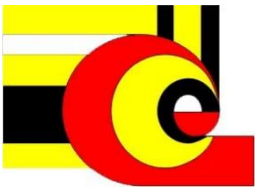


Instituto Politécnico Nacional
Centro de Investigación en Computación

**Actividad 21:**
Operadores de selección en algoritmos genéticos

**Asignatura:**
Metaheurísticas

**Profesor:**
Dra. Yenny Villuendas Rey

**Alumnos:**
Miguel Angel Soto Hernandez
César Macías Sánchez

**28 de Abril del 2022**

1. **Enuncie las ventajas y desventajas de los Algoritmos Genéticos**

**Ventajas:**

- Buscan en una población y no en una muestra.
- Soporta optimización multiobjetivo.
- Utiliza reglas de transición probabilísticas en lugar de las no determinísticas.
- Son buenos para entornos con ruido.
- Son estocásticos.

**Desventajas:**

- Su implementación puede variar mucho.
- Requieren poca información sobre el problema, pero diseñar una función objetivo, obtener operadores y su representación puede tornarse muy difícil.
- Son computacionalmente caros.

2. **Diga las diferencias entre fenotipo y genotipo.**

El genotipo es el conjunto de los genes y la información genética que conforman a un individuo de cualquier especie y se transmite de generación en generación, mientras que el fenotipo es la expresión en forma física de las características de un individuo de cualquier especie. Ejemplo:

- El genotipo es el conjunto de genes del genoma de un individuo.
- El fenotipo es el resultado de la interacción entre genotipo y ambiente. Este se puede apreciar en la forma, color, complexión o alguna combinación de estos en un individuo.

3. **Enuncie las semejanzas y las diferencias en los modelos generacional y estacionario.**

En los modelos generacionales se sustituye por completo la población tras cada iteración. Por otro lado, los modelos estacionarios en cada iteración solo se eligen dos padres de la población a los cuales se les aplicarán los operadores genéticos, y en la población inicial solo se reemplazan uno o dos cromosomas.

4. **Explique el funcionamiento del operador de selección por ruleta.**

A cada uno de los individuos de la población se le asigna una parte proporcional a su ajuste de una ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente la población está ordenada en base al ajuste por lo que las porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo basta con generar un número aleatorio del intervalo [0, 1] y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido.

5. **Explique el funcionamiento del operador de selección por muestreo aleatorio universal.**

La asignación de partes a los individuos se realiza de forma similar al operador de selección por ruleta. Las variaciones se presentan en que ahora se tienen tantas flechas como individuos con un ángulo entre ellas de $\frac{2\pi}{\text{tamaño de la población}}$. Estas flechas se mueven aleatoriamente a cada momento definido por: $\text{aleatorio} * \frac{2\pi}{\text{tamaño de la población}}$. La selección de individuos se realiza en paralelo.

6. **Explique el funcionamiento del operador de selección por torneo.**

La idea principal de este método consiste en realizar la selección en base a comparaciones directas entre individuos. Existen dos versiones de selección mediante torneo:

- **Determinística**: se selecciona al azar un número $p$ de individuos (generalmente se escoge $p = 2$). De entre los individuos seleccionados se selecciona el más apto para pasarlo a la siguiente generación.

- **Probabilística**: aquí únicamente se diferencia en el paso de selección del ganador del torneo. En vez de escoger siempre el mejor se genera un número aleatorio del intervalo [0, 1], si es mayor que un parámetro $p$ (fijado para todo el proceso evolutivo) se escoge el individuo más alto y en caso contrario el menos apto. Generalmente $p$ toma valores en el rango $0.5 < p \leq 1$.
  Variando el número de individuos que participan en cada torneo se puede modificar la presión de selección. Cuando participan muchos individuos en cada torneo, la presión de selección es elevada y los peores individuos apenas tienen oportunidades de reproducción. Un caso particular es el elitismo global. Se trata de un torneo en el que participan todos los individuos de la población con lo cual la selección se vuelve totalmente determinística. Cuando el tamaño del torneo es reducido, la presión de selección disminuye y los peores individuos tienen más oportunidades de ser seleccionados.

7. **Explique el funcionamiento del operador de selección proporcional.**

Cada individuo tiene una probabilidad de ser seleccionado como padre que es proporcional al valor de su función objetivo, pero no es certero.

8. **Explique el funcionamiento del operador de selección por emparejado variado inverso (NAM).**

Un padre es seleccionado aleatoriamente, para elegir el otro padre se seleccionan $n$ padres de la población y se elige el más lejano al que se seleccionó inicialmente. Este operador esta orientado a generar diversidad.

