





Estimados estudiantes

Estoy adjuntando un laboratorio estilo tutorial con las siguientes carpetas

-  [1]Representacion
-  [2]Seleccion(Ruleta-Torneo)
-  [3]Reproduccion(Cruzamiento-Mutacion)
-  [4]AG-Completo

Los que deseen ver paso a paso el desarrollo de un AG, pueden revisar las carpetas en orden de numeración:

- La carpeta [1] contiene el código para representar una población con individuos tipo *string*
- La carpeta [2] contiene el código de la carpeta anterior más los procesos de selección tipo Ruleta y Torneo
- La carpeta [3] contiene el código de la carpeta anterior más los procesos de Cruzamiento y Mutación
- La última carpeta [4] contiene todo el desarrollo de un AG, para el problema presentado en la diapositiva nro 69 de la unidad 2.

Los que deseen ver el desarrollo de un algoritmo genético completo, pueden ver la carpeta [4]

El problema a resolver en este laboratorio es el siguiente:

- Problema: Buscar una frase (*target string*) a partir de una población de cadenas aleatorias.
- Representación: string, cada cromosoma puede generar n^p posibles frases, donde **n** representa el número de caracteres y **p** el tamaño de la frase

La búsqueda exhaustiva que una frase de 8 caracteres puede generar es de 53^8 . Donde cada carácter puede tomar una de 53 caracteres posibles (26 minúsculas, 26 mayúsculas y un espacio en blanco).

- Fitness: suma 1 a cada coincidencia entre los caracteres del cromosoma con el *target string*

```
fitness = 0
for i in range(len(chromosome)):
    if chromosome[i] == target_string[i]:
        fitness += 1
```

- Selección: Ruleta
- Cruzamiento: 1 punto
- Mutación: bit flip
- Selección de sobrevivientes: por ranking

Pueden ir probando la evolución cambiando los parámetros del problema completo en la carpeta [4]:

```
# Inicializa una poblacion de forma aleatoria
num_individuals = 80
population = init_population(num_individuals, len(target_string), allele_pool)

fitness_fn = matching_characters

ngen = 180 # Nro de generaciones
pmut = 0.8 # tasa de mutación

metodoSeleccion=['roulette','torneo']
size_torneo = int(5*num_individuals/100) #tamaño del torneo como porcentaje de la poblacion

crossover=['onepoint','uniform']
mutation=['position','onepoint']

# llama al algoritmo genetico
best_ind, bestfitness = genetic_algorithm(population, fitness_fn, target_string, ngen, metodoSeleccion[0],
                                         size_torneo, pmut, crossover[0], mutation[0])
```