

Optimizacija rojem čestica

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Nevena Soldat, Milena Kurtić, Tijana Živković, Ana Miloradović

nevenasoldat@gmail.com, mimikurtic67@gmail.com,
tijanazivkovic6@gmail.com, ana.miloradovic7@gmail.com

25. mart 2020.

Sažetak

Kennedy i Eberhart (2001):

“... gledamo u paradigmu koja je u svom začeću, puna potencijala i novih ideja i novih perspektiva... Istraživači u mnogim zemljama eksperimentišu sa rojevima čestica... Mnoga pitanja koja su postavljena još uvek nisu dobila dobar odgovor [4]”

U ovom radu opisujemo osnovni algoritam za optimizaciju rojem čestica, kao i neke od postojećih varijacija. Objašnjene su socijalne strukture na kojima se on zasniva. Navedeni su primeri korišćenja, kao i koje su moguće primene.

Sadržaj

1	Uvod	2
2	Algoritam za optimizaciju rojem čestica	2
2.1	Originalni PSO	3
2.2	Komponente brzine PSO algoritma	4
2.3	Algoritmi gbest i lbest PSO	5
2.3.1	Poređenje gbest i lbest PSO algoritma	5
3	Osnovne varijacije	5
3.1	Smanjenje brzina	5
3.2	Inercijalna težina	7
3.3	Koeficijent suženja	7
3.4	Modeli brzina	8
4	Primene	8
4.1	Primer rešavanja problema rekonfiguracije i planiranja distributivne mreže	9
5	Topologije uticaja	11
	Literatura	11

1 Uvod

Inteligencija rojeva predstavlja jednu od pet glavnih paradigmi Računarske Inteligencije (Computation Intelligence - CI). Jedinke u okviru grupe (roja) dele prikupljene informacije u zajedničkom cilju da reše neki problem, koje se propagiraju kroz celu grupu tako da se problem rešava mnogo efikasnije nego što bi to mogla pojedinačna jedinka.

Prvi i dosta značajan doprinos u polju inteligencije rojeva imao je južnoafrički pesnik Eugene N Marais koji je proučavao socijalno ponašanje kako majmuna, tako i mrava. Posle njega, ranih 1990-ih godina, Marco Dorigo modeluje ponašanje kolonija mrava. Zatim, 1995, Eberhart i Kennedy razvijaju algoritam optimizacije rojem čestica, na osnovu posmatranog jata ptica.

Optimizacija rojem čestica (eng. Particle Swarm Optimization - PSO) je stohastička optimizaciona tehnika zasnovana na veoma inteligentnom kolektivnom ponašanju nekih organizama kao što su insekti, ptice i ribe. Algoritam optimizacije rojem čestica otkriven je sasvim slučajno od strane Eberharta i Kenedija, pri pokušaju da se na računaru simulira kretanje jata ptica. Prvobitna namera bila je da se grafički prikaže nepredvidiva koreografija jata ptica, sa ciljem da se otkriju obrasci koji omogućavaju pticama da lete sinhronizovano, i da zadrže optimalnu formaciju pri naglim promenama pravaca. Sada je cilj kreiranje jednostavnog i efikasnog optimizacionog algoritma. Od kada je prvi put predstavljen 1995. doživeo je niz poboljšanja i nastale su brojne varijacije ovog algoritma.

2 Algoritam za optimizaciju rojem čestica

Kako bismo lakše razumeli algoritam možemo zamisliti roj pčela koje lete preko polja sa cvećem [1]. Roj ima urođenu želju da pronađe poziciju gde je cveće najgušće raspoređeno. Takođe, pčele nemaju nikakvo znanje o polju na kom se nalaze. Tako počinju svoju pretragu u različitim smerovima. Svaka pčela pamti mesta na kojima je bila i na kojim je bilo najviše cveća, i tu informaciju može preneti komunikacijom sa ostatkom roja. Kako vreme prolazi, pčele biraju da li će se vratiti na svoje prethodno pronađene najbolje lokacije ili će ići ka lokacijama koje su dobile od ostalih pčela. One koje oklevaju će ići u oba pravca i nalaziće se negde između ciljanih lokacija, u zavisnosti od toga da li će socijalan uticaj biti dominantan ili ne. Povremeno, pčela može da preleti preko dela polja u kom se nalazi više cveća od do sad otkrivenih lokacija. Tada će se čitav roj povlačiti ka novootkrivenoj lokaciji.

