Univerzitet u Sarajevu
Elektrotehnički fakultet
Predmet: Optimizacija resursa
Akademska godina: 2020/2021
Student: Samra Mujčinović
Datum: 21.1.2021.

Izvještaj sa laboratorijske vježbe 10

Zadatak 1

Ideja za rješenje zadatka

U ovoj laboratorijskoj vježbi traženo je da implementiramo algoritam optimizacije rojem čestica(PSO). Algoritam optimizacije rojem čestica (PSO - Particle Swarm Optimization) do određene mjere modelira način prenošenja informacija koji se susreće kod živih bi ća koja žive u rojevima. Osnovni algoritam PSO su Kennedy, Eberhart i Shi definirali 1995. godine,kao "kolektivni anarhični iterativni metod sa naglaskom na kooperaciji".Kako se PSO koristi za određivanje optimuma, nema potrebe modelirati sve elemente mehanizma prenošenja informacija koji se susreću npr. kod roja pčela. Radi toga svaka čestica prenosi samo vrijednost kriterija i tačku za koju se ta vrijednost kriterija postiže. Pri tome u algoritmu PSO nema ekvivalenta "povratku u košnicu", odnosno nekoj centralnoj tački u problemskom prostoru, u kojoj se vrši razmjena informacija. Osnovna varijanta PSO, u poređenju sa evolucionim algoritmima, ne posjeduje mehanizam selekcije. Odsustvo ovog mehanizma je motivirano očekivanjem da i trenutno loša potencijalna rješenja, kroz odredeni broj iteracija, mogu postati dobra, te se radi toga ne uklanjaju iz populacije. Na svakoj iteraciji, svaka čestica odabire određen broj čestica kojima će prenijeti informaciju. Isto tako, svaka čestica prima informaciju od određenog broja čestica (tzv. informanata), o najboljoj poziciji koju su pronašle do tekuće iteracije. Osim toga, svaka čestica memoriše vlastitu najbolju otkrivenu poziciju. U modelu mehanizma, na osnovu kojega se određuje nova pozicija čestice, se vodi računa i o brzini kojom se čestica kreće. Radi toga svaka čestica memoriše i svoju trenutnu brzinu, koja se modificira sa svakom iteracijom, na osnovu raspoloživih informacija:

- trenutne vrijednosti vlastite brzine
- memorisane najbolje vlastite pozicije
- najbolje pozicije svojih informanata na datoj iteraciji.

Matematski opis zadatka

U postavci ove vježbe date su određene informacije o ovom algoritmu, kao i pseudo kod koji će biti predstavljeni u ovom dijelu izvještaja.

Algoritam optimizacije rojem čestica (*engl.Particle Swarm Optimization– PSO*) u svojoj osnovnoj varijanti definira skup čestica koje se kreću kroz problemski prostor kao materijalne čestice koje razmjenjuju informacije o najboljoj vrijednosti kriterija. Kretanje

ovih čestica se posmatra na način sličan kretanju materijalne tačke. Koriste se sljedeće relacije u n-toj iteraciji:

$$x_{i,d}(n+1) = x_{i,d}(n) + v_{i,d}(n+1),$$
$$v_{i,d}(n+1) = W \cdot v_{i,d}(n) + C_1 r_1 \cdot [pb_{i,d}(n) - x_{i,d}(n)] + C_2 r_2 \cdot [gb_d(n) - x_{i,d}(n)],$$

gdje su:

- i– indeks čestice
- d– razmatrana dimenzija
- $x_{i,d}$– pozicija i-te čestice u dimenziji d
- $v_{i,d}$– brzina i-te čestice u dimenziji d
- $pb_{i,d}$– lokacija u dimenziji d sa najboljom vrijednosti funkcije kriterija od svih pozicija koje je posjetila i-ta čestica u dimenziji d
- gb_d– lokacija u dimenzijidsa najboljom vrijednosti funkcije kriterija
- W, C_1 i C_2– parametri algoritma
- r_1 i r_2 slučajni brojevi iz opsega [0,1]

Pseudokod algoritma:

