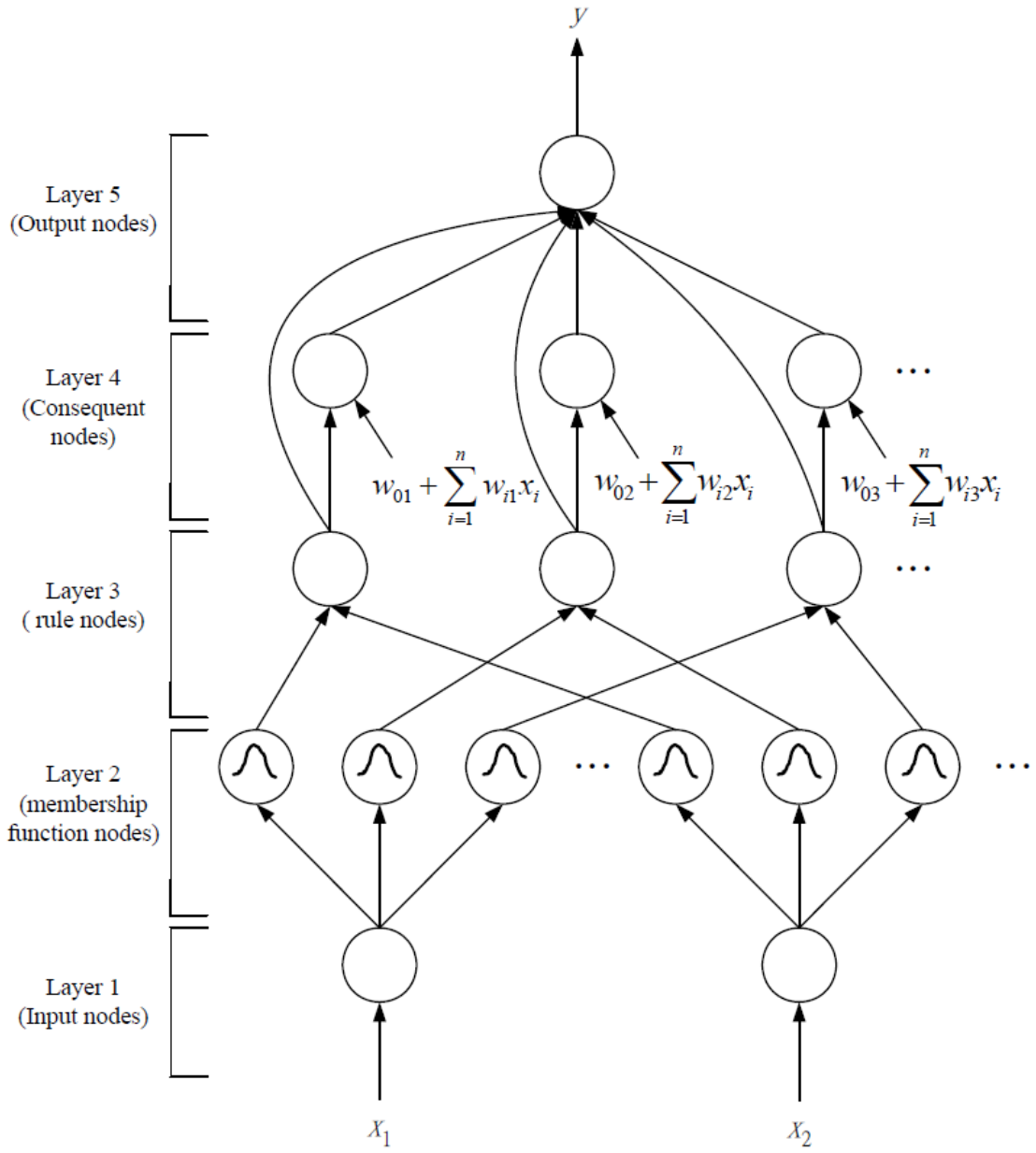
Četvrti sloj čine neuroni s funkcijama koje računaju konzekvens za što je, uz izlaze iz prethodnog sloja, potrebno dovesti i izlaze iz prvog sloja. Sukladno izrazu (25) izlaz ovog sloja glasi

$$u_j^{(4)} = u_j^{(3)}(w_{0j} + \sum_i^n w_{ij}\, u_i^{(1)}) \tag{25}$$

gdje su w_{ij} parametri konzekvensa i poprimaju vrijednosti iz domene realnih brojeva. U posljednjem sloju se vrši defazifikacija te se pritom koriste vrijednosti iz prethodna dva sloja kao što je opisano izrazom (26), gdje R predstavlja broj pravila mreže.

$$y = u^{(5)} = \frac{\sum_j^R u_j^{(4)}}{\sum_j^R u_j^{(3)}} \tag{26}$$

Prikaz cjelokupne mreže je moguće vidjeti na slici 5 (Lin, 2006).



Slika 5 – Mreža TSK ANFIS

5.2.2 Parametri

Valja primijetiti da do sada nismo nigdje definirali broj pravila koje će mreža koristiti s obzirom na to što i on predstavlja parametar mreže. Međutim, dok drugi parametri koje smo do sada razmotrili utječu samo na oblik funkcija u neuronima, broj pravila određuje strukturu mreže, a time i broj ostalih parametara. U ovom radu neće biti razmotreno učenje strukture mreže ANFIS te ćemo koristiti ili proizvoljno odabrani broj ili jednostavan algoritam grupiranja nad ulaznim podacima te će nam broj grupa

predstavljati broj pravila[7] (Lin, 2006). Stoga se za nas problem učenja mreže ANFIS svodi se na pronalaženje vrijednosti parametara m_{ij}, σ_{ij} i w_{ij} za koje je greška minimalna, odnosno za koje je razlika u odzivu mreže i očekivanog izlaza što manja.

Što se broja parametara tiče, određuje ga broj pravila, a točan broj je dan izrazom (27). Gdje R predstavlja broj pravila, a n broj/dimenziju ulaza. Svaki ulaz ima R funkcija pripadnosti koje su određene dvama parametrima (očekivanje i varijanca Gaussove razdiobe) te R konzekventnih neurona za koje je potrebno definirati $n+1$ parametara težinske sume.

$$Broj\ parametara = R * 2 * n\ +\ R * (n+1) = R * (n * 3 + 1) \tag{27}$$

5.2.3 Primjena

S obzirom na činjenicu da mreže ANFIS spadaju u skupinu univerzalnih estimatora[8] primijenjene su na velik broj problema u raznim područjima. Među ostalim primijenjene su na problem prepoznavanja boje puti na slikama, raznim problemima u geotehnologiji (Cabalar, 2012), robotici (Aware, 2000), kontrolnim sustavima (Tzafestas, 2002) te bioinformatici (Chuang, 2009).

U ovom ćemo radu razmotriti korištenje mreže ANFIS za učenje jednog nelinearnog dinamičkog sustava, pri čemu će za učenje mreže biti korištene razne inačice algoritma roja čestica. Sustav je zadan izrazima (28), (29) i (30), a njegov grafički prikaz možemo vidjeti na slici 6 (Ghomsheh, 2007).

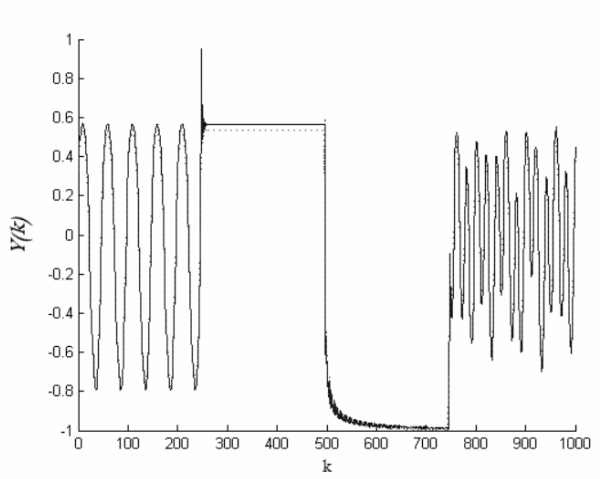
$$y_p(k+1) = f(y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)) \tag{28}$$

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_2^2 + x_3^2} \tag{29}$$

$$u(k) = \begin{cases} \sin(\pi k/25) & 0 < k < 250 \\ 1.0 & 250 \leq k < 500 \\ -1.0 & 500 \leq k < 750 \\ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/25) & \\ \quad + 0.6\sin(\pi k/10) & 750 \leq k < 1000 \end{cases} \tag{30}$$

[7] Grupe podataka možemo gledati kao skupove ulaznih vrijednosti za koje se sustav ponaša po jednakim pravilima.

[8] Mogu aproksimirati bilo koju kontinuiranu funkciju s proizvoljnom točnošću.