Na slici 1, isprekidane linije predstavljaju zamišljene putanje pčela, a strelice prikazuju dve komponente brzine (lokalno najbolju poziciju i globalno najbolju poziciju). Pčela u gornjem delu slike je pronašla globalno najbolju poziciju, dok je pčela sa leve strane pronašla lokalno najbolju poziciju. Pčela u donjem delu slike prikazuje da iako nije pronašla lokalno najbolju poziciju, ide ka globalno najboljoj poziciji.

Na ovaj način pčele pretražuju polje tako što menjaju brzine i pravac kretanja u zavisnosti od toga da li je imala uspeha da pronađe cveće u odnosu na čitav roj. Takođe pčele znaju da izbegavaju mesta koja nisu imala puno cveća. Na kraju će pretražiti celo polje, i nalaziće se iznad mesta na kojem je najveća gustina cveća.



Slika 1: Prikaz PSO algoritma na kojem pčele traže cveće

2.1 Originalni PSO

Algoritam za optimizaciju rojem čestica sadrži jedinke (čestice) koje se se kreću kroz višedimenzioni prostor pretrage, a pozicije jedinki se menjaju u skladu sa sopstvenim iskustvom, kao i sa iskustvom susednih jedinki. Svaka od tih čestica predstavlja jedno moguće rešenje. Neka je $x_i(t)$ pozicija čestice i u prostoru pretrage u trenutku t . Pozicija čestice se menja dodavanjem brzine, $v_i(t)$ na trenutnu poziciju.

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Njihovo kretanje se usmerava imajući u vidu njihovu trenutnu poziciju, njihovu do sada najbolju poziciju, kao i do sada najbolju poziciju čitavog roja. Kognitivna komponenta algoritma predstavlja tendenciju vraćanja u lično najbolje rešenje, dok socijalna komponenta predstavlja tendenciju ka globalno najboljem rešenju. Brzina se računa kao:

$$v_i(t+1) = v_i(t) + c_1 r_1 (p_i(t) - x_i(t)) + c_2 r_2 (p_g(t) - x_i(t))$$

gde $p_i(t)$ predstavlja najbolju do sada poziciju čestice i u trenutku t , dok je $p_g(t)$ globalno najbolje rešenje (pozicija) do trenutka t . Parametri r_1 i r_2 su nasumične vrednosti izabrane iz uniformne raspodele na intervalu $[0,1]$, dok su parametri c_1 i c_2 konstante koje predstavljaju pozitivna ubrzanja čija je uloga da skaliraju značaj kognitivne, odnosno socijalne komponente brzine. Pseudokod originalnog PSO algoritma:

Algoritam osnovni PSO:

Kreiraj i inicijalizuj n_s - dimenzioni roj;

ponavljaj

za svaku česticu $i = 1, \dots, n_s$ **uradi**

// postavi lokalno najbolju poziciju

ako $f(x_i) < f(p_i)$ **onda**

$p_i = x_i$;

kraj

//postavi globalno najbolju poziciju

ako $f(p_i) < f(p_g)$ **onda**

$p_g = p_i$;

kraj

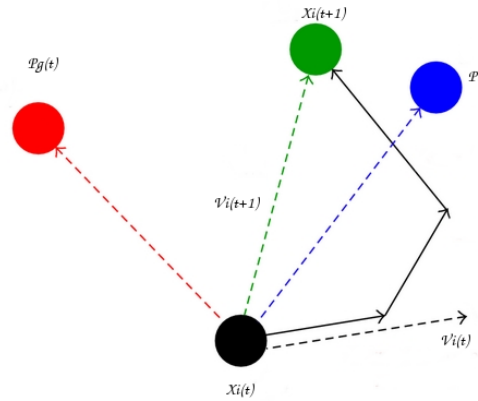
kraj
za svaku česticu $i = 1, \dots, n_s$ **uradi**
 ažuriraj brzinu;
 ažuriraj poziciju;
kraj
dok nije ispunjen zahtev za zaustavljanje;