9. **Proponga las estructuras de datos necesarias para la implementación de los operadores de selección.**

Población: listas (pueden variar entre binarias, reales o alfanuméricas dependiendo el problema)

10. **Implemente tres de los operadores mencionados.**

## Centro de Investicgación en Computación, IPN

# Metaheurísticas

## Algoritmos Genéticos: Selección

Cesar Macias Sanchez

Miguel Angel Soto Hernandez

### Importaciones necesarias

```
import numpy as np
import matplotlib.pyplot as plt
import math
import random
```

### Petición de datos

```
# Definiendo parámetros
tamano_poblacion = 50
numero_generaciones = 100
peso_maximo = 100
padres = 2

# Definir el tamaño del genoma
tamano_genoma = int(input('Tamaño del genoma: '))
tamano_genoma
```

```
10
```

### Funciones necesarias

```
def generar_pesos_valores():
    pesos = np.random.randint(0, 50, size = tamano_genoma).tolist()
    valores = np.random.randint(0, 100,size = tamano_genoma).tolist()
    return pesos, valores

pesos, valores = generar_pesos_valores()
print(f'Pesos: {pesos}')
print(f'Valores: {valores}')
```

```
Pesos: [1, 10, 7, 23, 6, 16, 39, 18, 10, 33]
Valores: [52, 60, 75, 82, 39, 84, 60, 37, 18, 22]
```

```
def evaluacion(genoma, pesos, valores, peso_maximo):
  total_valores = 0
  total_pesos = 0
```

```
  for indice, objeto in enumerate(genoma):
    if objeto == 1:
      total_valores += valores[indice]
      total_pesos += pesos[indice]

  if total_pesos > peso_maximo:
    return 0, total_pesos
  else:
    return total_valores, total_pesos
```

## 1. Generar población

```
# Inicializar la población
poblacion = []
for i in range(tamano_poblacion):
    poblacion.append(np.random.randint(2, size =
tamano_genoma).tolist())

poblacion[:4]
```

```
[[0, 0, 0, 1, 0, 0, 1, 0, 1, 1],
 [0, 0, 1, 0, 1, 0, 1, 0, 0, 1],
 [0, 1, 1, 0, 1, 1, 0, 1, 1, 0],
 [0, 1, 0, 0, 1, 1, 1, 0, 0, 0]]
```

## 2. Selecciones

### Parámetros generales

```
# Generar ajustes y pesos de la población
ajustes_poblacion = []
pesos_poblacion = []

for genoma in poblacion:
    ajuste_genoma, peso_genoma = evaluacion(genoma, pesos, valores,
peso_maximo)
    ajustes_poblacion.append(ajuste_genoma)
    pesos_poblacion.append(peso_genoma)

print(f'Ajustes poblacion: {len(ajustes_poblacion)}')
print(f'Pesos poblacion: {len(pesos_poblacion)}')
```

```
Ajustes poblacion: 50
Pesos poblacion: 50
```

```
total_ajustes = sum(ajustes_poblacion)
total_ajustes
```