Algoritam 1 Pseudokod algoritma optimizacije rojem čestica

Inizijalizacija populacije čestica sa slučajno odabranim pozicijama i brzinama za sve dimenzije u problemskom prostoru
while *Kriterij zaustavljanja* == **false do**
  for svaku *i*-tu česticu **do**
    Računanje nove brzine prema relaciji (2)
    Računanje nove pozicije čestice prema relaciji (1)
    Računanje vrijednosti kriterija funkcije $f(\boldsymbol{x}_i)$
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{pb}_i)$ **then**
      $\boldsymbol{pb}_i \leftarrow \boldsymbol{x}_i$
    end if
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{pg})$ **then**
      $\boldsymbol{pg} \leftarrow \boldsymbol{x}_i$
    end if
  end for
end while

Vodič kroz kôd/simulaciju

Za implementaciju ovog zadatka definisana je jedna funkcija PSO sa šest parametara:

- opseg– opseg pretraživanja problemskog prostora
- N– maksimalan broj iteracija
- M– maksimalan broj čestica
- eps– minimalna promjena po vrijednosti funkcije između dvije sukcesivne iteracije
- P– vektor koji sadrži parametreW, C1 i C2.

U nastavku je dat kod ove funkcije pisan u programskom jeziku Python, a temelji se na prethodno prikazanom pseudokodu, te ideji koja je ranije opisana u prvom poglavlju izvještaja.

```
def PSO(f, opseg, N, M, eps, P):
    #Inizijalizacija populacije čestica
    cestice = []
    brzine = []

    for i in range(M):
        cestice.append([r.uniform(opseg[0], opseg[1]), r.uniform(opseg[0], opseg[1])])
    for i in range(M):
        brzine.append([r.uniform((opseg[0] - opseg[1]) / 2, (opseg[1] - opseg[0]) / 2), r.uniform((opseg[0] - opseg[1]) / 2, (opseg[1] - opseg[0]) / 2)])

    W = P[0]
    C1 = P[1]
    C2 = P[2]
    rjesenje = cestice[0]

    i = 0
    while(i < N):
        pb = cestice[0]
        for j in range(1, M):
            if f(*cestice[j]) < f(*pb):
                pb = copy.deepcopy(cestice[j])

        if f(*pb) < f(*rjesenje):
            if abs(f(*pb) - f(*rjesenje)) < eps:
                return pb
            rjesenje = copy.deepcopy(pb)

        for j in range(M):
            for k in range(2):
                r1 = r.uniform(0, 1)
                r2 = r.uniform(0, 1)

                #Nova brzina
                brzine[j][k] = W[k] * brzine[j][k] + C1[k] * r1 * (pb[k] - cestice[j][k]) + C2[k] * r2 * (rjesenje[k] - cestice[j][k])

                #Nova pozicija
                cestice[j][k] = cestice[j][k] + brzine[j][k]

                if cestice[j][k] > opseg[1]:
                    cestice[j][k] = copy.deepcopy(opseg[1])
                if cestice[j][k] < opseg[0]:
                    cestice[j][k] = copy.deepcopy(opseg[0])

        i = i + 1
    return rjesenje
```

Osim ove funkcije, u kod su dodane i funkcije iz drugog zadatka za testiranje ispravnosti koda, zajedno sa funkcijom iscrtavanja grafika ovih funkcija. Pored navedenih funkcija iz drugog zadatka, za testiranje su dodane i još dvije funkcije:

1. Three-Hump Camel Function

$$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2 \qquad f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0,0) \quad x_i \in [-5, 5],$$

2. Matyas Function

$$f(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \quad f(\mathbf{x}^*) = 0, \text{ at } \mathbf{x}^* = (0,0) \quad x_i \in [-10, 10].$$