Funkcija $f : \mathbb{R}^{n_s} \rightarrow \mathbb{R}$ je fitnes funkcija. Fitnes funkcija računa koliko je dobijeno rešenje blisko optimalnom, odnosno meri kvalitet rešenja.

2.2 Komponente brzine PSO algoritma

Brzina i -te čestice ima tri komponente:

- **Prethodna vrednost brzine**, $\mathbf{v}_i(t)$, koja čuva prethodni pravac kretanja čestice. Može se reći da je ona moment koji sprečava česticu da drastično promeni pravac.
- **Kognitivna komponenta**, $c_1 \mathbf{r}_1(\mathbf{p}_i - \mathbf{x}_i)$, koja meri rezultat čestice i u odnosu na prethodne. Može se opisati kao pamćenje najbolje pozicije u kojoj se čestica do sada našla. Efekat ove komponente je tendencija čestica da se vrate na pronadjene najbolje pozicije. Kennedy i Eberhart su nazivali kognitivnu komponentu “nostalgija” čestice.
- **Socijalna komponenta**, $c_2 \mathbf{r}_2(\mathbf{p}_g - \mathbf{x}_i)$, koja meri rezultat čestice i u odnosu na sve susedne čestice. Efekat ove komponente je tendencija čestice da se kreće ka najboljoj poziciji pronađenoj od strane susednih čestica.



Slika 2: Prikaz promene brzine

U svakoj generaciji, iterativni proces čestice izgleda kao na slici 2. Analizirajući ažuriranje brzine, vidimo da je prvi deo uticaj prethodne brzine čestice. To znači da čestica veruje da je na dobrom putu i kreće se inercijalno. Drugi deo zavisi od trenutne pozicije čestice i njene najbolje pozicije do sada, odnosno od kognitivne komponente brzine. Treći deo se oslanja na udaljenost trenutne pozicije čestice i globalno najbolje pozicije u roju, odnosno na socijalnu komponentu brzine [6].

2.3 Algoritmi gbest i lbest PSO

U slučaju globalno najboljeg PSO algoritma (*gbest*), susedi za svaku česticu su zapravo čitav roj. On implementira topologiju zvezde. U topologiji zvezde socijalna komponenta brzine čestice predstavlja informaciju dobijenu od svih čestica u roju.

Lokalno najbolji PSO (*lbest*) koristi topologiju prstena, gde svaka čestica ima određeni broj suseda. Socijalna komponenta predstavlja informaciju koju razmenjuju susedi čestice. Doprinos jedinke brzini je proporcionalna razdaljini između čestice i najbolje pozicije koju su pronašli susedi.

Važno je napomenuti da kod osnovnog PSO algoritma čestice nisu povezane. Razvijene su mnoge strategije na osnovu kojih se bira susedstvo, kao što je razdaljina u prostoru. Međutim, preferirano rešenje je odabir na osnovu indeksa, i za to postoje dva glavna razloga:

1. Za pristupe koji podrazumevaju računanje udaljenosti između čestica, potrebno je naći Euklidsko rastojanje svih parova čestica, a to ima veliku složenost $((n_s)^2)$.
2. Lakše je prosleđivanje informacije o dobrim rešenjima svim česticama, nezavisno od njihove lokacije u prostoru pretrage.

Obe strategije zasnovane su na nekom socijalnom ponašanju. Kako bismo slikovito prikazali zašto nije dobro birati susedstvo na osnovu udaljenosti? Zamislite da koordinate u prostoru predstavljaju mentalne sposobnosti osobe. U tom slučaju, dve veoma slične osobe možda se nikada ne bi srele, a možemo imati dve veoma različite osobe koje će uvek biti susedne [3].