Slika 6 - Graf nelinearnog dinamičkog sustava

Za učenje mreže korišten je skup od 100 nasumično odabranih ulaznih i pripadnih izlaznih podataka generiran danim izrazom. S obzirom na izgled funkcije naizgled se logičnim čini struktura mreže s četiri pravila, koja bi odgovarala područjima na kojima funkcija prikazuje određenu vrstu pravilnosti. Općenito su mreže ANFIS pogodne za ovakvu vrstu funkcija, koju, na primjer, često susrećemo u kontrolnim sustavima, gdje se sustav za određena područja faznog prostora[9] ponaša po određenim pravilnostima.

[9] Prostor stanja, svaka točka faznog prostora opisuje sustav u cjelini, odnosno sve njegove parametre.

6. Usporedba algoritama

Kako bi dobili uvid u performanse pojedinih algoritama usporedili smo njihovo ponašanje na skupu ranije definiranih problema. Također, da bi opravdali korištenje kompleksnijih, odnosno računarski zahtjevnijih inačica, u usporedbu je dodana i osnovna inačica algoritma. Međutim prije nego smo ih mogli međusobno usporediti bilo je potrebno za svaki naći i vrijednosti parametara za koje se najbolje ponašaju na skupu problema.

6.1 Opis procedure za usporedbu

S obzirom na to da je detaljna analiza ponašanja svih algoritama za svaki problem zasebno u ovisnosti o njihovim parametrima izvan opsega ovog rada, provedena je pojednostavljena analiza i to za mali broj parametara. Stoga su parametri odabrani uzimajući u obzir parametre korištene u njihovim izvornim člancima uz seriju testova koji su imali za cilj optimiranje skupa parametara za ovaj konkretni skup problema. U tim je testovima svaki algoritam pokrenut s raznim vrijednostima parametara za koje su autori smatrali da imaju značajan utjecaj na njegove performanse, zadržavajući ostale parametre konstantnima. Pritom je svaki algoritam na svakom problemu iz standardnog skupa pokrenut 30 puta za svaki skup parametara s fiksnim brojem iteracija, konkretno 200. Na kraju izvođenja, za svaku se instancu inačice izračunala prosječna vrijednost funkcije kroz sva pokretanja, za najbolje pronađeno rješenje u pojedinom pokretanju. Svakoj, tako dobivenoj, prosječnoj vrijednosti je dodijeljen redni broj od 1 do n, gdje je n broj različitih skupova parametara s kojim je inačica testirana, a pritom manji broj označava bolji rezultat. Za svaku instancu inačice sumirali su se redni brojevi. Skup parametara za koji je tako definirana suma bila minimalna uzet je kao optimalan. Vrijedi napomenuti da je u pravilu pronađeni skup parametara davao bolje rezultate od onih preporučenih u izvornom članku.

Za primjer ćemo razmotriti odabir parametara za MCPSO inačicu. Prosječne minimalne vrijednosti funkcija pronađenih za različite vrijednosti parametara možemo vidjeti u tablici 2. Parametri čije su se vrijednosti nastojale optimirati su broj podrojeva, broj iteracija podrojeva u svakom koraku te broj čestica u podrojevima. Za druge parametre je zadržana vrijednost definirana u izvornom članku zato što je inačica

korištena na istom skupu problema te možemo pretpostaviti da su autori pronašli prikladne vrijednosti. Pokazalo se da dopuštanje većeg broja iteracija podrojevima znatno doprinosi performansama što vidimo po inačicama 4, 8 i 12 za koje je spomenuti parametar bio postavljen na 20. Znatno manji utjecaj ali i dalje primjetan je imalo korištenje većeg broja podrojeva s manjim brojem čestica, te se pokazalo da korištenje 10 podrojeva s po 5 čestica svaki (inačice 7, 8 i 15), daje bolje rezultate od preporučenih 3 podroja i 20 čestica (inačice 1 i 9).

Vrijedi primijetiti kako inačica 8 daje znatno bolje rezultate problem traženja minimuma funkcije Rosenbrock. Razlog tome je što je za odabir njenih parametara posvećena posebna pažnja s obzirom na činjenicu da su sve inačice pronalazile veoma loša rješenja za taj problem. Jedino su MCPSO i MRPSO algoritmi dali naznake da bi mogli biti prigodni za pronalaženje optimuma te funkcije, te su vrijednosti parametara za MCPSO uspješno pronađeni[10]. Međutim, za razliku od inačice 12 koja pokazuje relativno dobre rezultate na svim inačicama, ova inačica daje isključivo dobre rezultate samo za odabrani problem. Ovaj rezultat još jednom pokazuje da nije moguće pronaći instancu algoritma koja bi imala zadovoljavajuće performanse na problemima različitih karakteristika.

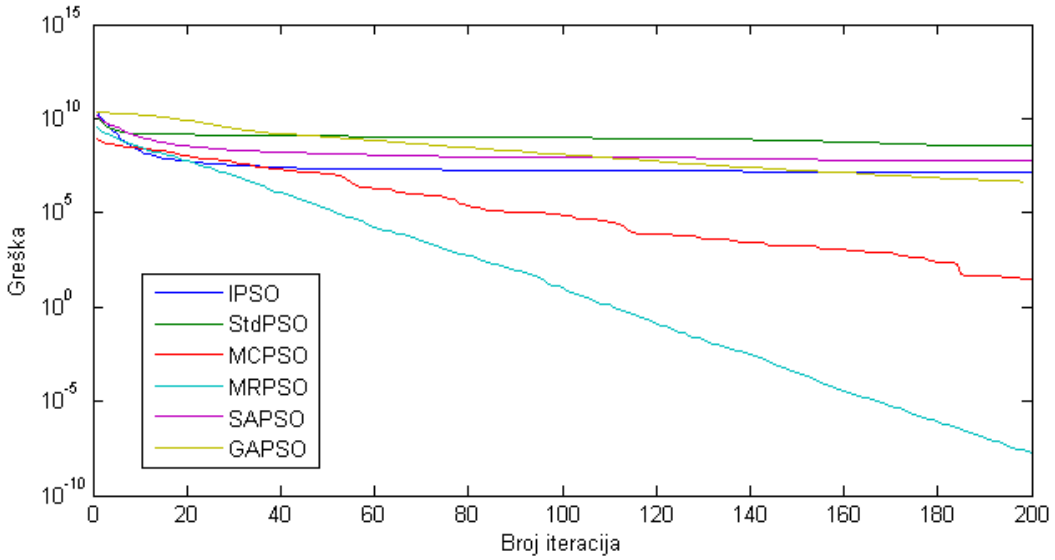
Tablica 2 - Pronađeni minimumi za različite vrijednosti parametara

Inačica	#	Sphere	Schwefel	Rosenbrock	Quartic	Rastrigrin	Griewangk
COL	1	$4.29*10^{9}$	$5.82*10^{21}$	$8.25*10^{16}$	$1.67*10^{4}$	$1.93*10^{5}$	$3.61*10^{5}$
	2	$4.17*10^{8}$	$1.53*10^{6}$	$1.75*10^{16}$	$9.38*10^{2}$	$8.39*10^{4}$	$3.11*10^{4}$
	3	$1.73*10^{8}$	$1.65*10^{4}$	$3.97*10^{15}$	$6.49*10^{2}$	$7.36*10^{4}$	$1.67*10^{4}$
	4	$4.54*10^{3}$	$5.64*10^{2}$	$1.20*10^{7}$	$3.23*10^{2}$	$6.20*10^{4}$	2.46
	5	$3.59*10^{8}$	$1.02*10^{4}$	$2.00*10^{16}$	$9.77*10^{2}$	$8.91*10^{4}$	$2.55*10^{4}$
	6	$3.88*10^{8}$	$7.19*10^{4}$	$1.54*10^{16}$	$8.70*10^{2}$	$8.70*10^{4}$	$2.97*10^{4}$
	7	$1.82*10^{8}$	$6.46*10^{3}$	$1.17*10^{16}$	$5.67*10^{2}$	$7.62*10^{4}$	$1.38*10^{4}$
	8	$1.1*10^{5}$	$1.38*10^{5}$	**2.19**	$1.94*10^{11}$	$1.81*10^{9}$	$1.59*10^{2}$
COM	9	$4.55*10^{9}$	$4.05*10^{20}$	$4.04*10^{16}$	$1.25*10^{4}$	$1.91*10^{5}$	$3.96*10^{5}$
	10	$1.97*10^{8}$	$3.84*10^{3}$	$5.16*10^{15}$	$5.32*10^{2}$	$7.48*10^{4}$	$1.75*10^{4}$
	11	$1.01*10^{8}$	$1.95*10^{3}$	$2.97*10^{14}$	$3.36*10^{2}$	$5.84*10^{4}$	$7.48*10^{3}$
	12	**30.73**	**$1.38*10^{2}$**	$4.42*10^{2}$	**$1.99*10^{2}$**	**$5.21*10^{4}$**	**0.70**
	13	$2.16*10^{8}$	$3.94*10^{3}$	$3.29*10^{15}$	$5.18*10^{2}$	$6.96*10^{4}$	$1.78*10^{4}$
	14	$1.58*10^{8}$	$3.38*10^{3}$	$3.56*10^{15}$	$4.69*10^{2}$	$6.87*10^{4}$	$1.47*10^{4}$
	15	$1.46*10^{8}$	$3.23*10^{3}$	$3.04*10^{15}$	$4.66*10^{2}$	$6.74*10^{4}$	$1.32*10^{4}$

