Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti: Kandić Stefan, SW73/2017

Papić Nikola, SW14/2017

**Predmet: Nelinearno programiranje i evolutivni algoritmi**

Broj projektnog zadatka: 12

Tema projektnog zadatka: PSO algoritam, „black box“ optimizacija

# Opis problema

Neka je data proizvoljna funkcija f(x), gde x može da predstavlja jednu ili kolekciju vrednosti. Jedino što znamo o ovoj funkciji je da izračunamo njenu vrednost za ulaz koji zadovoljava zahteve njenog ulaznog parametra.

Zadatak je pronaći minimalnu vrednost funkcije, tj. ulaz koji za koji će vrednost funkcije biti najmanja.

Posto nam funkcija može biti matematički nepoznata, analitičko rešenje nekada neće biti moguće, ali korišćenjem PSO algoritma možemo samo na osnovu njenih izlaza odrediti njen optimum.

# Uvod

Particle swarm optimization (PSO) je optimizacioni algoritam razvijen od strane Dr. Eberhart i Kennedy – ja, koji su njime pokusali da simuliraju ponašanje jata ptica.

Algoritam pocinje kreiranjem populacije proizvoljnog broja cestica i računanjem vrednosti nepoznate funkcije u njima.

Nakon toga, ove čestice se u prostoru pomeraju u zavisnosti od pozicije čestice koja ima najmanju vrednost kao i odnosu na sopstvenu poziciju koja je imala optimalnu vrednost.

Ideja je da ce na ovaj način čestice pretražiti “prostor” funkcije u potrazi za najmanjom vrednošću i na kraju se okupiti oko pozicije gde se nalazi sam optimum.

Algoritam se zavrsava ili posle određenog broja iteracija, tj pomeranja čestica ili nakon što same promene u rezultatu padnu ispod zadate preciznosti.

# Implementacija

# PSO

Algoritam je implementiran u vidu klase PSO. Njeni važniji atributi su koeficijenti koji se koriste pri računanju brzine čestice ω(faktor inercije), cp(kognitivni faktor) i cg(socijalni faktor) kao i sam kriterijum optimizacije, globalna najbolja tačka I vrednost kriterijuma u toj tački. Algoritam se započinje metodom optimize() u kojoj se za određen broj iteracija u svakoj sve čestice pomere za jednom.

Vrednosti koeficijenata se računaju po ovoj formuli nakon svake iteracije:

coeff= (coeffmax – coeffmin) \* ((Niter - i) / Niter) + coeffmin

Gde je coeffkoeficijent koji se računa coeffmax maksimalna moguća vrednost koeficijenta, a coeffmin minimalna, Niter ukupan broj iteracija i iredni broj trenutne iteracije.

Odabir parametara algoritma

Koeficijenti u toku algoritma mogu biti:

1. Konstantni: ω = 0.72, cp = 1.49, cg = 1.49
2. Promenljivi (Ratnaweera) – na pocetku algoritma imaju vrednosti ω : 0.9 - 0.4, cp : 2.5 - 0.5, cg : 0.5 - 2.5, i postepeno se menjaju do ω = 0.4, cp = 0.5, cg = 2.5

Ideja kod promenljivih koeficijenata je da na pocetku algoritma cestice prave vece pomeraje u svojim pozicijama kako bi bolje istrazile prostor i da se ponasaju manje zavisno od globalno najbolje pozicije, a da kako algoritam odmice sve vise usporavaju i pozicioniraju se blize najboljoj poziciji.

Particle

Čestice su implementirane u posebnoj klasi Particle. Značajni atributi ove klase su position – označava trenutnu poziciju lestice

p – najbolja pozicija čestice

v – brzina čestice

Ova klasa sadrži samo dve metode move I evaluate. U prvoj se računa nova pozicija čestice tako pto se stara pozicija sabira sa novom brzinom koja se računa po formuli: Vi­­­­ = ω \* Vi-1 + cp \* rp \* (p - position) + cg \* rg \* (g - position)

U metodi evaluate se proverava da li nova pozicija daje bolju vrednost kriterijuma optimizacije u odnosu na svoju najbolju I globalnu. Ako je manja od sopstvene najbolje vrednosti onda se trenutna pozicija proglašava za najbolju za ovu česticu,a ako je bolja I od globalne najbolje i njena vrednost se menja na trenutnu poziciju te čestice.

# Alternativni topologije PSO algoritma

U originalnoj verziji algoritma, sve čestice se nalaze u kontaktu, tj one poseduju zajedničku najbolju poziciju.