2.3.1 Poređenje gbest i lbest PSO algoritma

Dve verzije PSO algoritma koje smo opisali su slične u smislu da obe teže ka globalno najboljem rešenju. Međutim postoje dve osnovne razlike [2]:

- Zbog bolje povezanosti čestica u gbest algoritmu, on brže konvergira ka rešenju. Cena brže konvergencije je manja raznovrsnost.
- Kao posledica veće raznovrsnosti (što podrazumeva da je veći deo prostora pretrage pokriven), lbest PSO ima manje šanse da ostane zarobljen u lokalnom optimumu. U većini slučajeva, topologija prstena kao kod lbest PSO algoritma poboljšava rezultate.

3 Osnovne varijacije

Postoji nekoliko modifikacija osnovnog PSO algoritma [5], a one su razvijane da bi poboljšale brzinu konvergencije i kvalitet rešenja koji ovaj algoritam nalazi.

3.1 Smanjenje brzina

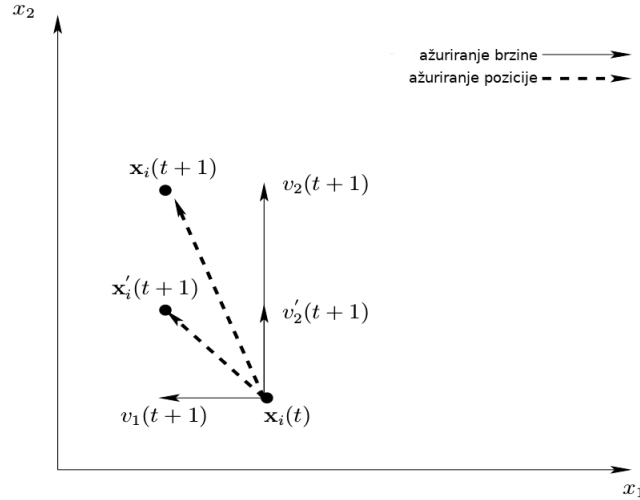
Za efikasnost i tačnost algoritma optimizacije važno je napraviti balans između eksploracije i eksploatacije. *Eksploracija* se odnosi na sposobnost algoritma pretrage da istražuje različite regione prostora pretrage da bi se pronašao dobar optimum, dok je *eksploatacija* sposobnost da se pretraga koncentriše oko regije koje garantuje nalaženje rešenja da bi se poboljšao kandidat rešenja. Ovo su kontradiktorni ciljevi koje treba dovesti u balans radi dobijanja dobrog optimizacionog algoritma.

Ažuriranje brzine u gorenavedenoj jednačini doprinosi veličini koraka čestice. U ranijim, osnovnim PSO algoritmima je primećeno da brzina brzo dostiže velike vrednosti, specijalno kod čestica koje su daleko od najboljih u okruženju i najboljih sopstvenih pozicija. Čestim ažuriranjem položaja jedinke polako napuštaju granice prostora pretrage, tj. divergiraju. Zato se brzine smanjuju kako bi čestice ostale u okviru granica. Ako brzina prelazi maksimalnu vrednost onda joj se dodeljuje baš ta vrednost, inače se na standardni način ažurira.

$$v_i(t+1) = \begin{cases} v'_i(t+1), & v'_i(t+1) < V_{max} \\ V_{max}, & v'_i(t+1) \geq V_{max} \end{cases} \quad (1)$$

Velike vrednosti maksimalne brzine olakšavaju istraživanje na globalnom nivou, dok manje vrednosti podstiču lokalnu eksploataciju. I premale i prevelike vrednosti maksimalne brzine imaju svoje mane. Dok veoma male vrednosti mogu da povećaju broj vremenskih koraka do nalaženja optimalne vrednosti, tako veoma velike vrednosti brzina mogu dovesti da se sasvim propusti dobar region, ali se uz to čestice ipak brže kreću. Tako osim problema balansa eksploracije i eksploatacije treba pronaći i dobar balans za svaki V_{max} , da ne bude ni premala, a ni prevelika vrednost.