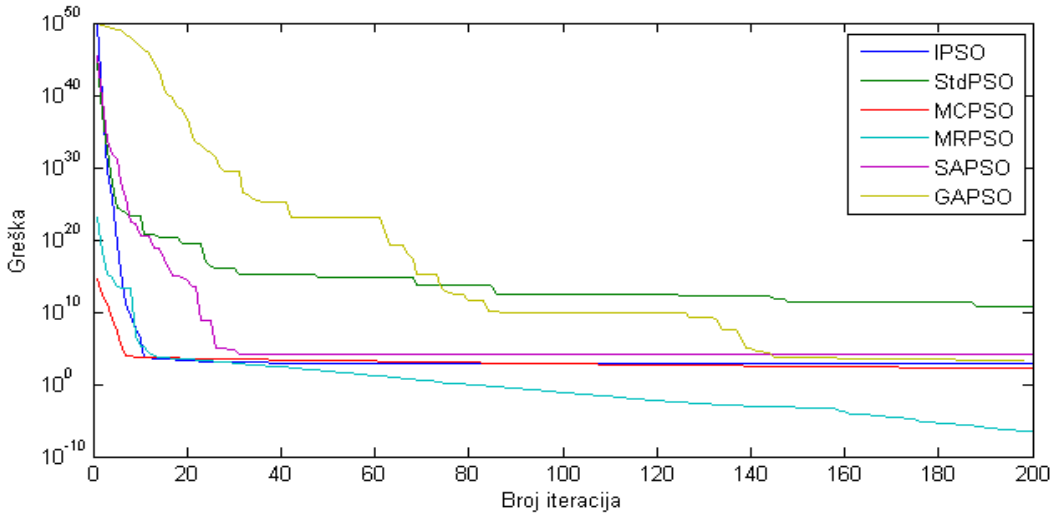
[10] Ova izjava može zvučati kontradiktorno s obzirom na to da za pokrenute testove niti jednom nije pronađen minimum funkcije, ali valja uzeti u obzir da se svaki algoritam izvodio samo 200 iteracija.

6.2 Eksperimentalni podaci i analiza

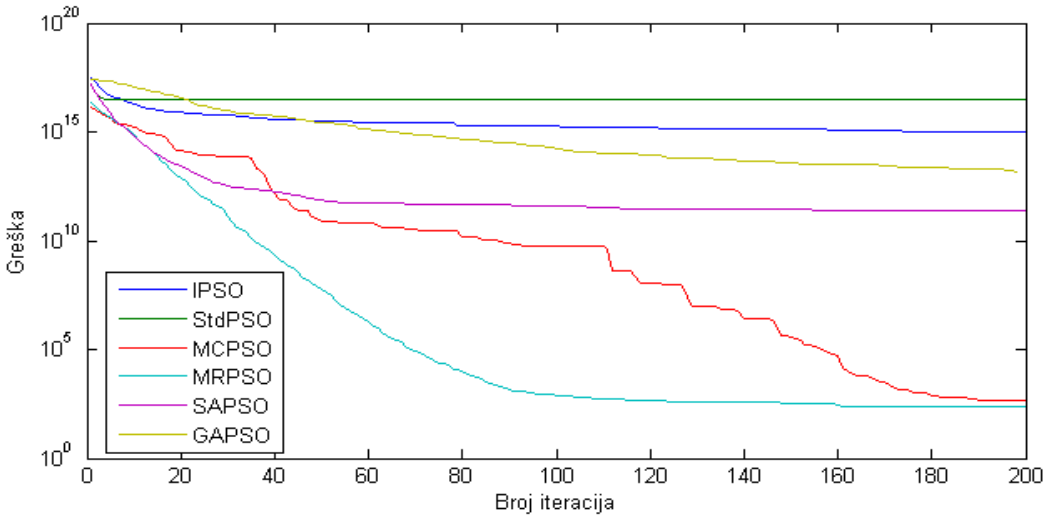
Nakon što je za svaku inačicu pronađen „optimalan“ skup parametara uspoređeno je njihovo ponašanje inačica na svim razmatranim problemima, uključujući i učenje mreže ANFIS za problem opisan u prijašnjem poglavlju. Ponašanje algoritama je promatrano razmatrajući srednju vrijednost funkcije kroz 30 pokretanja za svaku iteraciju te je ono prikazano u obliku grafova (slike 7-12). Ovaj prikaz nam daje uvid u dinamiku algoritma, napreduje li u skokovima ili postepeno, teži li ka optimumu ili zaglavljuje u lokalnim optimumima.



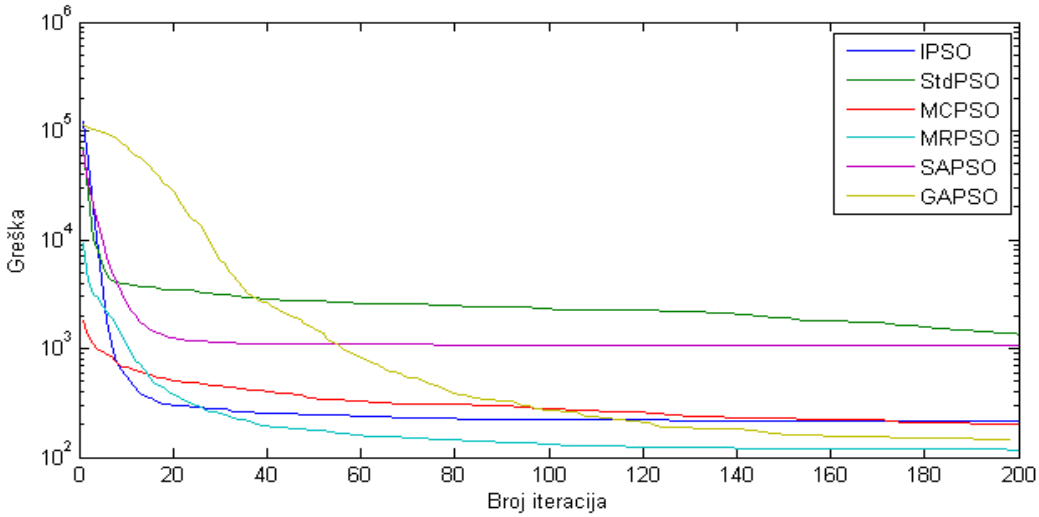
Slika 7 - Kretanje greške za funkciju Sphere



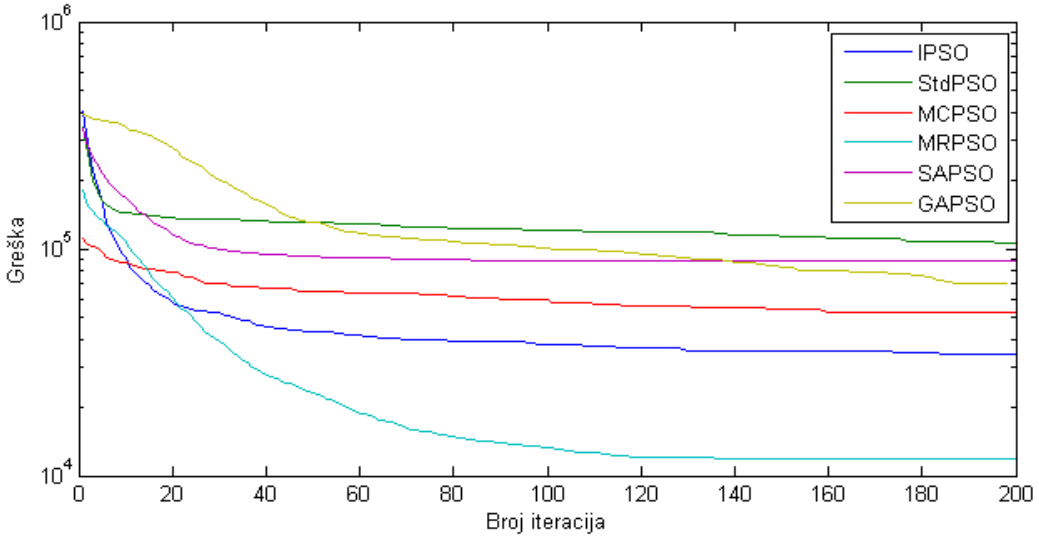
Slika 8 - Kretanje greške za funkciju Schwefel