Rezultati kôda/simulacije

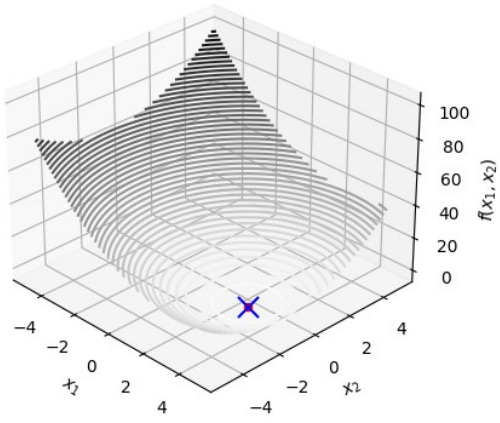
U ovom dijelu prikazat ćemo rezultate koda za funkcije date u drugom zadatku, kao i za dvije dodatne koje su prethodno navedene.

Početne vrijednosti parametara N, M, eps i P su sljedeće:

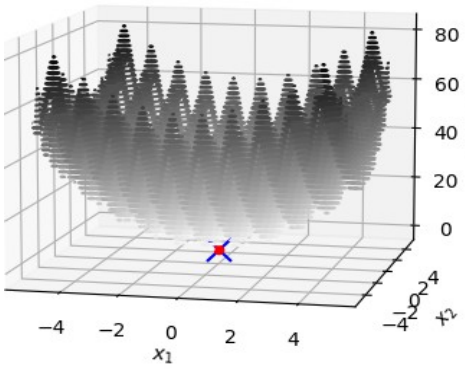
N = 1000

M = 40
eps = 0.00001
W = [0.7, 0.8]
C1 = [r.uniform(0, 1.47), r.uniform(0, 1.62)]
C2 = [r.uniform(0, 1.47), r.uniform(0, 1.62)]
P = [W, C1, C2]

