Operația de mutație în algoritmi genetici

Asist.univ.dr Mihai Tudor

Obiective

- Înțelegerea conceptului de mutație genetică în contextul algoritmilor genetici.
- Implementarea diferitelor tipuri de mutație în Python.
- Vizualizarea efectului mutației asupra populației și diversității soluțiilor.

1 Conceptul de mutaţie

Mutația este o operație aplicată asupra indivizilor dintr-o populație pentru a introduce variație genetică. Aceasta împiedică convergența prematură a algoritmului spre soluții locale. Mutația modifică aleator o parte a cromozomului (de exemplu, schimbă o genă din 0 în 1).

Exemple:

- Într-un cromozom binar: $101010 \rightarrow 101110$
- Într-un șir de caractere: HELLO $\rightarrow$ HEALO
- Într-un vector numeric: $[1.2, 0.8, 3.0] \rightarrow [1.2, 1.0, 3.0]$

2 Tipuri de mutație

- **Bit flip**
  - Schimbă valoarea unei gene binare din 0 în 1 sau invers.
  - Utilizare: pentru cromozomi reprezentați binar.
  - Exemplu: $101010 \rightarrow 111010$
- **Mutația de tip swap**
  - Descriere: interschimbă pozițiile a două gene alese aleator.
  - Utilizare: pentru reprezentări bazate pe permutări (ex: probleme de rutare).
  - Exemplu: $[A, B, C, D] \rightarrow [A, D, C, B]$

- **Mutația de tip scramble**
  - Descriere: selectează un segment aleator din cromozom și amestecă ordinea genelor din acel segment.
  - Utilizare: pentru optimizări combinatorii (unde ordinea elementelor contează).
  - Exemplu: $[1, 2, 3, 4, 5, 6] \rightarrow [1, 4, 3, 2, 5, 6]$
- **Mutația de tip gaussiană**
  - Descriere: adaugă o valoare aleatoare (zgomot gaussian) genelor numerice.
  - Utilizare: pentru cromozomi cu valori reale (optimizări continue).
  - Exemplu: $[1.0, 2.0, 3.0] \rightarrow [1.1, 2.1, 2.9]$

3 Implementare

1. Mutație binară (bit flip)

```python
from random import random

def mutatie_bit_flip(cromozom, rata=0.1):
    #Aplicam mutatie binara (flip) asupra unui cromozom.
    for i in range(len(cromozom)):
        if random() < rata:
            cromozom[i] = 1 - cromozom[i]
    return individ

# Generam un cromozom binar avand in structura sa 10 gene
cromozom=[]
for i in range(10):
   gena=randint(0,1) #generam genele cromozomului binar in
      mod random
   cromozom.append(gena)
print("Cromozom initial:", cromozom)
print("Cromozom ce prezinta mutatie:",mutatie_bit_flip(
   cromozom, rata=0.2))
```

2. Mutație de permutare (swap mutation)

```python
from random import random, randint

def mutatie_swap(cromozom, rata=0.2):
    #Schimbam pozitiile a doua gene cu o anumita
       probabilitate.
    if random() < rata:
        #selectam doua pozitii random pentru genele din
           cromozom
        i = randint(0, len(cromozom) - 1)
        j = randint(0, len(cromozom) - 1)
```

```
        while j == i:  #ne asiguram ca am selectat pozitii diferite
            j = randint(0, len(cromozom) - 1)
        cromozom[i], cromozom[j] = cromozom[j], cromozom[i]
    return cromozom

# Testam
cromozom = [1, 2, 3, 4, 5]
print("Initial :", cromozom)
print("Mutat   :", mutatie_swap(cromozom,0.1))
```

3. Mutație de amestecare (scramble mutation)

```
from random import random, shuffle, randint
def mutatie_scramble(cromozom, rata=0.2):
    #selectam un segement aleator din cromozom si "amestecam" genele
    if random() < rata:
    #alegem doua pozitii random din cromozom pentru a putea selecta segmentul
        start = randint(0, len(cromozom) - 2)
        end = randint(start + 1, len(cromozom) - 1)
        subset = cromozom[start:end]
        #amestecam genele din segmentul selectat aleator
        shuffle(subset)
        #atasam segmentul cu genele "amestecate" la cromozomul initial
        cromozom[start:end] = subset
    return cromozom

# Testam
cromozom = [1, 2, 3, 4, 5, 6]
print("Initial :", cromozom)
print("Mutat   :", mutatie_scramble(cromozom,0.1))
```

4. Mutaţia numerică (gaussiană)

```
from random import random, gauss

def mutatie_gaussiana(individ, rata=0.1, sigma=0.1):
    #vom adauga zgomot gaussian la unele genele ale cromozomului selectate random
    for i in range(len(cromozom)):
        if random() < rata:
            cromozom[i] += gauss(0, sigma)
    return cromozom

# Test
cromozom = [0.5, 1.0, 2.5, 3.0]
print("Initial :", cromozom)
print("Mutat   :", mutatie_gaussiana(cromozom))
```

Observaţie (Distribuţia gaussiană) Dacă o genă are o valoare numerică x, atunci mutația gaussiană produce o valoare nouă x':

$$x' = x + \mathcal{N}(0, \sigma^2)$$

unde:

$$\mathcal{N}(0, \sigma^2)$$

este o variabilă aleatoare gaussiană cu:

$$\mu = 0 \quad \text{(centrată în zero)}$$

$$\sigma^2 \text{ – varianța (controlează „amplitudinea” variației)}$$

$$\sigma \text{ – abaterea standard, controlează cât de puternică este mutația.}$$

Cu alte cuvinte, valoarea se modifică ușor în jurul valorii originale, cu o probabilitate mare pentru schimbări mici și o probabilitate mică pentru schimbări mari.

4 Aplicaţii

Folosind biblioteca Matplotlib redaţi efectul mutaţie gaussiene asupra unui cromozom printr-un grafic sugestiv.

Implementare

```
from random import random, gauss
from matplotlib.pyplot import*

def mutatie_gaussiana(individ, rata=0.1, sigma=0.1):
    #vom adauga zgomot gaussian la unele genele ale cromozomului
       selectate random
    for i in range(len(cromozom)):
        if random() < rata:
            cromozom[i] += gauss(0, sigma)
    return cromozom

# Cromozom numeric (10 gene)
cromozom = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
cromozom_mutat = mutatie_gaussiana(cromozom.copy(), rata=0.8,
   sigma=0.5)

# Afisare comparativa
print("Valori initiale :", cromozom)
print("Dupa mutatie    :", [round(x, 2) for x in cromozom_mutat])

# Vizualizare
figure(figsize=(8,5))
scatter(range(len(cromozom)), cromozom, color='blue', label='
   Initial', s=80)
scatter(range(len(cromozom_mutat)), cromozom_mutat, color='red',
   label='Mutat', s=80)
```

```
xlabel("Index gena")
ylabel("Valoare")
title("Efectul mutatiei gaussiane")
legend()
show()
```