Pràctica: Algoritme Genètic en Python

Dra Sundus Zafar

Objectiu de la Sessió

L'objectiu d'aquesta sessió és implementar un Algoritme Genètic (AG) en Python per resoldre un problema d'optimització simple. Aprendràs els components d'un AG i com escriure'n el codi, així com analitzar i ajustar-lo per millorar els resultats.

Descripció de l'Algoritme Genètic

L'Algoritme Genètic és una tècnica d'optimització inspirada en la selecció natural. Els components principals d'un AG són:

- **Població inicial**: Un conjunt de solucions possibles (cromosomes).
- **Funció de fitness**: Mesura la qualitat d'una solució.
- **Selecció**: Triar les millors solucions per generar-ne de noves.
- **Creuament**: Combinació de dues solucions per crear-ne noves.
- **Mutació**: Introduir canvis aleatoris per mantenir la diversitat.
- **Condicions d'aturada**: Quan es compleixi un criteri específic (nombre de generacions, millora del fitness, etc.).

Exercici: Implementació d'un Algoritme Genètic per al Problema de la Paraula ÇAT"

En aquest exercici, implementarem un AG per trobar la paraula correcta ÇAT"a partir de lletres aleatòries. Les solucions inicials seran cadenes de caràcters aleatòries.

Pas 1: Inicialització de la Població

Començarem creant una població inicial de cadenes de caràcters aleatòries.

```
import random
import string

# Paràmetres
tamany_poblacio = 10
longitud_paraula = 3
lletra_objectiu = "CAT"

# Funció per generar una cadena aleatòria
def generar_cadena():
return ''.join(random.choice(string.ascii_uppercase) for _ in range(longitud_paraula))

# Generar població inicial
poblacio = [generar_cadena() for _ in range(tamany_poblacio)]
print("Població inicial:", poblacio)
```

Pas 2: Funció de Fitness

La funció de fitness mesura la qualitat de cada solució comparant-la amb la paraula objectiu. La qualitat es basa en el nombre de caràcters que coincideixen amb ÇAT".

```
# Funció de fitness
def calcular_fitness(cadena):
return sum([1 if cadena[i] == lletra_objectiu[i] else 0 for i in range(len(cadena))])

# Exemple de càlcul de fitness
fitness = [calcular_fitness(cadena) for cadena in poblacio]
print("Fitness de la població:", fitness)
```

Pas 3: Selecció

La selecció tria les millors solucions basades en la seva fitness. Podem utilitzar un mètode de selecció per torneig o ruleta per triar els pares.

```
# Selecció per torneig
def seleccio(poblacio, fitness, k=3):
pares = []
for _ in range(len(poblacio) // 2):
torneig = random.sample(range(len(poblacio)), k)
guanyador = max(torneig, key=lambda i: fitness[i])
pares.append(poblacio[guanyador])
return pares
```

```
pares = seleccio(poblacio, fitness)
print("Pares seleccionats:", pares)
```

Pas 4: Creuament (Crossover)

El creuament combina dues solucions per crear una nova. Aquesta funció fa servir un crossover de punt simple.

```
# Creuament
def creuament(pare1, pare2):
punt = random.randint(1, longitud_paraula - 1)
fill1 = pare1[:punt] + pare2[punt:]
fill2 = pare2[:punt] + pare1[punt:]
return fill1, fill2

# Exemple de creuament
fill1, fill2 = creuament("BAT", "CAT")
print("Fills després de creuament:", fill1, fill2)
```

Pas 5: Mutació

La mutació canvia aleatòriament un caràcter de la cadena per evitar que l'algoritme es quedi atrapat en solucions locals.

```
# Mutació
def mutacio(cadena, taxa_mutacio=0.1):
cadena_llista = list(cadena)
for i in range(len(cadena)):
if random.random() < taxa_mutacio:
cadena_llista[i] = random.choice(string.ascii_uppercase)
return ''.join(cadena_llista)

# Exemple de mutació
cadena_mutada = mutacio("CAT")
print("Cadena mutada:", cadena_mutada)
```

Pas 6: Crear la Nova Població

Després de seleccionar, fer el creuament i la mutació, hem de crear la nova població.

```
# Crear nova població
def nova_poblacio(poblacio, fitness):
pares = seleccio(poblacio, fitness)
```

```
fills = []
for pare1, pare2 in zip(pares[::2], pares[1::2]):
fill1, fill2 = creuament(pare1, pare2)
fills.extend([mutacio(fill1), mutacio(fill2)])
return fills

# Nova població
poblacio_nova = nova_poblacio(poblacio, fitness)
print("Nova població:", poblacio_nova)
```

Problemas

1. **Objectiu de Paraula Diferent:**

   Modifica l'algoritme perquè trobi una altra paraula objectiu (per exemple, "HELLO"). Quins efectes té sobre el rendiment de l'algoritme? Necessites canviar algun paràmetre (com la mida de la població, taxa de mutació, etc.) per aconseguir bons resultats?

2. **Augment de la Mida de la Població:** Modifica la mida de la població (per exemple, augmentant-la a 50 o 100 individus) i observa com això afecta la convergència i el temps d'execució. Quin impacte té una població més gran en la qualitat de la solució final?

3. **Addició de Paràmetres Adicionals:** Modifica el codi per afegir una nova funció de fitness que penalitzi les solucions si tenen més d'un caràcter repetit. Com afecta això a la cerca de la solució?

4. **Població Inicial Aleatòria:** Actualment, la població inicial es genera aleatòriament. Què passaria si inicialitzem una part de la població amb cadenes de caràcters que ja continguin alguna lletra de la paraula ÇAT"? Prova-ho i observa si l'algoritme convergeix més ràpidament.

5. **Càlcul de Resultats:** Una vegada que l'algoritme troba la solució (ÇAT"), implementa un sistema per fer un seguiment de l'evolució de la fitness al llarg de les generacions. Com canvia la fitness de la població a mesura que les generacions avancen? Mostra els resultats en un gràfic.

Conclusió

Aquest exercici t'ha proporcionat una comprensió pràctica de com els Algoritmes Genètics poden ser utilitzats per resoldre problemes d'optimització. Recorda que la implementació i els resultats poden variar segons els paràmetres, com la mida de la població, la taxa de mutació, i el nombre de generacions. Experimenta amb aquests paràmetres per veure com influeixen en el procés d'optimització.