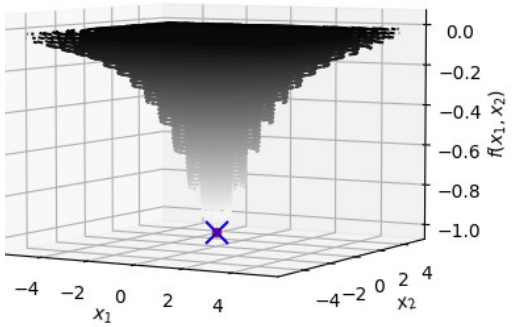
Paraboloid



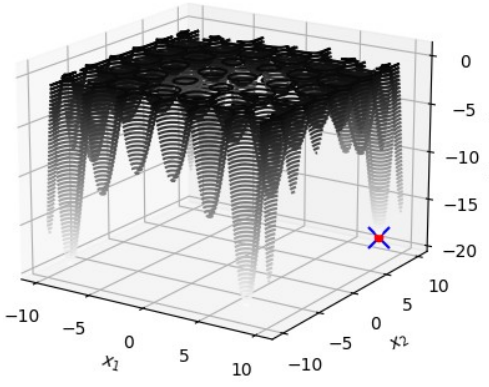
Rastrigin



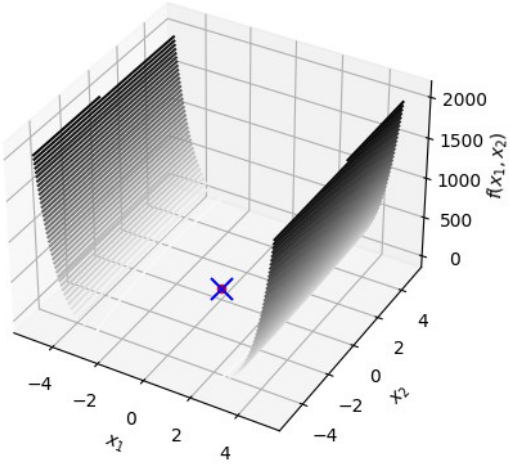
Drop-Wave



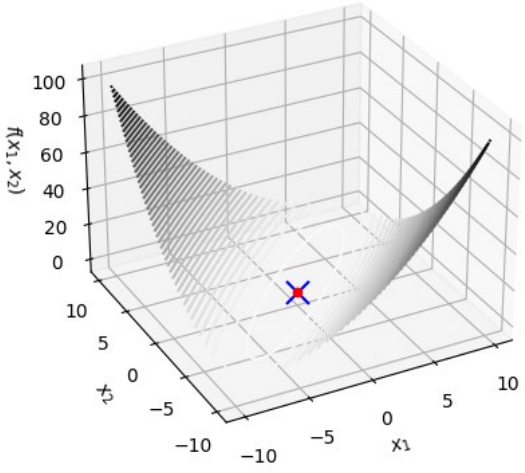
Holder Table



Three-Hump Camel



Matyas

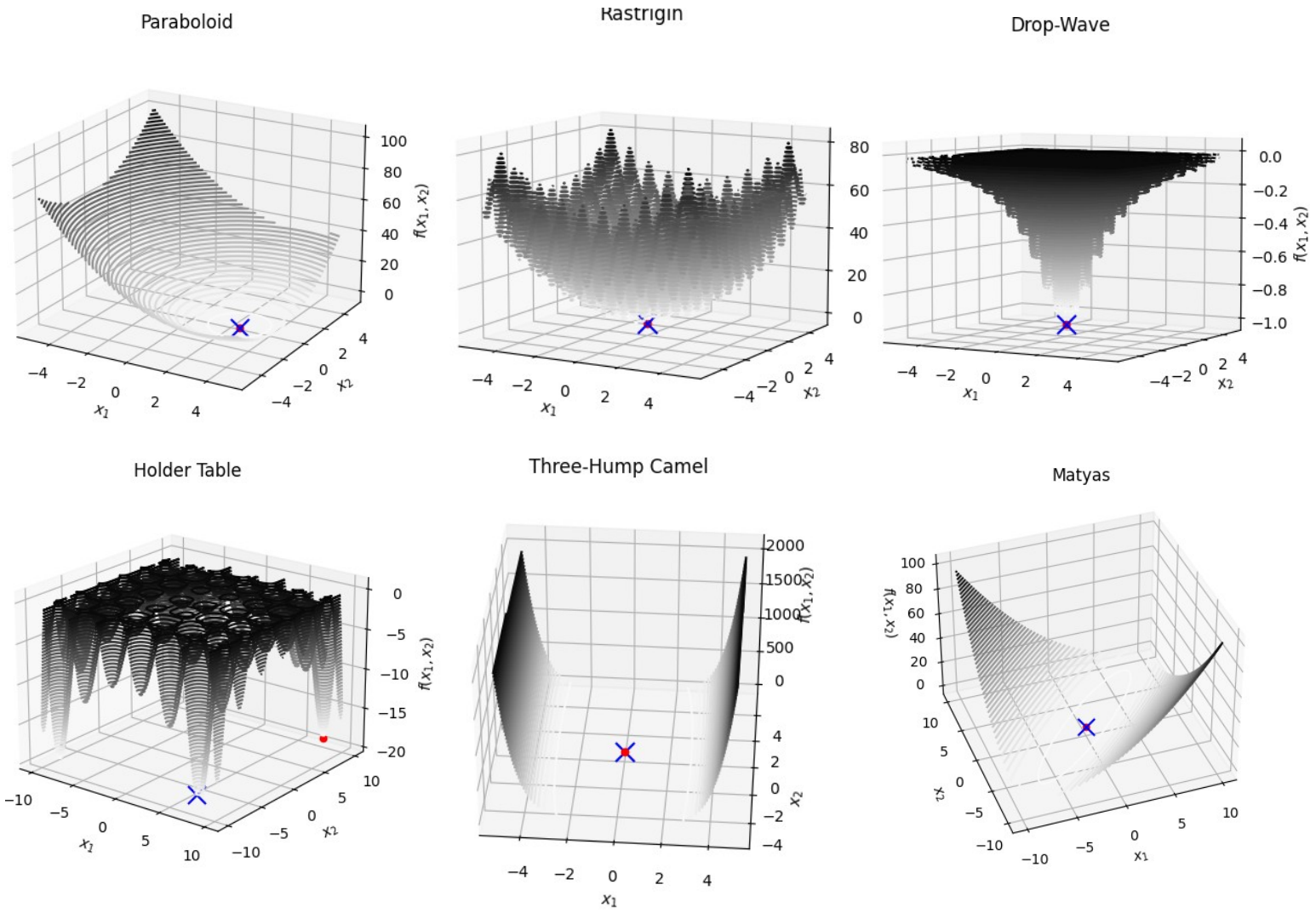


Na prethodnim slikama su grafički prikazani minimumi funkcija, dok su na sljedećoj slici prikazane tačne koordinate svakog minimuma kojeg je pronašla prethodno opisana funkcija PSO za svaku od funkcija koje su prikazane iznad. Rezultati koda, tj prikazane vrijednosti odozgo prema dolje odgovaraju slikama u redoslijedu s lijeva na desno:

```
samra@samra-laptop:~/Documents/Optimizacija resursa/lab 10$ /usr/bin/python3 "/home/samra/Documents/Optimizacija resursa/lab 10/zadatak2.py"
[3.0011451026746503, -0.9988396163561345]
[3.5293005922868535e-06, 3.0246631904636088e-05]
[-0.0006849274236691156, 1.78302174805488e-05]
[8.055554025252208, 9.664727227460409]
[0.0043174072962161615, -0.0013603289194254187]
[0.004477178834084855, -0.0006429688319201741]
```

Vidimo da pri izboru ovakvih vrijednosti početnih parametara, dobijamo izuzetno dobre rezultate, tj pronađene vrijednosti minimuma su izuzetno bliske globalnom minimumu.

Ovdje je već razmatran velik broj iteracija, te ćemo u sljedećem testiranju smanjiti ovu vrijednost(na 500). Također ćemo smanjiti broj M na donju preporučenu granicu(20). Rezultati ovog testiranja su prikazani u nastavku.



Na prethodnim slikama moguće je uočiti da smo i za ove vrijednosti parametara dobili jako dobre rezultate, a sljedeća slika na kojoj se vide koordinate svakog od pronađenih minimuma će to bolje pokazati.