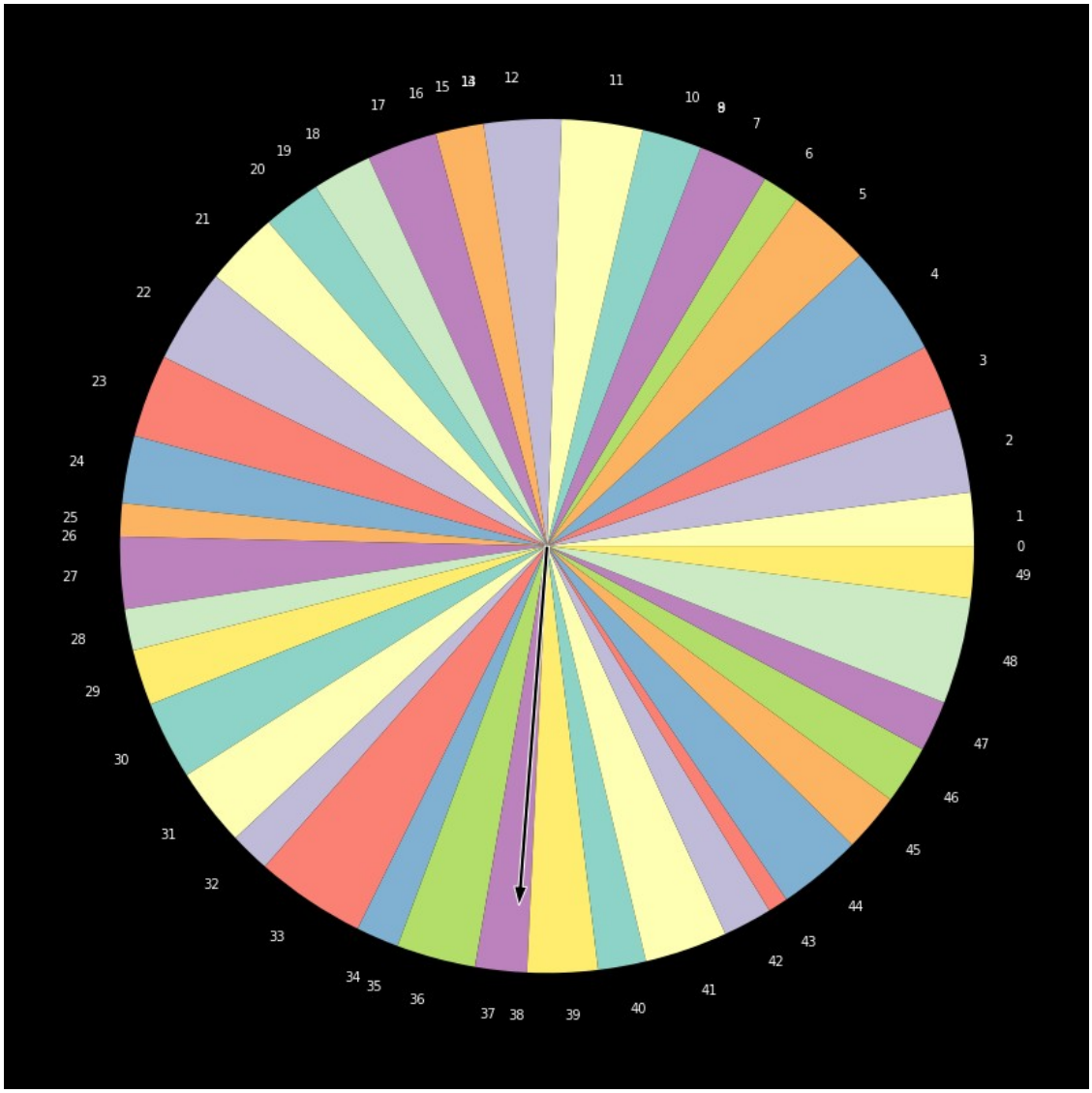
```
9754
```

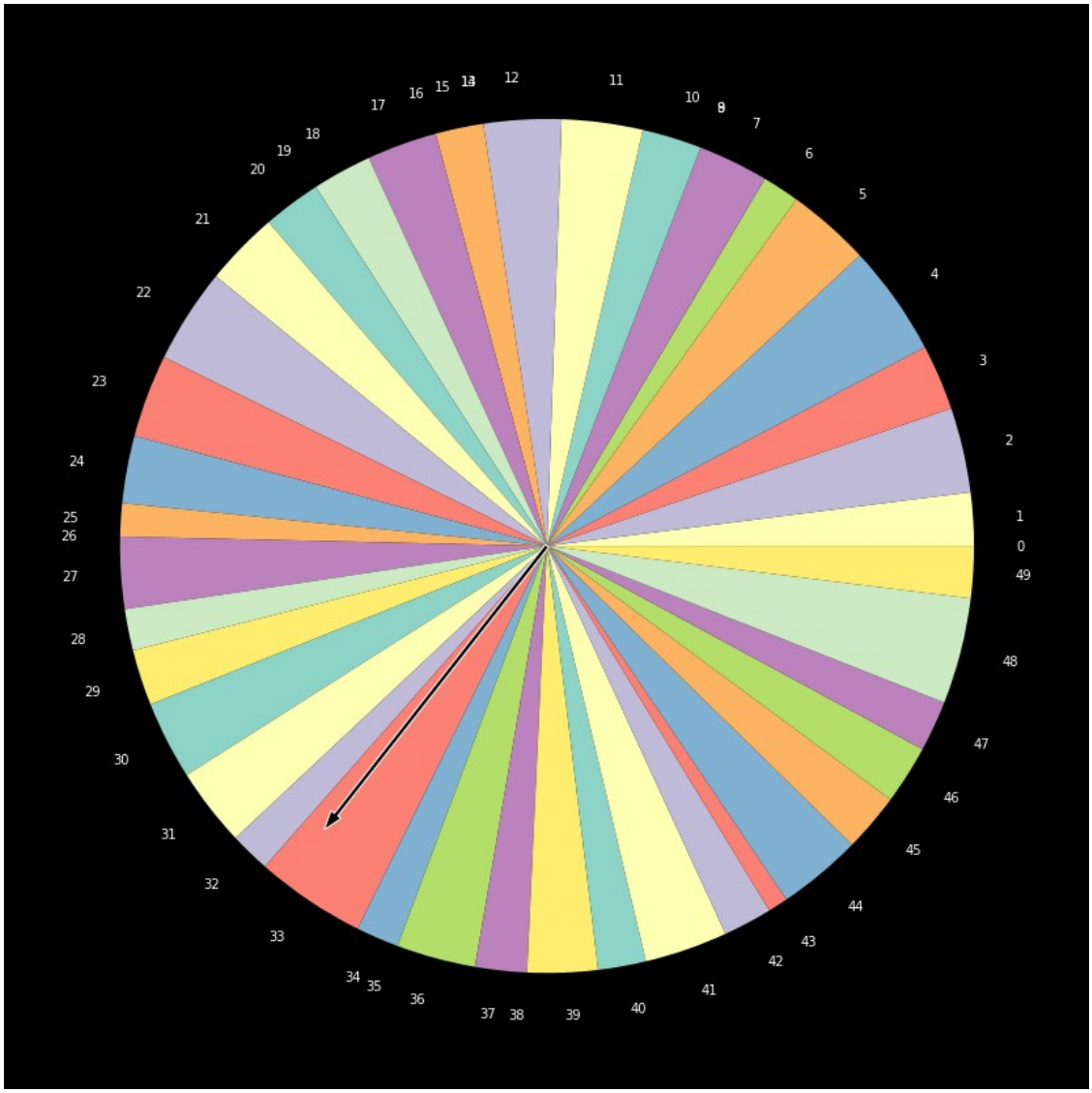
```
porcentajes = []
for ajuste in ajustes_poblacion:
    porcentaje = round((ajuste / total_ajustes) * (2 * math.pi), 4)
```

```
    porcentajes.append(porcentaje)

porcentajes[:4]
```

