Ejercicios

Johana Tellez - Michael Caicedo

1. Maximizar la función $f(x) = x\,sen(10\,\pi\,x) + 1$, con $x \in [0,1]$
   - L=8: número de genes del cromosoma (bits de longitud del cromosoma).
   - k: un gen cualquiera del cromosoma (puede tomar valor 0 o 1).
   - K=8: número de cromosomas en la población.
   - M=4: número de generaciones ejecutadas en la simulación.
   - x: valor real representado por un cromosoma, decodificado desde la cadena binaria mediante:

$$x = \frac{DEC}{2^L - 1}, DEC\epsilon[0,255]$$

   - f(x): función de aptitud aplicada en el punto x, en este caso se busca **maximizarla**

Primera generación:

Indice	Bits	Dec	x	f(x)	Aptitud	Prob. Selección
1	01101100	108	0,423529	1,28533	1,28533	0,149377
2	01010101	85	0,333333	0,711325	0,711325	0,082668
3	01000010	66	0,258824	1,248943	1,248943	0,145148
4	00011000	24	0,094118	1,017294	1,017294	0,118227
5	10100011	163	0,639216	1,602879	1,602879	0,186282
6	10001011	139	0,545098	0,461353	0,461353	0,053617
7	00111001	57	0,223529	1,150591	1,150591	0,133718
8	00111000	56	0,219608	1,126884	1,126884	0,130963

Para la selección se usó el **método de ruleta**, cada cromosoma obtiene una probabilidad de ser elegido padre proporcional a su aptitud, que en este caso es directamente el resultado de la función a maximizar. Se aplicó un **cruce de un punto**, se selecciona al azar un punto dentro de los bits del cromosoma y se intercambia la parte restante entre dos padres para formar dos hijos. Se aplicó una **mutación simple de bit**, con una baja probabilidad al azar, se cambia un bit del cromosoma (de 0 a 1 o de 1 a 0).

Resultado obtenido después de la cuarta generación:

$X \approx 0.839 \Rightarrow f(x) \approx 1.803$

Al ejecutar el algoritmo genético con los parámetros dados (L=8, K=8, M=4), se obtuvo como mejor solución:

$$x \approx 0.839, \qquad f(x) \approx 1.80$$

El valor máximo que puede alcanzarse en este problema es aproximadamente:

$$x \approx 0.851, \qquad f(x) \approx 1.85$$

La diferencia se debe a que la población inicial fue pequeña y el número de generaciones bajo, lo que limitó la exploración y diversidad del algoritmo.

Para mejorar los resultados, se recomienda:

- aumentar el número de generaciones,
- incrementar la población inicial,
- y aplicar el elitismo (conservar siempre el mejor cromosoma encontrado).

2. Tome el algoritmo de la dieta y ahora incluya costos. Ahora encuentre una dieta que trate de satisfacer la dieta, pero con un costo mínimo. Este es un ejemplo de AG multi-objetivo con dos funciones objetivo.

   Lo que se modificó fue la definición del problema para pasar de un algoritmo genético mono-objetivo (minimizar solo la diferencia de calorías) a un enfoque multi-objetivo que también considera el costo total de la dieta. Para ello se agregó una columna de costos en la tabla de productos, se almacenaron en una lista, se redefinió el tipo de Fitness como multi-objetivo con dos criterios (calorías y costo), y se adaptó la función de evaluación para devolver ambos valores. Además, se cambió el operador de selección a selNSGA2, que está diseñado para optimizar varios objetivos y construir una frontera de Pareto. De esta forma, el algoritmo no solo busca cumplir la meta calórica, sino también minimizar los gastos, ofreciendo un conjunto de soluciones que equilibran ambos factores.

