Algoritmo Genético Binario

Morales Zepeda Ivan Yutlanih

Código: 215467762

Ingeniería en computación

Abordar el problema de las monedas planteado en la actividad no resultó muy complicado debido a que el proceso de definir los individuos y el algoritmo evolutivo fue facilitado por el profesor. Sin embargo, definir la función objetivo es un reto diferente, por lo menos en mi caso.

Comencé por comprender el funcionamiento del algoritmo de la mochila que nos brindó el profesor. Comprendí que la idea era dejar al AG ejecutar el proceso una vez de manera aleatoria una cierta cantidad de veces. De esta forma, se le penalizaría si el resultado no es el más efectivo posible, para obtener mejores resultados a lo largo de las generaciones. En resumen, la función objetivo es un algoritmo con el proceso para solucionar el problema con el adicional de que existe una penalización para individuos cuya respuesta no satisface nuestros criterios de aceptación, evitando que estos pasen a la siguiente generación.

Una vez que comprendí el funcionamiento de ese enfoque, lo que restaba era definir la lógica para el problema de las monedas. Primero, al igual que en el caso de la mochila, identificamos los elementos que fueron seleccionados. Después, sumamos la cantidad del individuo seleccionado a la cantidad total e indicamos que se tomó una moneda. Dentro de este proceso, se verifica si la anterior moneda fue tomada, para evitar la adyacencia. Si existe adyacencia, el valor de la última moneda se añadirá también al valor total de penalización. Finalmente, al valor total se le restará el valor de la penalización, más un 10% del mismo valor. Código:

```python
class Monedas:
    def __init__(self, monedas):
        self._monedas = monedas

    def f(self, cromosoma):
        valorTotal = 0 #Para retornar la cantidad total obtenida
        penalizacion = 0 #La cantidad que penalizaremos si se toman monedas adyacentes
        adyacenteTomada = False #Para si agarró una moneda con monedas adyacentes tomadas

        for i, gen in enumerate(cromosoma): #Recorre el arreglo de monedas tomadas
            if gen: #Si la moneda está tomada
                valorTotal += self._monedas[i] #Suma su vlaor
                #If:
                penalizacion += self._monedas[i] if adyacenteTomada else 0 #Si hay una adyacente tomada, suma a la penalización, caso contrario, no aumenta.
                adyacenteTomada = True #Se define como tomada porque la siguiente es adyacnete.
            else:
                adyacenteTomada = False #No hay adyacencia. Se la moneda de la siguiente vuelta...
        return valorTotal - (penalizacion + (penalizacion * .10)) #Suma de las monedas tomadas - (cantidad acumulada de la penalizacion + 10% cantidad acumulada de la penalización)
```

De esta forma fue abordado el problema. Los resultados obtenidos no eran constantes, tenían una variación, sobre todo al momento de probar con diferentes cantidades de individuos y generaciones. Sin embargo, el resultado no variaba drásticamente.