```
samra@samra-laptop:~/Documents/Optimizacija resursa/lab 10$ /usr/bin/python3 "/home/samra/Documents/Optimizacija resursa/lab 10/zadatak2.py"
[3.0005258348119814, -0.9997479991714654]
[0.00014661444253686787, -1.1197459625448007e-05]
[-0.000718796829624697 2, -0.0005778228774165305]
[8.051105961350935, -10]
[0.0005143532158314282, 0.0012923252503149324]
[-0.0032307900291905084, -0.004592760822781358]
```

Vidimo da smo su lokalni minimumi pronađeni funkcijom PSO jako bliski stvarnim globalnim minimumima.

Zaključak

U ovom dijelu, nakon sprovedene analize i koda i rješenja, moguće je donijeti zaključak da ovaj algoritam, u poređenju sa dosadašnjim koje smo razmatrali na ovom predmetu, daje mnogo bolje i tačnije rezultate i to za svaku od navedenih funkcija. Naravno, i kod ovog algoritma je prisutan faktor randomizacije, te u skladu s tim u pojedinim iteracijama su rezultati varirali, ali su i dalje ostajali u bliskoj okolini minimuma. Također, veliku ulogu u pronalasku ovako dobrih rezultata imaju i početne vrijednosti parametara funkcije PSO koje su uzete na početku. One su izabrane u skladu sa preporučenim optimalnim vrijednostima iz sljedeće tabele:

Parametar —	Vrijednost
m	$20-40$
K	$3-5$
c_i^1	$0,7$ ili $0,8$
c_i^2	slučajni broj iz opsega $[0, c_{max}]$
c_i^3	slučajni broj iz opsega $[0, c_{max}]$
c_{max}	$1,47$ ili $1,62$

Zadatak 2

Ovaj zadatak je objedinjen u prvom zadatku, a rezultat njegove simulacije je prikazan u paragrafu “Rezultat koda/simulacije” u prvom zadatku. Također, opis dodatnih funkcija objedinjen je u prvom zadatku.