Loša strana promene brzine leži u tome što se menja i pravac kretanja čestice, što omogućava bolju eksploraciju, ali isto tako može da dovede do toga da optimum uopšte ne bude pronađen.



Slika 3: Efekat smanjenja brzine

Postoji još i problem kada su sve vrednosti jednake V_{max} , a ako se ne preduzme ništa povodom toga, čestice se zaglavljaju unutar ograničene oblasti $[x_i(t) - V_{max}, x_i(t) + V_{max}]$. Ovaj problem se može rešiti uvođenjem inercijske težine ili smanjivanjem V_{max} vrednosti vremenom, tako što pretraga počine s velikim vrednostima i tako se pretražuje prostor, a zatim se postepeno smanjuje. Tako se ograničava globalna pretraga i osvrće se na lokalnu u već kasnim fazama pretraživanja. V_{max} se može menjati kada nema napretka u globalno najboljoj poziciji u toku određenog broja uzastopnih iteracija, ili se može eksponencijalno smanjivati.

3.2 Inercijalna težina

Ovaj pristup balansa između eksploracije i eksploatacije nastao je da bi se eliminisalo korišćenje smanjenje brzina. Težina w kontroliše koliko će prethodni pravac leta uticati na novu brzinu. Sada jednačina izgleda ovako:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)]$$

Dolazi do promene jednačine ažuriranja brzina kod *gbest* i *lbest* PSO algoritama. Kod ovog pristupa je vrednost w izuzetno važna za ostvarivanje konvergencije, kao i za prethodno pomenuti balans između eksploracije i eksploatacije. Razlikuju se slučajevi kada je $w=1$, tada se brzine povećavaju tokom vremena prema V_{max} i roj divergira, dok za $w < 0$ brzine opadaju sve dok ne dostignu nulu i time se algoritam zaustavlja, a za $0 < w < 1$ čestice usporavaju, pa konvergencija zavisi od vrednosti c_1 i c_2 . Tako velike vrednosti w olakšavaju eksploraciju, dok male vrednosti podstiču lokalnu eksploataciju. Kao i kod V_{max} , i ovaj pristup zavisi od samog problema. U početku se koristila konstantna vrednost w , za svaku česticu u svakoj dimenziji, a kasnije se započelo sa dinamičkim promenama vrednosti, počinjavši od velike inercijske vrednosti koja se vremenom smanjivala. Time se u početnim koracima dozvoljava da čestice istražuju, a zatim počinju da favorizuju određene delove područja kako vreme odmiče. Vrednost w se mora birati zajedno sa vrednostima c_1 i c_2 (konstante ubrzanja).

$$w > \frac{1}{2}(c_1 + c_2) - 1$$

Ovo garantuje da će putanje čestica voditi ka konvergenciji, a ukoliko ovaj uslov nije zadovoljen, može doći do cikličnog ponašanja, ili pak divergencije.

3.3 Koeficijent suženja

Clerc je razvio pristup sličan inerciji težina, gde su brzine ograničene konstantom χ koji se naziva koeficijentom suženja. Jednačina sada izgleda ovako:

$$v_{ij}(t+1) = \chi[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) + \phi_2(\hat{y}_{ij}(t) - x_{ij}(t))]$$

gde je

$$\chi = \frac{2k}{|2 - \phi - \sqrt{\phi(\phi - 4)}|},$$

sa $\phi = \phi_1 + \phi_2$, $\phi_1 = c_1r_1$ i $\phi_2 = c_2r_2$. Jednačina se primenjuje pod ograničenjima gde je $\phi \geq 4$ i $k \in [0, 1]$ Ovaj pristup napravljen je kao prirodan, dinamički način da se osigura konvergencija ka stabilnoj tački, bez potrebe za smanjivanjem brzine, roj će konvergirati zbog gorenavedenih ograničenja. Parametar k u jednačini kontroliše sposobnost eksploracije i eksploatacije, za 0 se postiže brza konvergencija uz lokalnu eksploataciju, dok za 1 sporo konvergira, sa visokim stepenom eksploracije. k se obično postavlja na konstantnu vrednost.