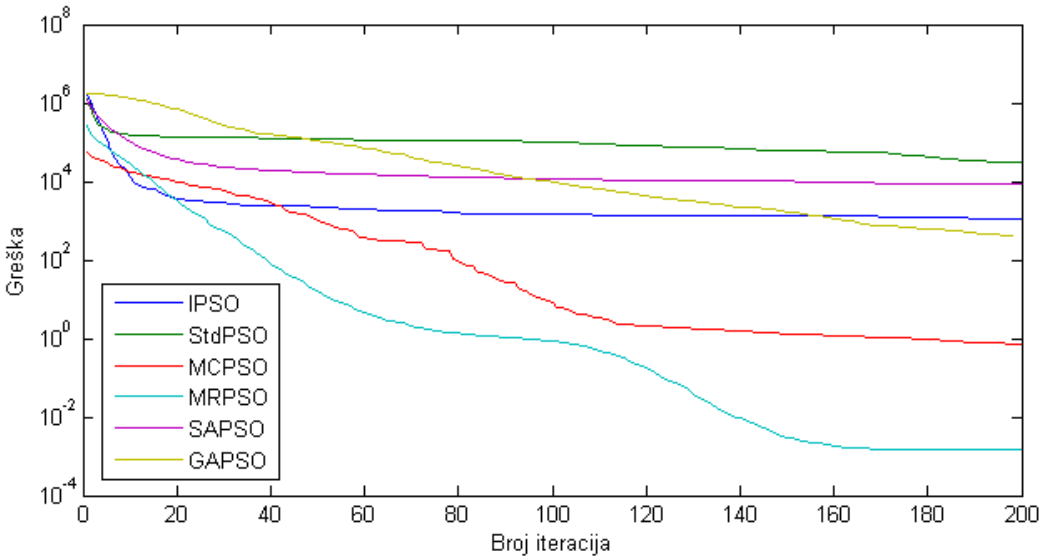
Slika 9 - Kretanje greške za funkciju Rosenbrock



Slika 10 - Kretanje greške za funkciju Quartic



Slika 11 - Kretanje greške za funkciju Rastrigrin



Slika 12 - Kretanje greške za funkciju Griewangk

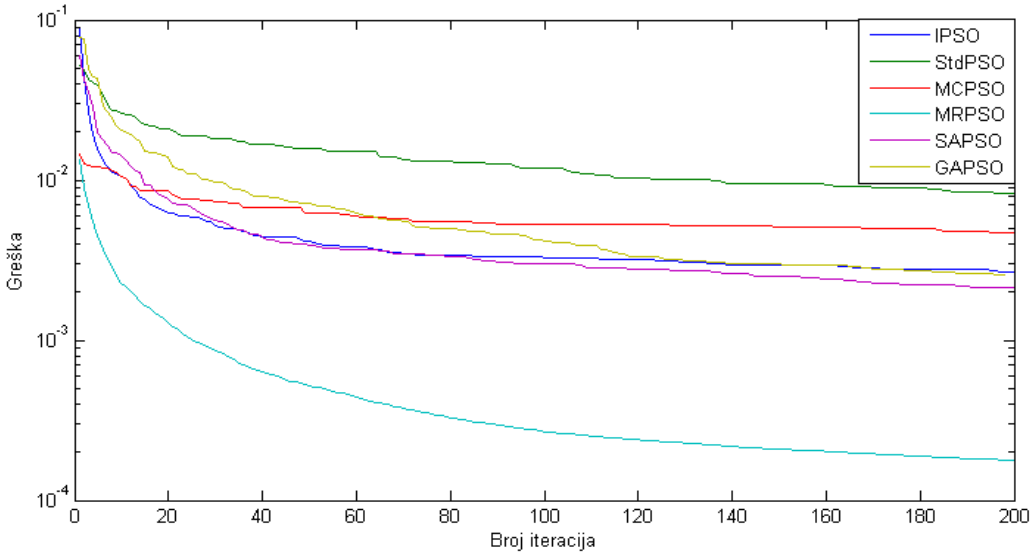
Promatrajući grafove, za koje je bitno primijetiti da je y-os u logaritamskoj skali, očekivano vidimo da osnovna inačica algoritma roja čestica ima najslabije performanse. Istovremeno MRPSO inačica je za sve probleme davala najbolje rezultate i to često za nekoliko redova veličina bolje od druge najbolje inačice. Uz nju, dosta dobre rezultate je pokazala i MCPSO inačica koja je za funkcije Sphere, Rosenbrock i Griewangk imala značajno bolje performanse od ostalih inačica, dok je u ostalima bila jednako dobra kao i ostatak naprednijih inačica. IPSO, SAPSO i GAPSO, inačice koje su razvijene za rješavanje problema učenja mreže ANFIS su u pravilu imale slične performanse na svim problemima, uz činjenicu da je GAPSO inačica pokazala najsporiju konvergenciju od svih inačica, uključujući i osnovnu.

Nadalje, možemo primijetiti da su osnovna inačica, IPSO, SAPSO na ovom skupu problema veoma rano zaglavile u lokalnim optimumima te su do svojih konačnih rezultata došli već nakon 40 iteracija. Ovo vrijedi i za MCPSO inačicu za polovicu testnih primjera. U slučaju MRPSO inačice možemo u većini slučajeva primijetiti ubrzano smanjenje greške na početku te postepeno pronalaženje sve boljih rješenja, što je upravo ponašanje koje želimo vidjeti u populacijskom metaheurističkom algoritmu.

U rijetkim slučajevima se može primijetiti skokovito ponašanje algoritma što upućuje na činjenicu da algoritmi nisu nasumično nailazili na bolja rješenja već su postepeno pronalazili nova kvalitetna područja koja su se dalje pretraživala. Najviše

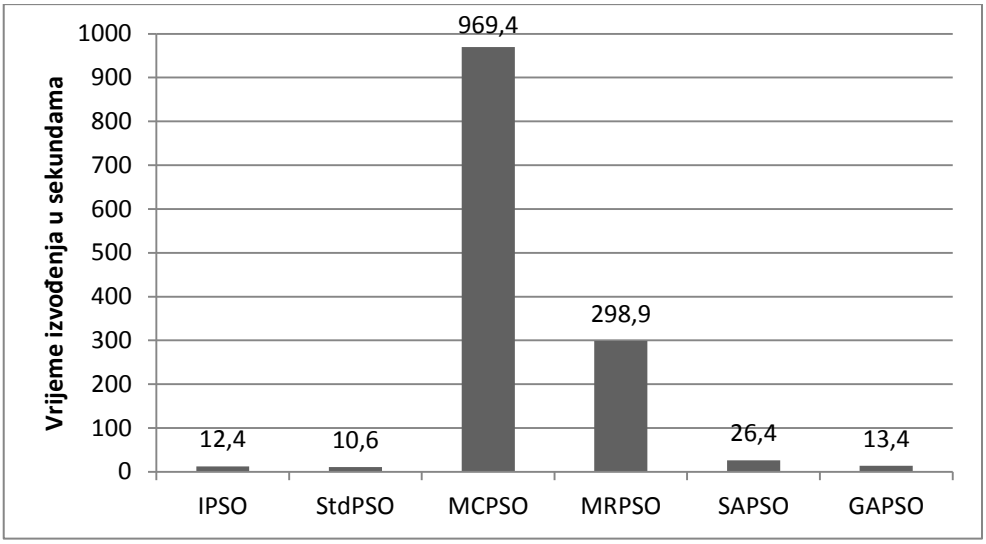
slučajeva skokovitog ponašanja možemo primijetiti kod MCPSO inačice. Takvo ponašanje možemo objasniti kretanjem podaničkih rojeva koji neovisno pretražuju prostor rješenja te u svakom trenutku jedan od njih može naići na rješenje znatno različito od trenutačno najboljeg. Prema tom će rješenju glavni roj krenuti relativno velikom brzinom zbog velike vrijednosti pripadnog parametra. Iako su i druge vrijednosti parametara za računanje brzine jednako veliki njihov će utjecaj biti malen iz razloga što su čestice veoma vjerojatno u njenoj neposrednoj blizini.

Sad kad smo razmotrili ponašanje inačica na skupu standardnih problema ostaje nam analizirati njihovo ponašanje na specifičnom problemu učenja mreže ANFIS. Za sve inačice su korišteni isti parametri kao i u prošlim testovima te je promatrana srednja kvadratna greška mreže nad skupom podataka za učenje. Iako to nije pravilan način učenja umjetnih mreža u sklopu ovog eksperimenta nije se obraćala pažnja na prenaučenost mreže i općenito kvalitetu njenog učenja kako bi se postigle dobre performanse na općem skupu ulaznih podataka. Cilj je bio dobiti uvid u ponašanje opisanih inačica na znatno različitom problem u od onih opisanih u standardnom skupu za koje je karakteristična simetričnost i pravilna raspodjela lokalnih optimuma.



Slika 13 - Kretanje greške za mrežu ANFIS

Pritom je korištena struktura s tri pravila čime je broj parametara 48 u odnosu na 64 za strukturu s četiri pravila[11] čime je ubrzan rad algoritama, a pritom nije primijećena velika razlika u kvaliteti pronađenih rješenja. Također, za razliku od prethodnog skupa problema, razmotrit ćemo i vrijeme izvođenja algoritama s obzirom na značajne razlike među inačicama. Na slici 13 možemo vidjeti kretanje prosječne greške na 20 pokretanja za sve inačice kroz 200 iteracija. Dok su na slici 14 prikazana prosječna vremena izvođenja za jedno pokretanje.



