Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti:	Rodušek Vladimir, SW23/2017 Pavlov Milan, SW35/2017
Predmet:	Nelinearno programiranje i evolutivni algoritmi
Broj projektnog zadatka:	12
Tema projektnog zadatka:	PSO algoritam, "black-box" optimizacija

Opis problema

Data je funkcija f(**w**) koju smatramo nepoznatom. Argument funkcije **w** je vektorska veličina sa 60 dimenzija. Funkcija predstavlja veštačku neuronsku mrežu i nepoznata je u analitičkom smislu. Rešenje problema je pronalazak optimuma te funkcije koje predstavlja najmanju grešku same mreže.

Za optimizaciju funkcije koristiti PSO algoritam (Optimizacija rojem čestica).

Opis algoritma

PSO algoritam se oslanja na roj čestica za koji svaka čestica je opisana svojom pozicijom (**w** vektorska veličina sa **n** dimenzijom), vrednošću funcije u toj poziciji, njenom brzinom takođe definisanom u datoj **n** dimenziji i pored toga sadrži poziciju najbolje vrednosti funkcije (najbliža pozicija, te čestice, minimumu).

On za definisan broj iteracija pomera čestice, tj. čestice "zuje" oslanjajući se na pomeranje ka najboljoj poziciji koji je roj imao. U svakoj iteraciji se ažurira najbolja pozicija i same čestice, ali i roja tj. najbolja pozicija od svih čestica. Vektor brzine čestice se prema formuli pomera ka najboljem trenutnom rešenju.

Formula izleda ovako:

i – predstavlja trenutnu iteraciju
v – predstavlja brzinu
ω – predstavlja faktor inercije
c_p – kognitivni faktor
c_g – socijalni faktor
r_p, r_g – slučajni brojevi u intervalu (0, 1)
p – najbolja pozicija čestice
x – trenutna pozicija čestice
g – najbolja pozicija od svih čestica

$$v[i] = \omega[i] \cdot v[i-1] + c_p[i] \cdot r_p[i] \cdot (p[i] - x[i]) + c_g[i] \cdot r_g[i] \cdot (g[i] - x[i])$$

inercijalna komponenta
kognitivna komponenta
socijalna komponenta

Opis faktora

- Uočeno je da se performanse algoritma (naročito preciznost u okolini rešenja) bitno poboljšavaju uvođenjem promenljivog inercionog faktora. Korišćeno je da se **ω** smanjuje od 0.9 do 0.4
- Faktor $\mathbf{c_p}$ se tokom preterage smanjuje; u početku se jedinke vode sopstvenim iskustvom. Korišćeno je da se $\mathbf{c_p}$ smanjuje od 2.5 do 0.5
- Faktor $\mathbf{c_g}$ se u toku pretrage povećava; uticaj grupnog iskustva Korišćeno je da se $\mathbf{c_g}$ povećava od 0.5 do 2.5

Struktura programa

Algoritam je implementiran u **Python3** programskom jeziku, za lakši rad sa matricama i nizovima korišćena je biblioteka **numpy**. Okruženje koje je korišćeno je Eclipse IDE.

Čestica je realizovana kao klasa sa adekvatnim poljima i metodama.

```
class Particle(object):
    """
        Klasa predstavlja jednu cesticu roja
        Sadrzi sve potrebne atribute vezane za cesticu
    """

    def __init__(self, w):
        self.position = np.copy(w)
        self.velocity = np.random.uniform(-1, 1, len(w))
        self.peronal_best = self.position
        self.feval = None
        self.feval_best = None
```

Prilikom svake iteracije za svaku česticu se pozivaju metode “evalute” i “update_data”.

```
def update_data(self, swarm_best_pos, iteration, opt):
    """
        Azurira sve atribute za datu iteraciju
    """
    w = interpolation(opt['w_start'], opt['w_end'], opt['max_it'], 0, iteration)
    cp = interpolation(opt['cp_start'], opt['cp_end'], opt['max_it'], 0, iteration)
    cg = interpolation(opt['cg_start'], opt['cg_end'], opt['max_it'], 0, iteration)

    rp = np.random.random()
    rg = np.random.random()

    cognitive = np.multiply(cp * rp, np.subtract(self.peronal_best, self.position))
    social = np.multiply(cg * rg, np.subtract(swarm_best_pos, self.position))
    self.velocity = np.add(np.multiply(w, self.velocity), np.add(cognitive, social))

    self.peronal_best = np.add(self.position, self.velocity)
```

Za promenu faktora tokom petlji se koristi interpolacija prema datoj iteraciji

```
def interpolation(start, end, max_it, min_it, curr_it):
    """
        Interpolira izmenu za prosledjeni faktor
    """
    return start + ((end - start) / (max_it - min_it)) * (max_it - curr_it)
```

Za lakše biranje parametara algoritma realizovan je rečnik koji predstavlja opcije algoritma

```
# opcije algoritma
options = { 'max_it'       : 100,       # broj iteracija
            'particle_num' : 30,        # broj cestica
            'dim'          : 60,        # dimenzija
            'bounds'       : (-10, 10), # opseg nasumicnih pocetnih vrednosti za svaku dimenziju
            'cp_start'     : 2.5,       # startna vrednost kognitivnog faktora
            'cp_end'       : 0.5,       # finalna vrednost kognitivnog faktora
            'cg_start'     : 0.5,       # startna vrednost socijalnog faktora
            'cg_end'       : 2.5,       # finalna vrednost socijalna faktora
            'w_start'      : 0.9,       # startna vrednost inercijalnog faktora
            'w_end'        : 0.4 }      # finalna vrednost inercijalnog faktora
```

Prilikom traženja optimalnog rešenja menjani parametri su broj iteracija, broj čestica kao i granice vrednosti za svaku od dimenzija čestice.

Rezultati algoritma

PSO algoritam ne može da garantuje da li je optimum samo lokalni ili je i globalni. Zbog nepoznavanja funkcije ne možemo konkretno odrediti, a ni proveriti tačnost rešenja, ali možemo da za različite faktore diskutujemo neka rešenja.

Za početno testiranje smo suzili intervale granice na dužinu od 5 tj. svaki interval se pomerao za 5. Time smo tražili lokalne minimume za podintervale intervala od -100 do 100. Najbolje rešenje nam je bilo na intervalu [-5, 0] i tu smo dobili da je greška vrednosti **0.11527677194021697**. Za svaki podinterval je korišćeno 100 čestica i 100 iteracija.

Primetivši da je interval [-5, 0] najbolji, fokusirali smo se na taj interval za dalje testiranje.

Najbolje rešenje je bilo za date parametre (broj čestica i broj iteracija):

```
Particle number: 100
Max iteration: 1000
Bounds: (-5, 0)

Best position = [
        -3.851326839468945
        -1.075956647763559
        -3.184342704745325
        -4.801991186280192
        .
        .
        .
        -0.18405967947447976
        -1.1201941808379026
        -0.2728784926364032
        -4.795702464647143
]
Value of best position = 0.09453836991165618
Execution time: 1039 seconds
```

Napomena! Odsečen je deo pozicije jer je prevelika slika za 60 dimenzija.
U log datoteci je u celokupnom stanju.