Razlika između pristupa smanjenja brzina i koeficijenta suženja je ta da smanjenje brzina nije neophodno za model suženja (Eberhart i Shi su pokazali da ako se koriste zajedno mogu da dovedu do brže konvergencije). Model suženja garantuje konvergenciju pod datim ograničenjima i kod njega svaka promena smera čestica mora se izvršiti preko konstanti ϕ_1 i ϕ_2 .

3.4 Modeli brzina

Modeli se razlikuju u komponentama koje su uključene u jednačinu brzine i u tome kako se utvrđuju najbolji položaji. Neki od njih su:

- **Cognition-Only**, kognitivni model isključuje socijalnu komponentu iz jednačine, pa jednačina izgleda ovako:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)(y_{ij}(t) - x_{ij}(t)).$$

Ovaj model se može uporediti sa nostalgijom i ilustruje stohastičku tendenciju da se čestice vraćaju u svoju najbolju poziciju. Ovaj model je sporiji, potrebno je više iteracija da bi se dostiglo dobro rešenje, a, takođe, ne uspeva kada je malo smanjenje brzina i koeficijent ubrzanja.

- **Social-Only**, socijalni model isključuje kognitivnu komponentu iz jednačine: $v_{ij}(t+1) = v_{ij}(t) + c_2 r_{2j}(t)(\hat{y}_j(t) - x_{ij}(t))$ u *gbest* PSO algoritmu. Kod ovog modela čestice nemaju tendenciju da se vraćaju na prethodno najbolje pozicije, već sve čestice idu ka najboljem položaju svog okruženja. Ovaj model se pokazao bržim od originalnog i kognitivnog modela.
- **Selfless**, kod ovog modela koji je sličan socijalnom, najbolje rešenje iz okruženja se bira isključivo iz suseda čestica. On se pokazao kao brži od socijalnog modela u čak nekoliko problema.

4 Primene

Postoje različiti načini primene PSO algoritma. Neki su opštiji, dok su ostali specifičniji. Pod opštim se misli na pristup koji obuhvata širok opseg specifičnih primena. A specifične primene podrazumevaju primenu na jasno definisan problem. Neke od primena su sledeće:

- U dizajnu antena – kontrola i dizajn faznih polja, dizajniranje i modeliranje širokopojasnih antena, ispravljanje grešaka u polju, dizajniranje ugradbenih antena. . .
- U biomedicini- u svrhu detekcije Parkinsonove bolesti na temelju drhtavice, optimizacije biomehaničkog ljudskog pokreta, klasifikacije raka i predviđanja ostatka života, dizajniranje lekova.
- Za dizajniranje bluetooth mreža, usmeravanje, za izgradnju radar-skih mreža.
- Kombinatorni problemi- rešavanje problema trgovačkog putnika, optimizacija puta. . .
- U kombinaciji s neuronskim mrežama, PSO se koristi za inverziju neuronskih mreža, kontrolu neuronskih mreža za nelinearne procese, kontrolu mobilnih neuronskih mreža, izgradnju neuronskih kontrola.
- Koristi se u funkciji predviđanja kvaliteta i klasifikacije vode, u ekološkim modelima, meteorološkim predviđanjima.

- U kombinaciji s neuronskim mrežama, PSO se koristi za inverziju neuronskih mreža, kontrolu neuronskih mreža za nelinearne procese, kontrolu mobilnih neuronskih mreža, izgradnju neuronskih kontrola.
- Koristi se u funkciji predviđanja kvaliteta i klasifikacije vode, u ekološkim modelima, meteorološkim predviđanjima.

