Univerzitet u Sarajevu
Elektrotehnički fakultet
Predmet: Optimizacija resursa
Akademska godina: 2020/2021
Student: Samra Mujčinović
Datum: 5.1.2021.

Izvještaj sa laboratorijske vježbe 8

Zadatak 1

Ideja za rješenje zadatka

U ovoj laboratorijskoj vježbi potrebno je implementirati algoritam ES(1+1) iz klase evolucionih strategija. U postavci je dat pseudokod algoritma po kojem bi on trebao biti implementiran, što je objašnjeno u sljedećem poglavlju. Za implementaciju ovog algoritma traženo je da implementiramo klasu ES sa odgovarajućim atributima I metodama. Sličnost ovog algoritma sa evolucijom živih bića proizilazi iz činjenice da su se samo bolje modifikacije uzimale u obzir(opstanak boljih jedinki), te da je modifikacija formirana mutiranjem postojećeg oblika. Evolucione strategije, za razliku od GA vrše adaptaciju fenotipa vještačkih jedinki, a mutacija se pojavljuje kao osnovni(u početku I jedini) operator pomoću kojeg ES pretražuju problemski prostor. ES(1+1) strategija ima populaciju koja se sastoji samo od jedne jedinke I nju predstavlja vektor problemskih varijabli. Mutiranjem ove jedinke dodavanjem slučajnih vrijednosti se formira jedan potomak. Za jedinku populacije nove generacije se bira roditelj ili potomak,ovisno koja jedinka je bolja. Ovom strategijom naglašava se da jedan roditelj formira jednog potomka, a za narednu generaciju se bira bolja jedinka od te dvije. ES(1+1) se može posmatrati kao tehnika lokalnog pretraživanja sa probabilističkim pretraživanjem okoline tekućeg rješenja. Rechenberg je formulirao pravilo da udio uspješnih mutacija u ukupnom broju mutacija treba održavati na vrijednosti 1/5. Ukoliko broj uspješnih mutacija postane veći, potrebno je povećati standardnu devijaciju da bi se uspješnost mutacija smanjila. Ukoliko broj uspješnih mutacija postane manji, potrebno je smanjiti standardnu devijaciju da bi se uspješnost mutacija povećala. Iako je ovo pravilo definirano sa ciljem povećanja efikasnosti pretraživanja,za kompleksnije funkcije može dovesti do zapadanja u lokalne ekstremume.

Matematski opis zadatka

U postavci vježbe dat je pseudokod algoritma po kojem bismo trebali napisati implementaciju. U ovom dijelu će biti prikazan algoritam I opis njegovih parametara.

1: Inicijalizacija: $k \leftarrow 0,\ \boldsymbol{x}_p,\ \boldsymbol{\sigma}^k$
2: Izračunati fitness $y \leftarrow f(\boldsymbol{x}_p)$
3: **repeat**
4: $\quad \boldsymbol{x}_o \leftarrow \boldsymbol{x}_p + \boldsymbol{\sigma}^k \boldsymbol{Z}^k$
5: $\quad y_o \leftarrow f(\boldsymbol{x}_o)$
6: $\quad$ **if** $y_o \leq y_p$ **then**
7: $\quad\quad \boldsymbol{x}_p \leftarrow \boldsymbol{x}_o$
$\quad\quad y_p \leftarrow y_o$
8: $\quad$ **end if**
9: $\quad$ Modificirati $\boldsymbol{\sigma}^k$ prema pravilu "1/5"
10: $\quad k \leftarrow k + 1$
11: **until** Ispunjen uslov zaustavljanja

Parametar σ^k u ovom slučaju predstavlja dijagonalnu matricu (dimenzija problemskog prostora) standardnih devijacija, a Z^k predstavlja vektor kolonu slučajno odabranih vrijednosti između -1 i 1 prema Gaussovoj raspodjeli. Pravilo "1/5" podrazumijeva da se parametar σ^k modificira kao

$$\boldsymbol{\sigma}^{k+1} \leftarrow \begin{cases} \boldsymbol{\sigma}^k \cdot a, & \text{ako je } P_S > 1/5, \\ \boldsymbol{\sigma}^k / a, & \text{ako je } P_S < 1/5, \\ \boldsymbol{\sigma}^k, & \text{ako je } P_S = 1/5, \end{cases}$$

gdje P_s predstavlja uspješnost mutacije. Parametar a se bira između 1.1 i 1.5.

