Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti: Jokić Jovan, SV47/2020

Bogdanović Vukašin, SV09/2020

Predmet: Nelinearno programiranje i evolutivni algoritmi

Broj projektnog zadatka: 12

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

# **Opis problema**

Za zadatu funkciju f koja prima ulazni vektor w kao parametar potrebno je izračunati globalni minimum. Funkciju f je moguće računarski evaluirati pomoću date funkcije optimality\_criterion, međutim ona je u analitičkom smislu nepoznata. Ovaj problem je moguće rešiti pomoću „black-box“ optimizacije. „Black-box“ optimizacijom ćemo se fokusirati na razvoju metodologije koja se oslanja na statističko i mašinsko učenje kako bi rukovali eksperimentalnim i simulacionim podacima. U tu svrhu ćemo koristiti PSO (Particle Swarm Optimization) algoritam.

# **Uvod**

PSO algoritam funkcioniše slično kao i jato ptica u potrazi za hranom. Počinjemo sa određenim brojem tačaka koje su slučajnim putem raspoređene u prostoru i koje ćemo u daljem tekstu nazivati „čestice“, a zatim ćemo ih pustiti da traže minimume u krećući se u slučajno odabranim pravcima. Svakim sledećim korakom, svaka čestica će tražiti minimume oko najminimalnije tačke koju ona ikada pronašla, kao i oko minimalne tačke koju je pronađena od strane celog roja čestica. Posle određenog broja koraka, minimumom funkcije možemo smatrati najminimalniju tačku koju je ceo roj čestica pronašao.

Ako imamo N čestica, tada ćemo česticu k tokom iteracije i obeležavati sa Xk(t), pri čemu Xk(t) predstavlja tačku sa 60 koordinata.

Xk(t)=( X1k(t), X2k(t),..., X60k(t))

Analogno tome, brzinu ćemo označavati sa Vk(t), i važiće

Vk(t)=( V1k(t), V2k(t),..., V60k(t))

Koordinate čestica u svakoj narednoj iteraciji ćemo računati po formuli

**Xk(t+1)= Xk(t)+ Vk(t+1)**

Brzine čestica u svakoj sledećoj iteraciji ćemo računati po formuli

**Vk(t+1) = wk Vk(t) + cprp(pb- Xk(t)) + cgrg(gb- Xk(t))**

Iz jednačine za računanje brzine čestice u t+1 iteraciji vidimo da se ona sastoji od 3 komponente:

1. Inerciona komponenta - wk Vk(t)

Inerciona komponenta definiše stepen mobilnosti čestice, odnosno koliko je čestici lako da se kreće kroz prostor. Ova komponenta je definisana proizvodom:

* + - * 1. Konstante inercije wk koja uzima vrednosti od 0.9 do 0.4 po formuli

wk=0.9-k/N(0.9-0.4)

N- ukupan broj iteracija, k- trenutna iteracija

* + - * 1. Brzinom čestice k u prethodnoj iteraciji Vk(t)

1. Kognitivne komponente- cprp(pb- Xk(t))

Kognitivna komponenta definiše koliko će se svaka čestica oslanjati na sopstveno iskustvo. Ova komponenta je definisana proizvodom:

* + - * 1. Konstanti ubrzaznja cp irp, pri čemu cp uzima vrednosti od 2.5 do 0.5, a rp uzima slučajnu vrednost u opsegu od 0 do 1 pri čemu se njena vrednost menja prilikom svake iteracije
        2. Trenutne udaljenosti od lične najbolje pozicije pb

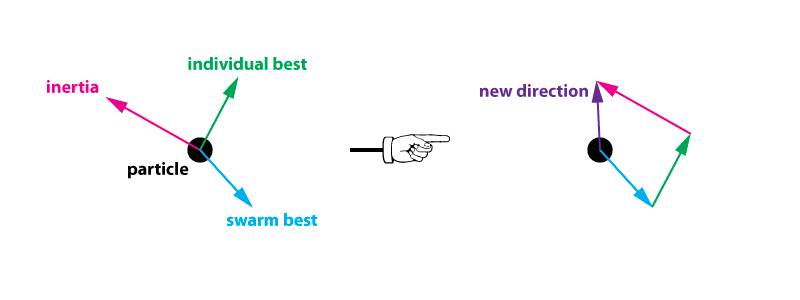
1. Socijalne komponente komponente- cgrg(gb- Xk(t))

Socijalna komponenta definiše koliko će se svaka čestica „slušati“ iskustvo drugih čestica iz roja. Ova komponenta je definisana proizvodom:

* + - * 1. Konstanti ubrzaznja cg irg, pri čemu cp uzima vrednosti od 0.5 do 2.5, a rg uzima slučajnu vrednost u opsegu od 0 do 1 pri čemu se njena vrednost menja prilikom svake iteracije
        2. Trenutne udaljenosti od globalne najbolje pozicije gb