4.1 Primer rešavanja problema rekonfiguracije i planiranja distributivne mreže

Topologije distributivnih mreža su dugi niz godina bile sa ograničenim mogućnostima promene strukture, zbog niskog stepena automatizacije i ograničenih mogućnosti daljinskog upravljanja. Problem određivanja optimalne konfiguracije distributivne mreže je kompleksan, kombinatoran, nelinearan, diskretan za čije rešavanje su idealne **metaheurističke** metode koje koriste modele zasnovane na prirodnim procesima među koje spada i **optimizacija rojem čestica**. Pored toga, uvažavanjem tehničkih ograničenja i različitih optimizacionih kriterijuma, određivanje optimalnog uklopnog stanja mreže se značajno usložnjava.

Cilj rekonfiguracije i planiranja distributivnog sistema je optimizacija ukupnih troškova elektroenergetske kompanije da opslužuje potrošačke zahteve – opterećenja koji obično rastu sa vremenom.

Za rešavanje ovog problema koristi se metaheuristička metoda zasnovana na optimizaciji rojem čestica sa ciljem minimizacije ukupnih gubitaka aktivne snage u mreži.

Formulisanje problema:

Rekonfiguracija distributivne mreže sa ciljem minimizacije ukupnih gubitaka aktivne snage u mreži prema teoriji grafova svodi se na pronalazak optimalne konfiguracije mreže, odnosno pronalazak razapinjućeg stabla grafa za koji je optimizacioni kriterijum ispunjen. Pod razapinjućem stablom (eng. spanning trees) podrazumevamo podgraf koji se dobija iz grafa uklanjanjem određenog broja grana iz grafa, a da pritom graf ostane povezan sa istim brojem čvorova.

Faktori koji se moraju razmatrati u optimizaciji distributivnog sistema:

- minimizacija gubitka energije,
- minimizacija ulaganja u nove objekte i distributivnih vodova,
- maksimizacija pouzdanosti sistema.

Povrh toga, neka ograničenja se moraju uzeti u obzir:

- kapacitet vodova,
- naponski nivo opterećenja u čvorovima,
- grafikon povezivanja,
- radikalnost mreže.

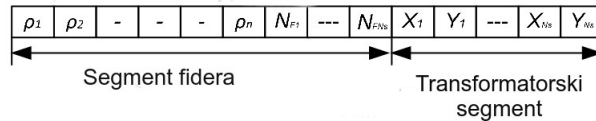
Problem je formulisan sa dve funkcije cilja. Prva funkcija cilja odnosi se na gubitke energije. U ovoj funkciji figurišu troškovi održavanja i troškovi gubitaka kao i fiksni troškovi. Druga funkcija cilja koja se koristi je vezana za pouzdanost sistema i smanjuje ukupnu neisporučenu energiju. U ovoj funkciji figurišu stope otkaza i trajanja popravke svake napojne grane. Minimizacija ovog cilja maksimizuje pouzdanost mreža.

Prema $x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1}$ pozicija i -te čestice u d -dimenzionalnom prostoru rešenja izračunata u $k+1$ -oj iteraciji u opštem slučaju nije celi broj, što znači da se PSO algoritam ne može primenjivati za rešavanje diskretnih optimizacionih problema. U tu svrhu 1997. godine Eberhart i Kennedy predstavili su binarnu verziju PSO algoritma, koja predstavlja modifikaciju PSO algoritma za rešavanje diskretnih optimizacionih problema. Modifikacija se ogleda kroz normalizaciju brzine čestice na datoj poziciji u intervalu $[0, 1]$.

Kodiranje čestica:

Šema kodiranja čestica kodira topologiju mreže, odnosno, transformatorske stanice i putanje fidera. Koriste se dve vrste šema kodiranja: direktno i indirektno. Najčešće korišćeno kodiranje je direktno međutim slabost takvog pristupa je potencijalno dobijanje upetljanih mreža. Da bi se to izbeglo, koristi se hibridni pristup direktnog i indirektnog kodiranja, gde čestice sadrže dva segmenta.