Vodič kroz kôd/simulaciju

U kodu je definisana klasa ES sa traženim atributima i metodama. Dodan je konstruktor u kojem se provjerava validnost parametra a, te se parametri postavljaju na odgovarajuće vrijednosti. Unutar ove klase definisane su i tri metode: mutate, step i run. Implementaciju ovih metoda ćemo prikazati u nastavku. Pored ovih, definisana je i funkcija unutar_opsega koja unutar mutate metode provjerava da li problemska varijabla pripada opsegu.

```
def mutate(self, opseg):
    # metoda koja vrši mutaciju na osnovu Gauss-ove raspodjele

    x1 = [500, 500]

    while not unutar_opsega(x1, opseg):
        Z = [truncnorm(a=-1., b=1., scale=1).rvs(size=1)[0], truncnorm(a=-1., b=1., scale=1).rvs(size=1)[0]]
        x1 = self.x + np.matmul(self.sigma0, Z)

    return x1
```

```
def step(self, f, i, opseg):
    # metoda koja izvršava jednu iteraciju algoritma;
    y = f(*self.x)

    x0 = self.mutate(opseg)
    y0 = f(*x0)

    if y0 < y:
        self.x = x0
        self.ps = self.ps + 1

    p = self.ps / float(i + 1)

    if p > 0.2:
        self.sigma0 = np.asarray(self.sigma0) * self.a
    elif p > 0 and p < 0.2:
        self.sigma0 = np.asarray(self.sigma0) / self.a

def run(self, f, opseg):
    # metoda kojom se poziva izvršavanje algoritma

    for i in range(self.MaxIter):
        self.step(f, i, opseg)

    print("Najbolje rjesenje: " + str(f(*self.x)) + " u tacki " + str(self.x) + "\n")
```

Rezultati kôda/simulacije

U ovom dijelu prvog zadatka za provjeru ispravnosti koda iskoristit ćemo postavku drugog zadatka, te ćemo rezultate prikazati u nastavku. U kod su dodane funkcije iz drugog zadatka. Dodana je i funkcija koja iscrtava graf.

U prvom prolazu kroz kod korišteni su sljedeći parametri:

MaxIter = 100

a = 1.3

Početne tačke za svaku testnu funkciju respektivno:

xp = (3, -3)

xp = (3, -2)

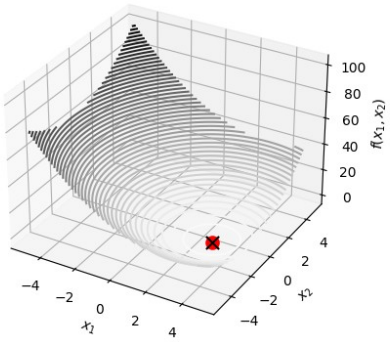
xp = (2, 1)

xp = (8, -8.5)

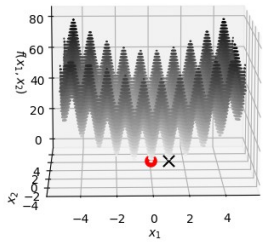
Dobijeni rezultati:

```
samra@samra-laptop:~$ /usr/bin/python3 "/home/samra/Documents/Optimizacija resursa/Lab 8/zadatak1.py
Najbolje rjesenje: 1.3669441537643606e-05 u tacki [ 3.00282935 -1.00237997]

Najbolje rjesenje: 4.9748235261454425 u tacki [ 0.99464346 -1.99017414]

Najbolje rjesenje: -0.7857513346523581 u tacki [ 1.03127188 -0.14567581]

Najbolje rjesenje: -19.18096013896526 u tacki [ 8.10754298 -9.66792031]
```

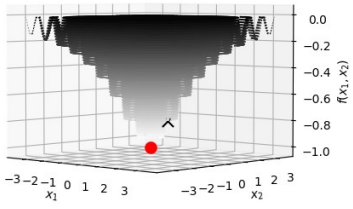
Paraboloid



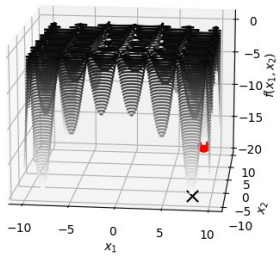
Rastrigin



Drop – Wave



HolderTable



Pokušamo li sada početne tače približiti globalnom minimumu, uzimajući sljedeće tačke za funkcije respektivno:

xp = (3, -2)

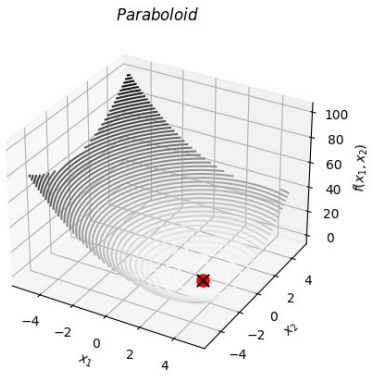
xp = (2, -1)

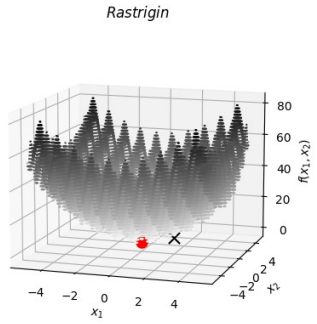
xp = (1, 0)

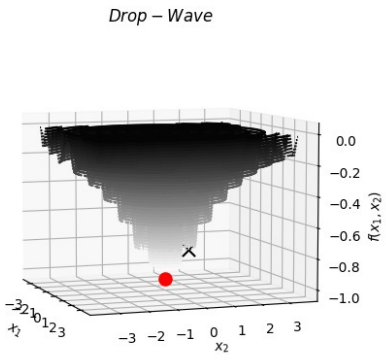
xp = (8, -8.5)