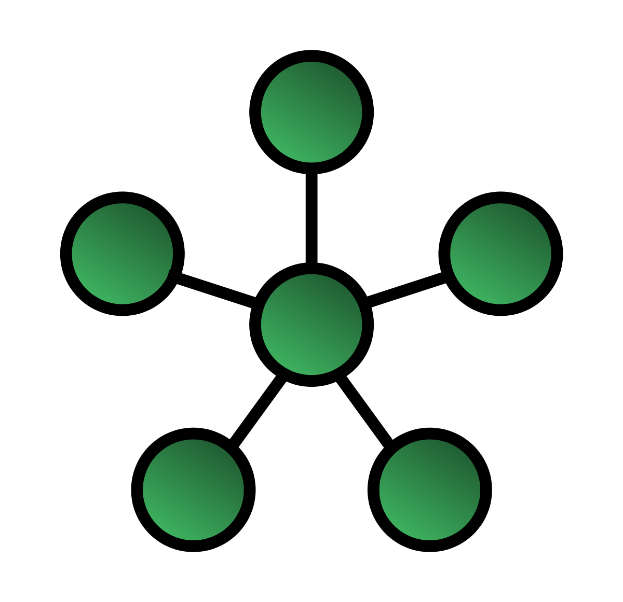
Međutim to može dovesti do gomilanja čestica u lokalnom minimumu, te je došlo do razvoja različitih topologija koje obezbeđuju razliciti nacin komunikacije između samih čestica.

1. Topologija zvezde

Sve informacije u roju prolaze kroz jedan centralni cvor, koji utiče i na koji utiču sve ostale cestice.

U svakoj iteraciji ceo roj se pomera u zavisnosti od centralne čestice, a centralna čestica se pomera u odnosu na najbolju globalnu vrednost.

Topologija zvezde na ovaj način izoluje čestice roja jednu od druge, i ovakav efekat bi trebao da onemoguci preranu konvergenciju samog roja u u lokalni optimum.



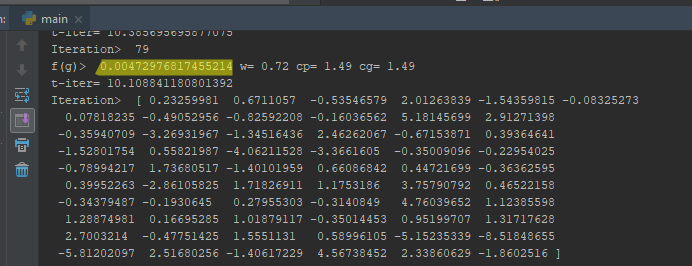
1. Topologija prstena

Drugi način da se izbegne prerana konvergencija je korišćenje topologije prstena. U ovoj topologiji svaka čestica ima referencu na dve druge čestice levu I desnu. Pri pomeranju koristi se ista formula kao i kod umrežene topologije samo što se umesto vektora u kojem je globalni minimum koristi manja od najboljih pozicija leve I desne čestice.

Ovo usporava konvergenciju i povećava vreme nalazka minimuma ali smanjuje šanse za preranu konvergenciju.

Rezultati

Najbolje pronađeno resenje za funkciju optimality\_criterion:



Poređenje resenja različitih topologija sa konstantnim koeficijentima:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Original PSO | Star PSO | Ring PSO |
| Broj čestica | 300 | 300 | 300 |
| Broj iteracija | 50 | 50 | 50 |
| Vrednost funkcije | 0.0179 | 0.0194 | 0.0345 |

Poređenje resenja različitih topologija sa ratweerom:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Original PSO | Star PSO | Ring PSO |
| Broj čestica | 300 | 300 | 300 |
| Broj iteracija | 50 | 50 | 50 |
| Vrednost funkcije | 0.0190 | 0.0197 | 0.0391 |

# Zakljucak

Za fuknkciju optimality criterion mozemo zakljuciti sledece:

* Prilikom rada sa konstantnim koeficijentima dobijamo malo bolje rezultate
* Koriscenje fully-connected (Original PSO) topologije moze dovesti do “zaglavljivanja” algoritma u lokalnom optimumu cesce nego kod ostalih
* Fully–connected i topologija zvezde imaju slicne rezultate u oba slucaja, ali je najbolji rezultat postignut potpuno povezanom topologijom
* Topologija prstena ima znacajno losije performanse u odnosu na ostale topologije
* Izbor topologije moze znacajno da utice na rezultate algoritma u zavisnosti od problema koji se resava

Ideje za dalji nastavak rada na projektu:

* Ispitivanje performansi drugih topologija, funkcija i njihovo poredjenje
* Primena PSO algoritma na koeficijente od kojih sam PSO zavisi (ω, cp, cg) i poredjenje rezultata