Operatori Genetici

Asist.univ.dr Mihai Tudor

Obiective

- Implementarea mecanismului de selecţie.
- Implemenatrea mecanismului de crossover.

1 Mecansimul de selecţie

Selecția este procesul prin care cromozomii cei mai „buni” (cu fitness mai mare) sunt favorizați pentru a produce descendenți în generațiile următoare.

Rolul selecţiei:

- să păstreze cromozomii performanți;
- se menține și diversitatea populației, permițând uneori și indivizilor mai slabi să participe la procesul de selecctie

Tipuri de selecţie:

- Roulette Wheel (selecţia proporţională) - Probabilitatea de a fi selectat este proporțională cu fitness-ul
  - Avantaje: Ușor de înțeles, menține diversitate;
  - Dezavantaje: Sensibil la valori extreme
- Tournament - Se aleg aleator un număr mic de indivizi și se selectează cel mai bun
  - Avantaje: Simplu, controlabil;
  - Dezavantaje: Poate reduce diversitatea
- Elitism - Se păstrează automat cei mai buni indivizi din generație
  - Avantaje: Nu se pierd soluțiile bune
  - Dezavantaje: Poate duce la convergență rapidă

Implementare

În vederea implementării vom avea ca exemplu următoarea populaţie de cromozomi.

Cromozm	Codificare binară	Fitness	Probabilitate selecție
A	10101	25	$25\ /(25 + 15 + 30 + 5) = 0.33$
B	01011	15	$15\ /\ (25 + 15 + 30 + 5) = 0.20$
C	11100	30	$30\ /\ (25 + 15 + 30 + 5) = 0.40$
D	00010	5	$5\ /\ (25 + 15 + 30 + 5) = 0.07$

1. Selecția Roulette Wheel

   - Se generează un număr aleator între 0 și suma totală a fitness-urilor.
   - Se „parcurge” roata ruletei, adunând fitness-ul indivizilor până se depășește valoarea aleatorie.
   - Individul ales este cel corespunzător acelui segment.

```python
from random import*

# Populatia si fitness-ul fiecarui individ
population = ['10101', '01011', '11100', '00010']
fitness_values = [25, 15, 30, 5]

#  Calculam suma totala a fitness-urilor
total_fitness = sum(fitness_values)
print("Suma fitness-urilor:", total_fitness)

# Simulam selectia prin ruleta de 10 ori
selected = []

for i in range(10):
    # Alegem un numar aleator intre 0 si totalul fitness-urilor
    pick = uniform(0, total_fitness)
    current = 0 #suma acumulata a fitness-urilor, creste pas cu pas.

    #  Parcurgem populatia si adunam fitness-urile
    for j in range(len(population)):
        current += fitness_values[j]
        if current > pick:
            selected.append(population[j])
            break  # oprim bucla individul a fost selectat

print("Indivizi selectati (prin ruleta):")
print(selected)
```

2. Selecţia Turneu (Tournament)

   - Se aleg aleator k indivizi (de exemplu 3).
   - Se selectează cel cu cel mai mare fitness.
   - Dacă k crește, selecția devine mai agresivă (favorizează cei mai buni).

```python
from random import*

population = ['10101', '01011', '11100', '00010']
fitness_values = [25, 15, 30, 5]

selected = []
k = 3  # dimensiunea turneului

for i in range(10):
    # Alegem aleator k indivizi din populatie
    chosen = []  # lista celor alesi pentru turneu
    while len(chosen) < k:
        index = randint(0, len(population) - 1)  # alegem o pozitie aleatorie
        if index not in chosen:  # ne asiguram ca nu alegem acelasi individ de doua ori
            chosen.append(index)

    # Determinam care dintre cei alesi are cel mai mare fitness
    best = chosen[0]
    for i in chosen:
        if fitness_values[i] > fitness_values[best]:
            best = i

    selected.append(population[best])

print("Indivizi selectati (prin turneu):")
print(selected)
```

Aplicaţii

1. Implementați selecția „ruletă” astfel încât să returneze o nouă populație de aceeași dimensiune.
2. Experimentați cu diferite valori ale lui k în selecția turneu (ex: k=2, 5). Ce concluzie obţinem?
3. Exercițiul 3: Comparați distribuția indivizilor selectați pentru cele două metode — care favorizează mai mult indivizii buni?

2 Mecansimul de crossover

Crossover-ul combină informația genetică a doi cromozomi pentru a produce unul sau mai mulți descendenţi.

Scopul mecanismului de crossover: exploatare — păstrăm trăsăturile bune de la cromozomii părinți și creăm variante noi pentru a explora cât mai multe regiuni din spaţiul soluţiilor optime.

Tipuri de crossover:

- One-point crossover (pentru cromozomi binari) - Se alege un punct de tăiere pe cromozomi. Părțile din stânga punctului vin de la un părinte, cele din dreapta de la celălalt.
  - Avantaje: păstrează structuri (subșiruri) din cromozomii părinți;
  - Dezavantaje: poate rupe combinații bune dacă punctul e ales prost.
- Two-point crossover (pentru cromozomi binari)-Se aleg două puncte; segmentul dintre ele se schimbă între cromozomii părinți.
  - Avantaje: Combină mai mult material genetic; mai variat decât one-point.
  - Dezavantaje: Poate rupe structuri mari utile.
- Uniform crossover (pentru cromozomi binari) - Fiecare genă este aleasă aleator din părintele 1 sau părintele 2 .
  - Avantaje: Produce cromozomi copii foarte diverși; nu depinde de poziția genelor.
  - Dezavantaje: Poate distruge complet structuri bune.
- Arithmetic crossover (pentru cromozomi numerici) - cromozomul copil este o combinație liniară convexă între părinți: $child = \alpha \cdot 1 + (1 - \alpha) \cdot 2$, unde $\alpha \in [0, 1]$.
  - Avantaje: potrivit pentru optimizare continuă; menține valori medii.
  - Dezavantaje: nu păstrează valori exacte din părinți; risc de convergență prematură.

Implementare

1. One-point crossover

```
from random import*

parent1 = "1011001"
parent2 = "0100110"

# alegem un punct de crossover (intre 1 si len-1)
point = randint(1, len(parent1)-1)

# cromozomii "copii" se formeaza prin combinarea segmentelor
child1 = parent1[:point] + parent2[point:]
```

```
child2 = parent2[:point] + parent1[point:]

print("p1", parent1)
print("p2", parent2)
print("point", point)
print("c1", child1)
print("c2", child2)
```

2. Two-point crossover

```
from random import*

p1 = "1011001"
p2 = "0100110"
a = randint(0, len(p1)-2)
b = randint(a+1, len(p1)-1)

c1 = p1[:a] + p2[a:b] + p1[b:]
c2 = p2[:a] + p1[a:b] + p2[b:]

print("a,b:", a, b)
print("c1", c1)
print("c2", c2)
```

3. Uniform crossover

```
from random import randint

p1 = "1011001"
p2 = "0100110"
child = ""

for i in range(len(p1)):
    if random() < 0.5:
        child += p1[i]
    else:
        child += p2[i]

print("p1", p1)
print("p2", p2)
print("child", child)
```

4. Arithmetic crossover

```
from random import*

p1 = [1.0, 2.0, 3.0]
p2 = [4.0, 5.0, 6.0]

alpha = random.random()  # generam alpha intre 0 si 1
child = []

```

```
for i in range(len(p1)):
    child.append(alpha * p1[i] + (1 - alpha) * p2[i])

print("alpha:", alpha)
print("child:", child)
```