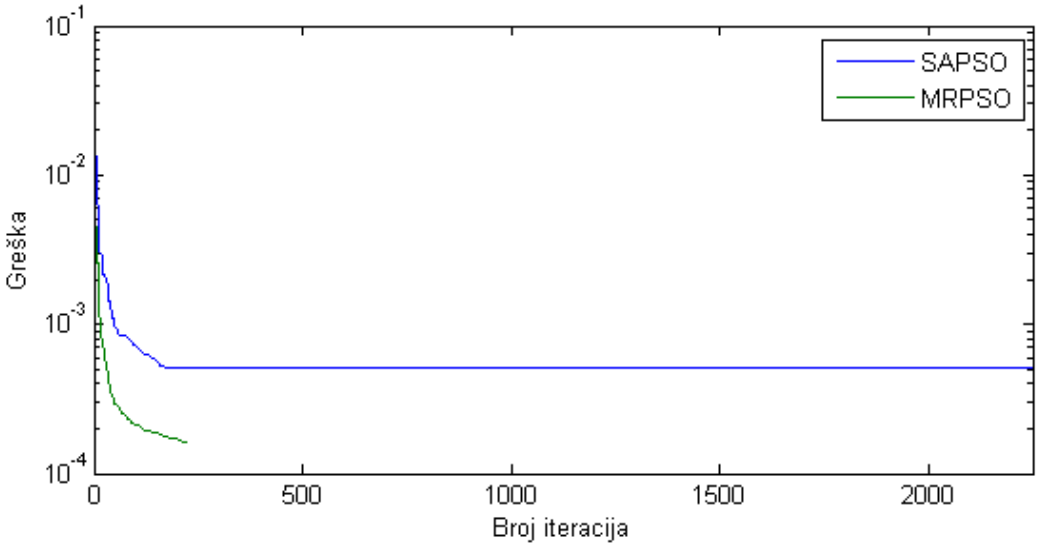
Slika 14 - Trajanje izvršavanja algoritama

Iako su tri od korištenih inačica (IPSO, SAPSO, GAPSO) napravljene posebno s namjenom učenja mreža ANFIS njihove performanse su se pokazale značajno slabijim od performansa MRPSO algoritma, koji se još jednom pokazao izrazito dobrim. Jedinu zamjerku MRPSO inačici možemo pronaći u prilično sporom izvođenju, koji je više od 10 puta sporiji od drugog algoritma po performansama (SAPSO). Pomalo neočekivani su značajno loši rezultati MCPSO inačice koja je na standardnom skupu pokazala značajno bolje rezultate.

S obzirom na značajnu razliku vremenima rada vrijedilo je ispitati kako se algoritmi odnose u slučaju kad je njihov rad ograničen vremenski, umjesto brojem iteracija. Rezultat takvog testa koji je ograničio vrijeme rada algoritma na 5 minuta je prikazan na slici 15. Unatoč sporom izvršavanju MRPSO algoritam pronalazi značajno bolja

[11] Broj parametara je određen izrazom (25).

rješenja, dok SAPSO inačica nije u stanju izvući se iz lokalnog optimuma jednom kad u njega upadne te u njemu ostaje skoro većinu vremena izvođenja algoritma.



Slika 15 - Prikaz rada za jednako vrijeme izvođenja

Iz prikazanih rezultata vidimo da inačice prilagođene učenju mreža ANFIS nisu uspjele ostvariti bolje performanse u odnosu na sve ostale algoritme. Iz ovoga zaključujemo da uvedene strategije za ovaj specifičan problem nisu pokazale bolje ili značajno bolje rezultate od ostalih inačica koje nisu razvijene za neku posebnu namjenu. Za kraj ćemo pokušati, po uzoru na Frankenstein's PSO (Montes, 2009), ukomponirati razne komponente i strategije prethodno razmatranih inačica kako bi iz njih sagradili inačicu koja će imati bolje performanse od inačica od kojih je sastavljena.

7. Razvijene inačice

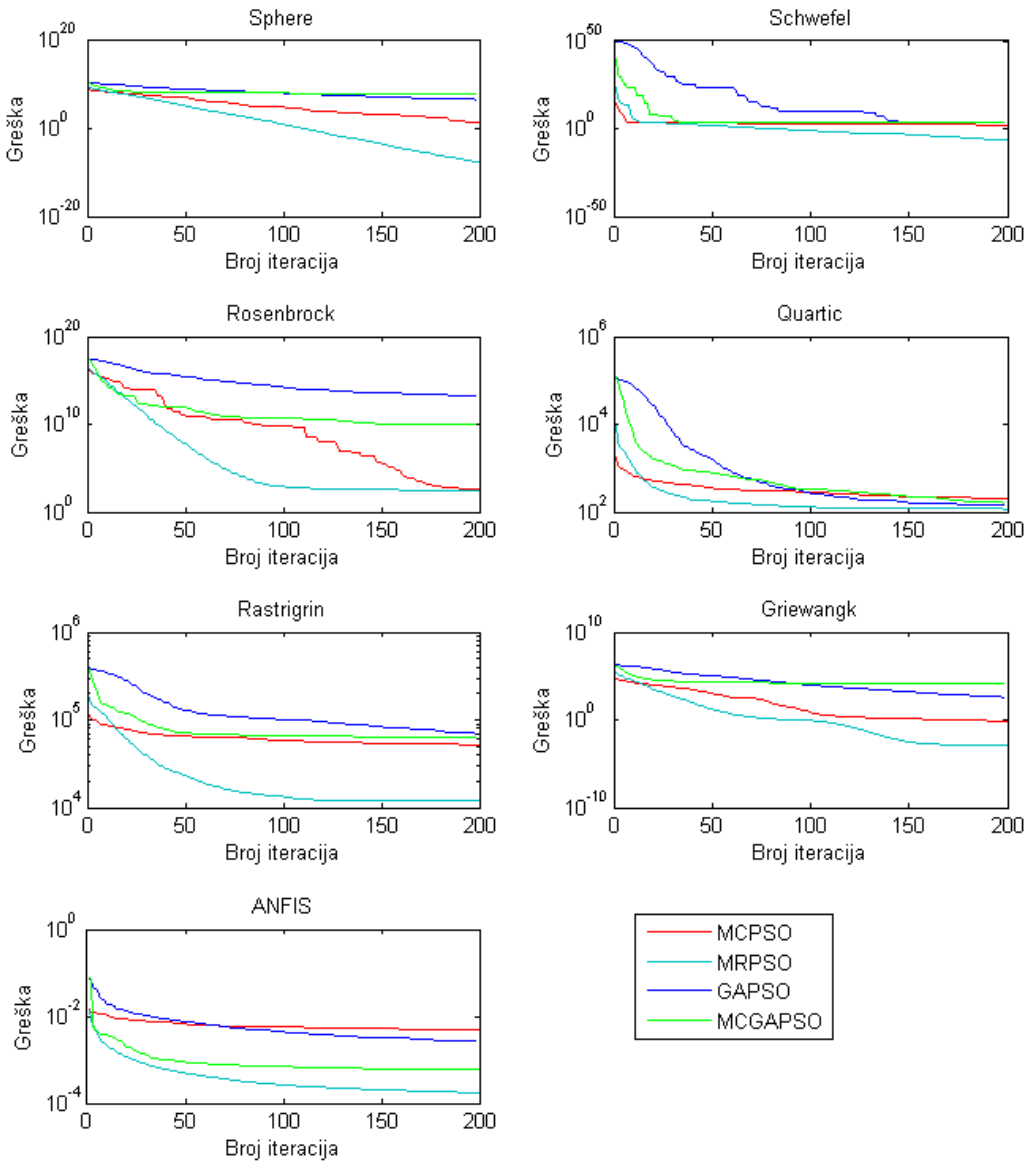
Do sad smo vidjeli razna proširenja osnovne inačice algoritma roja čestica pomoću kojih se nastojalo pomoću raznih strategija ispraviti nedostatke osnovne inačice te ponuditi dobro rješenje za konkretan problem ili skup problema. Vidjeli smo da je MRPSO inačica pokazala znatno bolje performanse od ostatka razmatranih inačica neovisno o razmatranom problemu. Prirodno se nameće pitanje ako je moguće dodatno poboljšati njene performanse dodavanjem komponenti drugih inačica. Naročito će bit zanimljivo vidjeti ako inačice specijalizirane za učenje mreže ANFIS mogu svojim strategijama poboljšati rad MRPSO-a na tom problemu. U nastavku će biti opisano nekoliko pokušaja pronalaženja bolje inačice i analize njihovih performansi u odnosu na inačice od kojih su sastavljene.

7.1 MCGAPSO

Krenut ćemo s pokušajem pronalaženja inačice koja bi korištenjem komponenti inačica prilagođenih za učenje mreže ANFIS mogla biti uspješnija u rješavanju problema. Započet ćemo prisjećanjem o kojim je točno komponentama riječ. IPSO inačica ne uvodi specifične prilagodbe za promatrani problem te neće biti razmatrana u nastavku. SAPSO za svoj rad koristi čestice koje predstavljaju djelomična rješenja (parametri jednog ANFIS pravila) koja se međusobno kombiniraju za stvaranje cjelovitog rješenja (svi parametri mreže ANFIS), uz to koristi izmijenjen izraz za računanje nove brzine te operator susjedstva. S druge strane GAPSO svoju strategiju bazira na zasebnom evoluiranju skupine parametara u svakoj iteraciji i izbacivanjem najgore čestice koja se zamjenjuje novom, nastalom križanjem dvaju nasumično odabranih čestica. Sljedeći korak nam je odabir inačice na kojoj ćemo temeljiti naš prošireni algoritam.