- **Segment fidera** – Sastoji se od podsegmenta “pristrasnosti” čvora vrednosti ρ , $\rho \in (-1, 1)$, odnosno, indirektna informacija koja dekodiranjem daje mrežnu topologiju i drugi podsegment sadrži broj fidera (N_{Fi} za transformatorske podstanice i) za svaku podstanicu, što je direktna informacija.
- **Segment trafostanica** – Sadrži direktne informacije o lokaciji svake transformatorske stanice X_i i Y_i koordinate transformatorskih stanica u dvodimenzionalnoj ravni).



Slika 4: Šema kodiranja

Šema dekodiranja:

U ovom pristupu čvorovi su uzastopno izabrani i dodati na terminalni čvor rastućom putanjom na osnovu minimalne vrednosti proizvoda troškova grane i vrednosti “pristrasnosti” čvora. Troškovi grane se uzimaju kao udaljenosti (D) između njegovih početnih i krajnjih čvorova. Po predloženoj šemi, dobija se radialna mreža i generišu se bočne grane u

odnosu na glavne grane. Osim toga, u realnoj distributivnoj mreži, razumno je da se ograniči povezivanje čvora sa samo nekoliko susednih čvorova. Za tu svrhu formirana je matrica povezivanja M , dimenzije $n \times n$, gde je n broj čvorova. Ako je dozvoljena veza između čvorova i i j , onda je $M(i,j) = 1$, inače $M(i,j) = 0$. Moguća povezanost čvora postoji sa fiksnim brojem svojih najbližih susednih čvorova.

Pseudokod:

Početak: /*Inicijalizacija - podešavanje veličine populacije i maksimalnog broja iteracija k_{max} */

$k = 0$;
 $s^k = s_i^0$; /*Generisanje početne tačke pretrage za celu populaciju po predloženoj šemi kodiranja */
 $v^k = v_i^0$; /*Generisanje početne brzine pretrage za celu populaciju po predloženoj šemi kodiranja */

Dekodiranje svih čestica; /*Po predloženoj šemi dekodiranja*/
 Oceniti $f(k)$; /* Izračunati vrednosti funkcije cilja svake čestice u tekućoj populaciji k */

Ponavljati: /*Globalna iteracija, k */
 $v_i^k + 1 = wv_i^k + c_1 rand_1 x(pbest_i - s_i^k) + c_2 rand_2 x(gbest_i - s_i^k)$; /*Ažuriraj brzinu za sve čestice */
 $s_i^{k+1} = s_i^k + v_i^{k+1}$; /*Ažuriraj poziciju za sve čestice*/

Dekodiranje svih čestica;
 Oceniti $f(k)$; /* Izračunati vrednost funkcije cilja svake čestice u tekućoj populaciji k */

If $f(k) > pbest(k)$ **Then** $pbest(k+1) = f(k)$ /*Prihvati funkciju cilja $f(k)$ čestice kao $pbest(k+1)$ za tu česticu*/

If $pbest(k) > gbest(k)$ **Then** $gbest(k+1) = pbest(k)$ i pamti k -tu česticu /*Prihvati $pbest(k)$ kao $gbest(k+1)$ i sačuvaj k -tu česticu kao najbolju*/
 $k = k + 1$;

Dok nije ispunjen zahtev za zaustavljanje; /* $k > k_{max}$ */

Izlaz: Najbolje nađeno rešenje

5 Topologije uticaja

Literatura

- [1] M.A. El-Shorbagy and Aboul Ella Hassanien. Particle swarm optimization from theory to applications. *Int. J. Rough Sets Data Anal.*, 5(2):1–24, April 2018.
- [2] Andries P Engelbrecht. *Computational intelligence: an introduction*. John Wiley & Sons, 2007.
- [3] Hongbo Liu, Bo Li, Ye Ji, and Tong Sun. Particle swarm optimisation from lbest to gbest. In *Applied soft computing technologies: the challenge of complexity*, pages 537–545. Springer, 2006.
- [4] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.
- [5] Dian Palupi Rini, Siti Mariyam Shamsuddin, and Siti Sophiyati Yuhani. Particle swarm optimization: technique, system and challenges. *International journal of computer applications*, 14(1):19–26, 2011.

- [6] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft Computing*, 22(2):387–408, 2018.