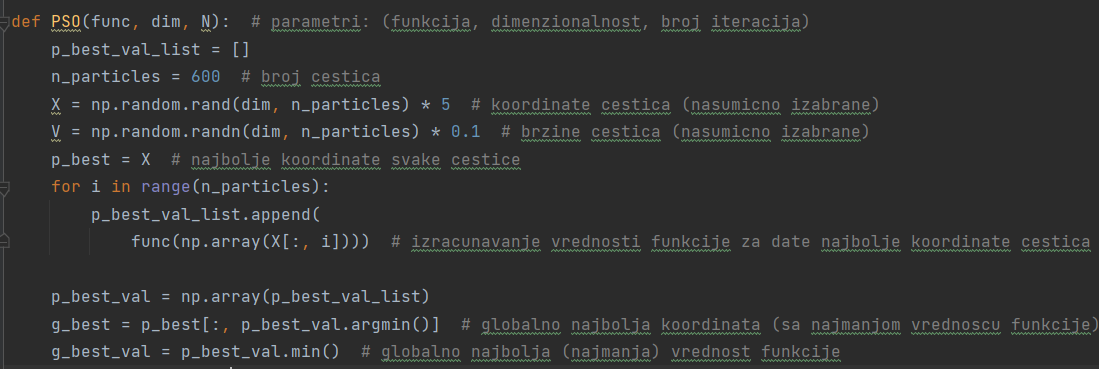
Nakon podešavanja novih vrednosti koordinata i brzina poredimo vrednosti trenutne najbolje lične vrednosti funkcije sa novim vrednostima podešavamo nove najbolje pozicije tako što biramo manju od svake dve. Takođe, ažuriramo i globalnu najbolju poziciju poređenjem vrednosti funkcija svih najboljih ličnih pozicija.

Zavisnost kretanja čestice od ove tri komponente najbolje sledeća opisuje slika



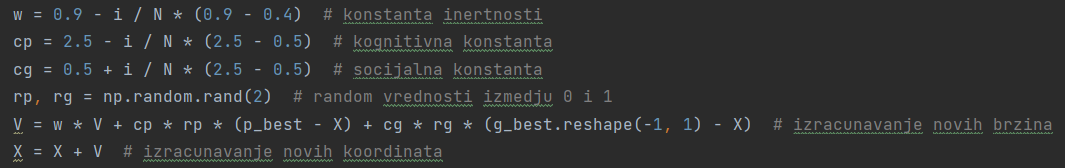
# **Implementacija**

Algoritam se sastoji iz 4 koraka:

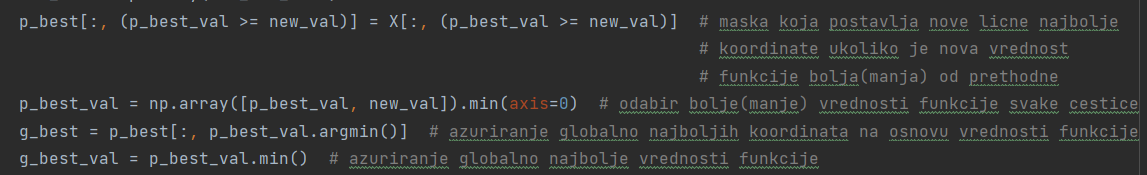
1. **Inicijalizacija**

Inicijalizaciju započinjemo definisanjem broja čestica. Zatim, svakoj čestici dodeljujemo nasumične koordinate kao i nasumičnu brzinu kretanja. Kako čestice uopšte nisu istraživale, njihova trenutna pozicija će takođe biti i njihova najbolja lična pozicija. Zatim, uvodimo vektor globalne najbolje pozicije, koji predstavlja najbolju poziciju ikad pronadjenu u čitavom roju čestica.

1. **Ažuriranje pozicije i brzine pomoću formula**

U ovom delu implementacije algoritma pristupamo petlji koja se izvršava zadati broj puta. Uvodimo konstantu inertnosti, kognitivnu konstantu i socijalnu konstantu koje ažuriramo prilikom svake iteracije. Nakon toga ažuriramo i brzinu čestica po navedenoj formuli.

1. **Ažuriranje ličnih i globalno najboljih pozicija**

****Na kraju iteracije ažuriramo podatke ličnih i globalnih pozicija čestica. Za svaku poredimo njenu ličnu najbolju vrednost sa trenutnom vrednoscu, zatim biramo manju od dve i uzimamo je kao ličnu najbolju vrednost. Sličnu proveru radimo i za globalnu najbolju poziciju, gde poredimo sve lične najbolje vrednosti čestica.

1. **Stajanje nakon zadatog broja iteracija**

Kada iteriramo kroz petlju zadati broj puta trenutne globalne najbolje pozicije i vrednost funkcije u toj tački predstavlja, respektivno, tačku u kojoj se nalazi globalni minimum i vrednost globalnog minimuma. Važno je napomenuti da preciznost izračunavanja zavisi od zadatog broja iteracija od strane korisnika.

# Rezultati

Na osnovu pokretanja programa sa različitim parametrima možemo otprilike da naslutimo gde se nalazi minimum, uprkos tome što se u samom algoritmo nalazi velik broj „slučajnih“ činilaca.

50 iteracija sa 600 čestica

Fmin=0.10089035411404096

50 iteracija sa 1200 čestica

Fmin= 0.1252087670018186

100 iteracija sa 600 čestica

Fmin= 0.0752560864180069

150 iteracija sa 600 čestica

Fmin= 0.06808575852783809

200 iteracija sa 600 čestica

Fmin= 0.045788299729392835

# Zaključak

Najentuzijastičniji deo PSO algoritma je stabilna topologija koja obezbeđuje **komunikaciju između čestica u roju** i pomaže bržem učenju svake jedinke u postizanju globalnog minimuma. Ovaj algoritam obavlja prilično dobar posao u potrazi za globalnim minimumom, međutim njegova tačnost nije zagarantovana i zato smo se, prilikom istraživanja ovog algoritma, sreli sa mišljenjima da PSO algoritam rezultuje najverovatnijim globalnim minimumom. Takođe, tokom istraživanja otkrili smo da „black-box“ optimizacije kao i PSO algoritam imaju potencijal i široku primenu u nauci.