Motivirani njihovim dobrim rezultatima na standardnom skupu problema, MCPSO i MRPSO predstavljaju dobre kandidate za traženu temeljnu inačicu. Obje inačice se pokušalo proširiti gore navedenim komponentama SAPSO i GAPSO inačica, osim operatora susjedstva za kojeg se pretpostavilo da ne bi imao veliki utjecaj s obzirom na činjenicu da ove inačice pokazuju bolje performanse s većim brojem manjih podrojeva od oko 5 čestica. Zbog malog broja čestica postepeno proširivanje susjedstva ne bi bilo od velike koristi. Pokazalo se da pokušaji dodavanja navedenih komponenti u MRPSO inačicu

nisu uspjeli ostvariti poboljšanja performansi već je samo uvedena dodatna kompleksnost, a time i porast vremena izvođenja algoritma, te je odbačena kao mogući kandidat.



Slika 16 - Performanse MCGAPSO algoritma

S druge strane korištenje izmijenjenih izraza za osvježavanje brzine čestica, zasebno osvježavanje skupina parametara te zamjena najgore čestice novom (dobivenu križanjem)

u MCPSO inačici, doprinijeli su osjetno boljim performansama na nekim problemima od inačica od kojih je algoritam sastavljen (pseudokod 7). Usporedbu MRPSO, inačica od kojih je ova inačica, nazovimo ju MCGAPSO, sastavljena i nje same možemo vidjeti na slici 16. Primjećujemo da na standardnom skupu problema nije ostvareno poboljšanje performansi što možemo objasniti činjenicom da odvojeno osvježavanje skupine parametara na ove probleme nema utjecaja, a ostale komponente same za sebe ne poboljšavaju osjetno rad algoritma na razmatranom skupu problema, dok se s druge strane MCGAPSO inačica je pokazala značajno bolje performanse na problemu učenja mreže ANFIS.

Pseudokod 7 - MCGAPSO

```
1. Inicijaliziraj čestice glavnog roja
2. Stvori podaničke rojeve i inicijaliziraj njihove čestice
3. Ponavljaj
4.   Izračunaj dobrote svih čestica
5.   Paralelno
6.     Pokreni osnovnu inačicu algoritma roja čestica s izmijenjenim
       izrazom računanja brzine (3) i odvojeno za svaku grupu
       parametara
7.   Za sve rojeve podanike
8.   Čekaj završetak svih rojeva
9.   Dohvati najbolju česticu iz svih podaničkih rojeva
10.  Evoluiraj glavni roj
11. Dok nije zadovoljen uvjet zaustavljanja
```

7.2 MCRPSO

Sljedeći pokušaj stvaranja nove inačice vođen je razmatranjem sljedećih činjenica dosadašnjih rezultata:

- MRPSO inačica se pokazala najboljom inačicom na svim problemima,
- MCPSO je za nekoliko problema iz standardnog skupa pokazao jako dobre performanse,

- izmijenjena MCPSO inačica pokazala je na nekim problemima bolje performanse od osnovne inačice te
- postoji određena sličnost između MRPSO i MCPSO inačica s obzirom na organizaciju rojeva i čestica.

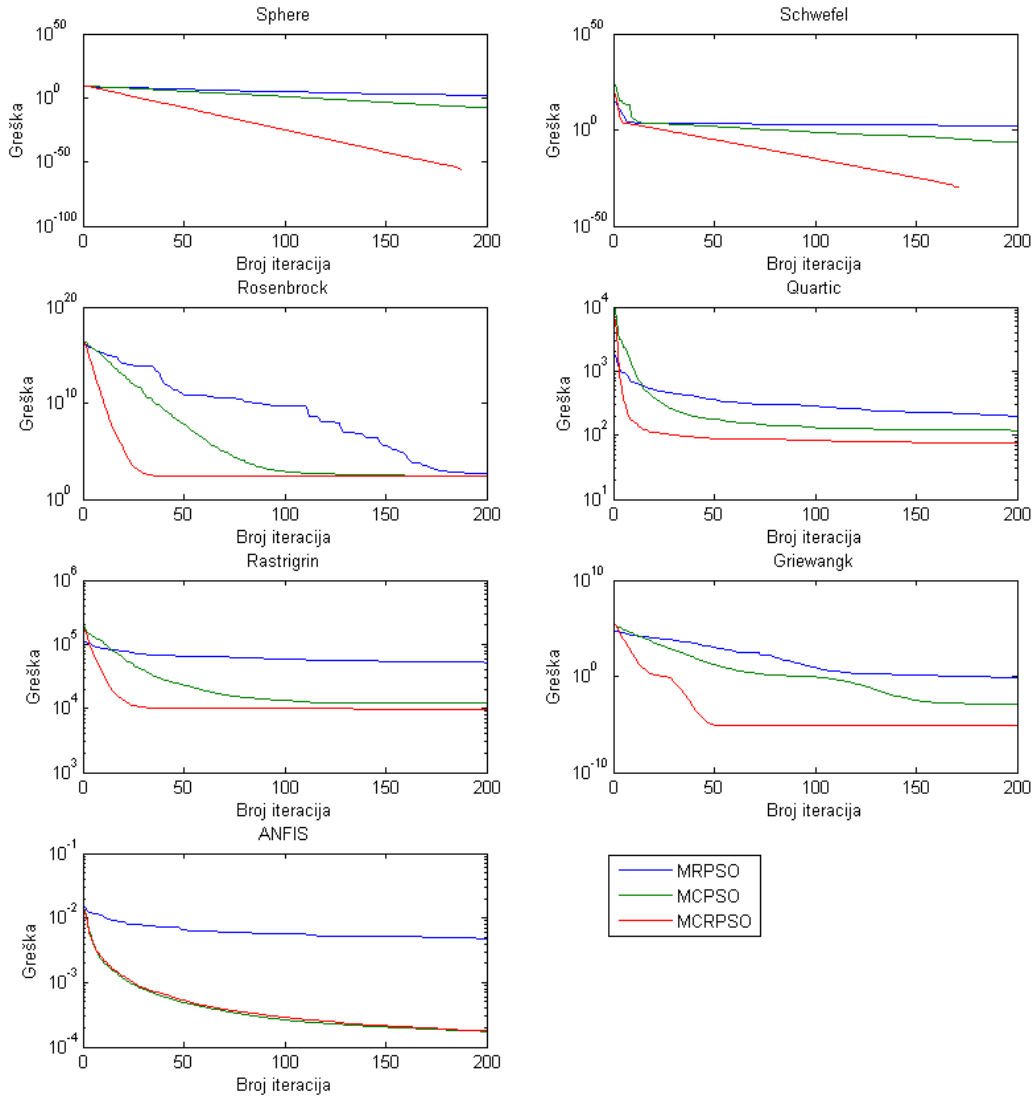
Na temelju ovih činjenica odlučeno je da se izgradi inačica koja će koristiti komponente spomenutih inačica. Kombinacija će biti provedena na način da se u potpunosti zadrži struktura i način funkcioniranja MRPSO inačice te nju nadogradi strategijom kojom združena inačica MCPSO algoritma koristi podaničke rojeve za evoluciju glavnog roja.

Drugim riječima, algoritam će na početku stvoriti zadani broj podrojeva povezanih u prstenastu strukturu, i jedan glavni roj. U svakom koraku će podrojevi neovisno jedan o drugom evoluirati uz odbojnu komponentu zbog koje će sve čestice podroja težiti udaljavanju od najbolje čestice prethodnog podroja u strukturi. Pri završetku evolucije svih podrojeva doći će do prijenosa najboljeg skupa čestica u sljedeći podroj u strukturi. Na kraju svakog koraka bit će odabrana najbolja čestica u svim rojevima koja predstavlja jednu od komponenti u izračunu novog vektora smjera za čestice glavnog roja (pseudokod 8).