### Selección por ruleta

```
def flecha():
    return random.random() * (2 * math.pi)

def seleccion_ruleta(porcentajes):
    i = 0
    porcentaje = 0
    flecha_aleatoria = flecha()

    while porcentaje <= flecha_aleatoria:
        porcentaje += porcentajes[i]
        indice = i
        i += 1

    plt.style.use('dark_background')
    plt.figure(figsize = (15, 15))
    plt.pie(porcentajes, labels = [i for i in
range(len(porcentajes))], )
    plt.arrow(0, 0, 0.8 * math.cos(flecha_aleatoria), 0.8 *
math.sin(flecha_aleatoria),
                                                        width = .01,
facecolor = 'black')
    plt.show()

    return indice

padres_ruleta = []

for i in range(padres):
    padres_ruleta.append(poblacion[seleccion_ruleta(porcentajes)])
```


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49




```
print(f'Padres ruleta: {padres_ruleta}')
```

```
Padres ruleta: [[1, 0, 0, 0, 0, 1, 0, 1, 1, 0], [1, 1, 1, 0, 1, 1, 1,
1, 0, 0]]
```

### Selección por torneo

```
k = 5

def seleccion_torneo(k):
    ajustes_torneo = []
    indices = np.random.randint(0, len(poblacion) - 1, size =
k).tolist()

    for indice in indices:
        ajustes_torneo.append(ajustes_poblacion[indice])

    seleccionado_torneo =
```

```
indices[ajustes_torneo.index(max(ajustes_torneo))]
    return seleccionado_torneo

padres_torneo = []

for i in range(padres):
    padres_torneo.append(poblacion[seleccion_torneo(k)])

print(f'Padres torneo: {padres_torneo}')
```

```
Padres torneo: [[1, 1, 1, 0, 0, 1, 0, 0, 1, 1], [1, 0, 1, 1, 1, 1, 0,
0, 1, 0]]
```

### Selección por emparejado variado inverso (NAM)

```
# Muestra aleatoria de padres
n = 10

def seleccion_nam(n):
    ajustes_nam = []
    distancias = []
    padre_1 = random.randint(0, len(poblacion) - 1)
    ajuste_padre_1 = ajustes_poblacion[padre_1]
    muestra = np.random.randint(0, len(poblacion) - 1, size =
n).tolist()

    for indice in muestra:
        ajustes_nam.append(ajustes_poblacion[indice])

    for ajuste in ajustes_nam:
        distancias.append(int(math.fabs(ajuste_padre_1 - ajuste)))

    padre_2 = muestra[distancias.index(max(distancias))]

    return [poblacion[padre_1], poblacion[padre_2]]

padres_nam = seleccion_nam(n)
print(f'Padres NAM: {padres_nam}')
```

```
Padres NAM: [[0, 0, 0, 1, 1, 1, 1, 0, 1, 1], [1, 1, 1, 1, 1, 1, 0, 0,
1, 0]]
```