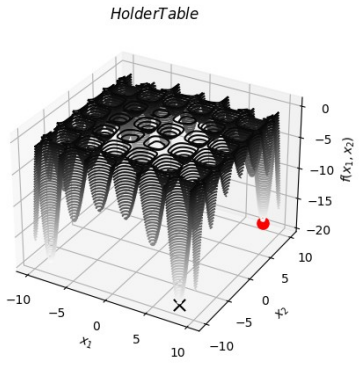
dobijamo sljedeće rezultate:

```
samra@samra-laptop:~$ /usr/bin/python3 "/home/samra/Documents/Optimizacija resursa/lab 8/zadatak1.py"
Najbolje rjesenje: 7.275114467359513e-09 u tacki [ 3.00008039 -1.00002852]

Najbolje rjesenje: 3.981567533323272 u tacki [ 1.98727376e+00 -1.34425226e-03]

Najbolje rjesenje: -0.7857513346520338 u tacki [0.8962554  0.53053668]

Najbolje rjesenje: -19.208502566596277 u tacki [ 8.0550329  -9.66459632]
```



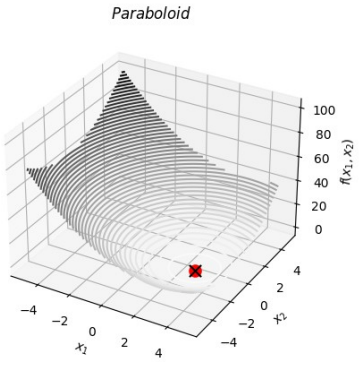


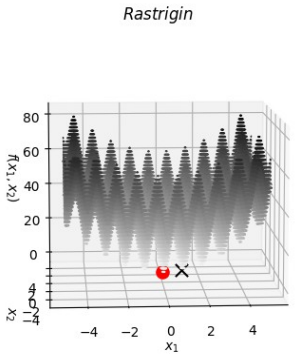


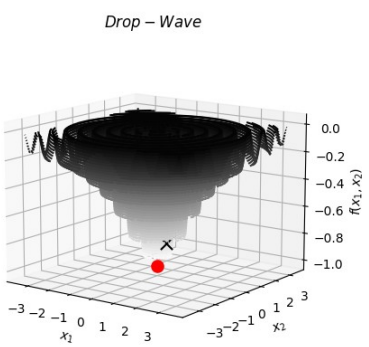


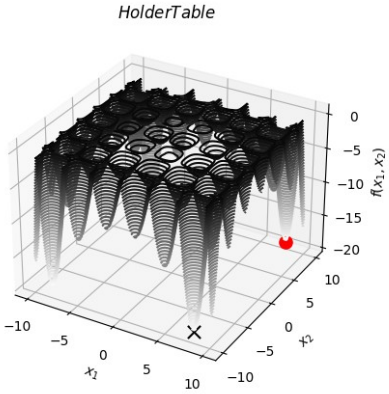
Povećajmo sada broj iteracija na 1000, dok ostali parametri ostaju kao u prethodnom testiranju:

```
samra@samra-laptop:~$ /usr/bin/python3 "/home/samra/Documents/Optimizacija resursa/lab 8/zadatak1.py"
Najbolje rjesenje: 0.0 u tacki [ 3. -1.]

Najbolje rjesenje: 0.9949590570932898 u tacki [9.94958640e-01 1.25273699e-09]

Najbolje rjesenje: -0.7857513346523582 u tacki [ 0.84484635 -0.60907946]

Najbolje rjesenje: -19.208502567884796 u tacki [ 8.05502369 -9.6645904 ]
```









Zaključak

Na osnovu prethodnih rezultata koda možemo zaključiti da kao u prethodno implementiranim algoritmima, pronalazak globalnog optimuma značajno zavisi od početne tačke od koje algoritam počinje. U svakom prolazu kroz algoritam mogli smo primjetiti da paraboloid i holder table funkcije pronalaze svoj globalni minimum, dok preostale dvije funkcije zaostaju. Nakon što smo promijenili vrijednost početnih tačaka, uočili smo poboljšanja kada su u pitanju rastrigin i drop-wave funkcije, medjutim još uvijek nisu bile tako blizu globalnom minimumu. Kako smo testiranjem prethodnih algoritama imali priliku vidjeti da pri dovoljnom broju iteracija algoritam pronalazi bolje rješenje, odlučili smo povećati maksimalan broj iteracija na 1000. Nakon testiranja

primjetne su znatne razlike u rastrigin funkciji, medjutim kada je riječ o drop-wave funkciji, još uvijek se nismo približili globalnom minimumu. Na osnovu ovih podataka, možemo zaključiti da bismo povećanjem broja iteracija i/ili boljim izborom početne tačke, i za dropwave funkciju dobili globalni minimum.

Zadatak 2

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