Pseudokod 8 - MCRPSO

```
1. Inicijaliziraj glavni roj i njegove čestice
2. Inicijaliziraj podrojeve i njihove čestice
3. Organiziraj podrojeve u prstenastu strukturu
4. Ponavljaj
5.    Izračunaj dobrotu čestica svih podrojeva
6.    Sve parne podrojeve udalji od susjeda s nižim indeksom
7.    Pokreni algoritam roja čestica nad svim podrojevima
8.    Iz svakog podroja odaberi k najboljih čestica te njima zamijeni k
       najgorih čestica iz podroja s većim indeksom
9.    Dohvati najbolju česticu iz svih podaničkih rojeva
10.   Evoluiraj glavni roj po izrazu (11)
11. Dok nije zadovoljen uvjet zaustavljanja
```

Performanse ove inačice, koju nazivamo **MCRPSO** po inačicama od kojih je sastavljena, na svim problemima moguće je vidjeti na slici 17 gdje je prikazana njena usporedba s MRPSO i MCPSO inačicama. Iz priloženog vidimo da su se na dva problema inačice pokazale jednako dobrima, na funkciji Griewangk je MRPSO pronalazio značajno bolja rješenja, dok se na svim ostalim problemima MCRPSO pokazao boljim algoritmom. Valja primijetiti kako je MCRPSO za funkcije Sphere i Schwefel prvi algoritam koji je u manje od 200 iteracija uspio uvijek pronaći njihovu optimalnu vrijednost. To vidimo po prekidu grafa koji signalizira da je vrijednost funkcije 0.



Slika 17 - Performanse MCRPSO algoritma

S obzirom na to da su se obje inačice od kojih je MCRPSO inačica sačinjena pokazale jako sporim u odnosu na druge razmatrane inačice postojala je zabrinutost oko presporog izvođenja novog algoritma. Ipak, s obzirom na to da je MRPSO u relativno maloj mjeri nadograđen, ispostavilo se da rast vremena izvođenja nije bio prevelik, iako je do njega očekivano i došlo. Tako se za isti problem i broj iteracija MRPSO izvodio već spomenutih 298.9 sekundi, dok je izvođenje za MCRPSO poraslo na 323.9 sekunde.

Potaknuti dobrim rezultatima ove inačice pokušalo ju se nadograditi i komponentama iz drugih inačica, međutim osim povećanja kompleksnosti, performanse su u najboljem slučaju ostale približno iste, a najčešće su se pogoršale. Jedan od pristupa koji se činio prikladnim je bio pokušaj korištenja jedne od naprednijih inačica umjesto osnovne za evoluciju podrojeva. Međutim ni taj pristup, kao ni pokušaj korištenja različitih inačica za različite podrojeve nisu opravdali značajno povećanje kompleksnosti i vremena izvođenja te su odbačeni.

Razvijene inačice ostavljaju dosta prostora za daljnji razvoj, naročito u vidu optimiranja algoritama kako bi se smanjila vremena izvođenja. Uz to, bilo bi potrebno provesti detaljnu analizu parametara i određivanje njihovog kvalitativnog utjecaja na rad algoritama. S obzirom na činjenicu da rojevi neovisno pretražuju prostor stanja osnovnom inačicom algoritma roja čestica, bilo bi poželjno uvesti promijene za koje se pokazalo da poboljšavaju njene performanse neovisno o problemu (Montes, 2009).

8. Zaključak

Razmotrili smo optimizacijske probleme i pristup njihovom rješavanju algoritmom roja čestica i njegovim inačicama. Sve izvedene inačice opravdale su dodatnu kompleksnost boljim performansama od osnovne inačice na svim korištenim primjerima. Vidjeli smo kako odabirom parametara možemo značajno utjecati na rad algoritma, specijalizirati ga za rješavanje jednog konkretnog problema ili osigurati određenu razinu performansi na širem skupu različitih problema. Nadalje, pokazali smo, na primjerima razvijenih inačica MCGAPSO i MCRPSO, da je moguće poboljšati performanse određene inačice uvođenjem strategija prilagođenih problemu kojeg nastojimo riješiti ili kombiniranjem s nekom drugom općom strategijom.

U konačnici, možemo reći da je algoritam roja čestica jedna moćna paradigma procesiranja informacija koju ne nalazimo samo u kretanju jata ptica, već svuda u prirodi, pa i u ljudskom ponašanju. U prilog tome govori rad (Harmon-Jones, 1999) u kojem se za prvu hipotezu navodi kako su dva glavna izvora kognicije osobno iskustvo i oponašanje, koji upravo čine srž algoritma roja čestica, a ujedno su i osnovni elementi prisutni u raznim drugim socijalnim teorijama. Pomoću ovih jednostavnih pravila omogućeno je da jedinke međusobno utječu jedna na drugu, što dovodi do socijalnog učenja i kao krajnju posljedicu pojavu inteligencije roja. Upravo zahvaljujući toj izranjajućoj inteligenciji, koja nije prisutna u većini drugih populacijskih algoritama, algoritam roja čestica je u mogućnosti uspješno rješavati teške matematičke problem.

Vidjeli smo i da je ova osnovna paradigma lako proširiva te da se jednostavno i dobro uklapa s drugim paradigmama, zbog čega je i nastao velik broj naprednijih inačica. Ono što se tim inačicama nastojalo izmijeniti su načini komunikacije među česticama, odnosno strategije razmijene informacija. Kao što smo imali prilike vidjeli u ovom radu, i za ovaj aspekt se nastoje oponašati razni prirodni procesi: razne vrste selekcija, razmjena informacija u obliku križanja, razni oblici natjecanja i suradnji, i drugi. Svi ti procesi definiraju način kolanja informacija koje jedinke koriste, uz svoje iskustvo, u pretrazi prostora rješenja.

Daljnja istraživanja trebala bi omogućiti bolji ili barem olakšati odabir strategije razmijene informacija prilagođene konkretnom problemu koji se nastoji riješiti. Iako bi

glavni cilj trebao ostati na tragu osnovnih postavki umjetne inteligencije, nastojati razviti inteligentni sustav prilagodljiv širokom skupu problema.

_____

Potpis

9. Literatura

Abraham, A.. „Adaptation of fuzzy inference system using neural learning.“ In Fuzzy Systems Engineering, pp. 53-83. Springer Berlin Heidelberg, 2005.

Aware, M. V., Kothari, A. G., Choube, S. O. „Application of adaptive neuro-fuzzy controller (ANFIS) for voltage source inverter fed induction motor drive.“ In Power Electronics and Motion Control Conference, 2000. Proceedings. IPEMC 2000. The Third International, vol. 2, pp. 935-939. IEEE, 2000.

Banks, A., Vincent, J., Anyakoha, C. „A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications.“ Natural Computing 7, no. 1 (2008): 109-124.

Bishop, J.M., „Stochastic Searching Networks“, Proc. 1st IEE Conf. on Artificial Neural Networks, pp 329–331, 1989.

Blackwell, T., Branke, I., „Multiswarm optimization in dynamic environments,” in Applications of Evolutionary Computing. ser. LNCS, vol. 3005, pp. 489–500, 2004.

Cabalar, A. F., Cevik, A., Gokceoglu, C. „Some applications of adaptive neuro-fuzzy inference system (ANFIS) in geotechnical engineering.“ Computers and Geotechnics 40 (2012): 14-33.

Chen, H., Zhu, Y. „Optimization based on symbiotic multi-species coevolution.“ Applied Mathematics and Computation 205, no. 1 (2008): 47-60.

Chen, K., Li, T., Cao T., „Tribe-PSO: A novel global optimization algorithm and its application in molecular docking“, in Chemometrics and Intelligent Laboratory Systems, Vol. 82, Issues 1-2, pp. 248-259, 2006.

Chu , S. C., Chen, Y. T. Ho, J. H., „Timetable scheduling using particle swarm optimization“, first international conference on innovation computing, Information and control, Vol. 3, pp.324-327, 2006.

Chuang, C.-L., Hung, K., Chen, C.-M., Shieh, G. „Uncovering transcriptional interactions via an adaptive fuzzy logic approach.“ BMC bioinformatics 10, no. 1 (2009): 400.

Clerc, M., „Particle Swarm Optimization“, 1st edition, London: ISTE, 2006.

Čupić, M., „Prirodom inspirirani optimizacijski algoritmi“, skripta iz kolegija Umjetna inteligencija, verzija 1.0.7, Zagreb: Fakultet elektrotehnike i računarstva, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave, 2010.

Dalbelo Bašić, B., Čupić, M., Golub, M. „Neizrazito, evolucijsko i neuroračunarstvo“, skripta iz kolegija Neizrazito, evolucijsko i neuroračunarstvo, Zagreb: Fakultet elektrotehnike i računarstva, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave, 2012.

Dorigo, M., „Optimization, Learning and Natural Algorithms“, PhD thesis, Politecnico di Milano, Italy, 1992.

Du, W., Bin Li. „Multi-strategy ensemble particle swarm optimization for dynamic optimization.“ Information sciences 178, no. 15 (2008): 3096-3109.

Eberhart, R. C., Kennedy, J., „A new optimizer using particle swarm theory“, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39-43, 1995.

Eberhart, R. C., Shi, Y. „Comparing inertia weights and constriction factors in particle swarm optimization.“ In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, pp. 84-88. IEEE, 2000.

Farmer, J. D., Packard, N. H., Perelson, A., „The Immune System, Adaptation, and Machine Learning", Physica D 22, pp. 187-204, 1986.

Frattale Mascioli, F. M., Varazi, G. M., Martinelli, G. „Constructive algorithm for neuro-fuzzy networks.“ In Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on, vol. 1, pp. 459-464. IEEE, 1997.

Geem, Z. W., Kim, J. H., Loganathan, G. V., „A New Heuristic Optimization Algorithm: Harmony Search“, Simulation, SIMULATION, Vol. 76, No. 2, pp. 60-68, 2001.

Ghomsheh, V. Seydi, M. Aliyari Shoorehdeli, M. Teshnehlab. „Training ANFIS structure with modified PSO algorithm.“ In Control & Automation, 2007. MED'07. Mediterranean Conference on, pp. 1-6. IEEE, 2007.

Harmon-Jones, E., Mills, J. „Cognitive dissonance: Progress on a pivotal theory in social psychology.“ In Scientific Conferences Program, 1997, U Texas, Arlington, TX, US; This volume is based on papers presented at a 2-day conference at the University of Texas at Arlington, winter 1997. American Psychological Association, 1999.

Holland, J. H., „Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence“, Ann Arbor, MI: University of Michigan Press, 1975.

Jang, J-SR. „ANFIS: adaptive-network-based fuzzy inference system.“ Systems, Man and Cybernetics, IEEE Transactions on 23, no. 3 (1993), 665-685.

Janson, S., Middendorf, M., „A hierarchical particle swarm optimizer for dynamic optimization problems“, in Application of Evolutionary Computing, ser. Lecture Notes in Computer Science, Vol. 3005, pp. 513-524, 2004.

Kang, Q., Wang, L., Wu, Q. D. „A novel ecological particle swarm optimization algorithm and its population dynamics analysis.“ Applied Mathematics and Computation 205, no. 1 (2008): 61-72.

Kennedy, J., Eberhart, R. C. Swarm intelligence. Prvo izdanje. San Francisco, California: Morgan Kaufmann, 2001.

Li, C., Chiang, T. W. „Intelligent financial time series forecasting: A complex neuro-fuzzy approach with multi-swarm intelligence.“ International Journal of Applied Mathematics and Computer Science 22, no. 4 (2012): 787-800.

Liang, J. J., Suganthan, P. N. „Dynamic multi-swarm particle swarm optimizer.“ In Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE, pp. 124-129. IEEE, 2005.

Lin, C.-J., Liu, Y.-C. „Image backlight compensation using neuro-fuzzy networks with immune particle swarm optimization.“ Expert Systems with Applications 36, no. 3 (2009): 5212-5220.

Lin, C.-J., Chen, C.-H., Lee, C.-Y. „Efficient Immune-Based Particle Swarm Optimization Learning for Neuro-Fuzzy Networks Design.“ J. Inf. Sci. Eng. 24, no. 5 (2008), 1505-1520.

Lin, C.-J., Peng, C.-C., Lee, C.-J. „Identification and prediction using neuro-fuzzy networks with

symbiotic adaptive particle swarm optimization.“ Informatica: An International Journal of Computing and Informatics 35, no. 1 (2011): 113-122.

Lin, C.-J., Xu, Y.-J. „A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications.“ *Fuzzy Sets and Systems* 157, no. 8 (2006): 1036-1056.

Lin, C.-J. „An efficient immune-based symbiotic particle swarm optimization learning algorithm for TSK-type neuro-fuzzy networks design.“ Fuzzy Sets and Systems 159, no. 21 (2008): 2890-2909.

Mendes, R., Cortez, P., Rocha, M., Neves, J. „Particle swarms for feedforward neural network training.“ In Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on, vol. 2, pp. 1895-1899. IEEE, 2002.

Mohais, A. S., Mendes, R., Ward, C., Posthoff, C. „Neighborhood re-structuring in particle swarm optimization.“ In AI 2005: Advances in Artificial Intelligence, pp. 776-785. Springer Berlin Heidelberg, 2005.

Montes de Oca, M. A., Stützle, T., Birattari, M., Dorigo, M., „Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm“, IEEE Transactions on Evolutionary Computation, vol. 13, No. 5, pp. 1120-1132, 2009.

Niu, B., Li, L. „An Improved MCPSO with Center Communication.“ InComputational Intelligence and Security, 2008. CIS'08. International Conference on, vol. 2, pp. 57-61. IEEE, 2008.

Niu, B., Zhu, Y., He, X., Wu, H. „MCPSO: A multi-swarm cooperative particle swarm optimizer.“ Applied Mathematics and Computation185, no. 2 (2007): 1050-1062.

Poli, R. "Analysis of the publications on the applications of particle swarm optimisation." *Journal of Artificial Evolution and Applications* 2008 (2008): 3.

Rohler, A. B., Stephen Chen „Multi-swarm hybrid for multi-modal optimization.“ In Evolutionary Computation (CEC), 2012 IEEE Congress on, pp. 1-8. IEEE, 2012.

Shi, Y., Eberhart, R. C. „Fuzzy adaptive particle swarm optimization.“ In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1, pp. 101-106. IEEE, 2001.

Takagi, T., Sugeno, M. „Fuzzy identification of systems and its applications to modeling and

control.“ Systems, Man and Cybernetics, IEEE Transactions on 1 (1985): 116-132.

Teodorović, D., Dell'Orco, M., „Bee colony optimization—a cooperative learning approach to complex transportation problems“, in Proceedings of the 10th EWGT Meeting, pp. 13–16, 2005.

Tzafestas, S. G. „Computational Intelligence in Systems and Control Design and Applications“. Vol. 22. Prvo izdanje. Springer, 2002.

Vanneschi, L., Codecasa, D., Mauri, G. „An empirical comparison of parallel and distributed particle swarm optimization methods.“ In Proceedings of the 12th annual conference on Genetic and evolutionary computation, Portland, Oregon (2010) pp. 15-22.

Vesterstrom, J., Thomsen, R. „A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems.“ In *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, pp. 1980-1987. IEEE, 2004.

Wolpert, D. H., Macready, W. G. „No free lunch theorems for optimization.“ *Evolutionary Computation, IEEE Transactions on* 1, no. 1 (1997): 67-82.

Zhan, Zhi-Hui, Jun Zhang, Yun Li, HS-H. Chung. „Adaptive particle swarm optimization.“ Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39, no. 6 (2009): 1362-1381.

Zhao, Shi-Zheng, Jing J. Liang, Ponnuthurai N. Suganthan, Mehmet Fatih Tasgetiren. „Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization.“ In Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, pp. 3845-3852. IEEE, 2008.

9.1 Izvori s WWW-a

[1] Članak „Evolutionary algorithm“, stvoren dana 2. ožujka 2003., Wikipedia, http://en.wikipedia.org/wiki/Evolutionary_algorithms, pristupljeno dana 21. prosinca 2013.

[2] Članak „Swarm intelligence“, stvoren dana 29. lipnja 2004., Wikipedia, http://en.wikipedia.org/wiki/Swarm_intelligence, pristupljeno dana 21. prosinca 2013.

[3] Članak „Particle swarm optimization“, stvoren dana 8. listopada 2003., Wikipedia, http://en.wikipedia.org/wiki/Particle_swarm_optimization, pristupljeno dana 23. prosinca 2013.

[4] Cervantes A., „PSO: Neighborhood topologies“, stvoreno 2005., TRACER, http://tracer.uc3m.es/tws/pso/neighborhood.html, pristupljeno dana 27. prosinca 2013.

[5] Dorigo M., et al., „Particle swarm optimization“, stvoreno 2008., Scholarpedia, http://www.scholarpedia.org/article/Particle_swarm_optimization, pristupljeno dana 26.prosinca 2013.

10. Sažetak

Algoritam roja čestica spada u skupinu prirodom inspiriranih algoritama za globalnu optimizaciju. U ovom radu smo proveli detaljnu analizu osnovne i raznih naprednijih inačica koje su nastale uvođenjem raznih strategija u rad algoritma, kombiniranjem različitih metaheuristika ili adaptacijom inačice za rješavanje specifičnog problema. Usporedbom razmatranih inačica na skupu problema različitih karakteristika uspješno su određene komponente koje su zaslužne za dobre performanse pojedinih inačica te su one iskorištene za izgradnju novih, poboljšanih, inačica. Novonastale inačice su omogućile ili postizanje boljih rezultata na određenim, za ostale inačice, teškim problemima uz pad performansi na širem skupu problema ili općenito poboljšanje performansi na svim razmatranim problemima.

Ključne riječi: algoritam roja čestica, PSO, inteligencija roja, prirodom inspirirani algoritmi, evolucijsko računanje, metaheuristika, optimizacijski algoritmi, problem optimizacije

11. Summary

The particle swarm optimization algorithm is a nature-inspired algorithm for global optimization. In this work we have presented a detailed analysis of the basic, and some more advanced variants created by the introduction of various strategies in the algorithm, the combination of different metaheuristics or the adaptation of a variant for solving a specific problem. By comparing the performances of the considered variants we were able to successfully isolate the components responsible for a variants better performance, and combine them to create new, improved, variants. The newly-created variants were either more performant on a task on which the other variants were struggling, or showed a general improvement in the performance over the whole set of considered problems.

Keywords: particle swarm optimization, PSO, swarm intelligence, nature-inspired algorithms, evolutionary computing, metaheuristics, optimization algorithms, optimization problem