   Resultado:

	Nombre	Min	Max	Calorias	Gram_Prot	Gram_Grasa	Gram_Carb	Costo	escog_univariada
0	Banano 1u	0	4	89	1	0	23	0.5	0
1	Mandarina 1u	0	4	40	1	0	10	0.4	0
2	Piña 100g	0	7	50	1	0	13	0.7	0
3	Uvas 100g	0	7	76	1	0	17	0.8	0
4	Chocolate 1 bar	0	4	230	3	13	25	1.5	1

   En la tabla, cada fila representa un alimento disponible en la dieta, y cada columna tiene un propósito específico: **Nombre** indica el producto; **Min** y **Max** son los límites mínimo y máximo de unidades que se pueden consumir en la semana; **Calorías** muestra la energía que aporta una unidad del alimento; **Gram_Prot**, **Gram_Grasa** y **Gram_Carb** corresponden a los gramos de proteína, grasa y carbohidratos de ese alimento, respectivamente; **Costo** refleja el precio aproximado de una unidad; y finalmente, **escog_univariada** señala cuántas unidades de ese producto fueron seleccionadas por el algoritmo genético en la solución obtenida.

3. Verdadera democracia. Suponga que usted es el jefe de gobierno y está interesado en que pasen los proyectos de su programa político. Sin embargo, en el congreso conformado por 5 partidos, no es fácil su tránsito, por lo que debe repartir el poder, conformado por ministerios y otras agencias del gobierno, con base en la representación de cada partido. Cada entidad estatal tiene un peso de poder, que es el que se debe distribuir. Suponga que hay 50 curules, distribuya aleatoriamente, con una distribución no informe entre los 5 partidos esas curules. Defina una lista de 50

entidades y asígneles aleatoriamente un peso político de 1 a 100 puntos. Cree una matriz de poder para repartir ese poder, usando AGs.

Para resolver el problema de **“Verdadera democracia”**, se planteó un algoritmo que primero asigna de manera aleatoria y no uniforme las 50 curules del congreso entre los 5 partidos políticos. Luego, se definieron 50 entidades estatales, cada una con un peso de poder asignado aleatoriamente entre 1 y 100, representando la importancia política de cada cargo. Posteriormente, se aplicó un algoritmo genético (AG) para repartir los ministerios y agencias (poder político) entre los partidos, buscando que la distribución resultante guarde proporcionalidad con la representación que ya poseen en curules, lo cual asegura un balance entre legitimidad electoral y gobernabilidad.

En los **resultados**, la primera tabla muestra cómo se distribuyeron las curules entre los 5 partidos, reflejando la fuerza legislativa de cada uno.

```
Distribución inicial de curules (poder en el Congreso):
  Partido  Curules
Partido 1       23
Partido 2        1
Partido 3        5
Partido 4        2
Partido 5       19
```

La segunda tabla presenta las 50 entidades estatales con sus respectivos pesos de poder, generados aleatoriamente.

```
 Lista completa de entidades (50) y sus pesos (aleatorios 1-100)
    Entidad  Peso
  Entidad 1    95
  Entidad 2    97
  Entidad 3    87
  Entidad 4    14
  Entidad 5    10
  Entidad 6     8
  Entidad 7    64
  Entidad 8    62
  Entidad 9    23
 Entidad 10    58
 Entidad 11     2
 Entidad 12     1
 Entidad 13    61
 Entidad 14    82
 Entidad 15     9
 Entidad 16    89
 Entidad 17    14
 Entidad 18    48
 Entidad 19    73
 Entidad 20    31
```

Finalmente, la tabla consolidada expone cómo quedaron asignados tanto los curules como las entidades a cada partido, junto con la proporción respectiva para cada partido, lo que permite visualizar si la distribución fue equitativa y consistente con el objetivo.

```
=== Comparación final de poder y curules ===
   Partido  Curules  Porc_Curules  Poder_Entidades  Porc_Poder
Partido 1       23          46.0             1049   45.867949
Partido 2        1           2.0               47    2.055094
Partido 3        5          10.0              227    9.925667
Partido 4        2           4.0               96    4.197639
Partido 5       19          38.0              868   37.953651
```

Se puede apreciar una cercanía entre las proporciones del número de curules y el poder asignado en las entidades.
Por ejemplo, para el **partido 1**:

$$\frac{23}{50} = 0.46\ (46\%)$$

mientras que su poder en entidades es:

$$\frac{1049}{2287} \approx 0.458\ (45.8\%)$$

Es decir, ambas proporciones son muy cercanas, alrededor del **46%**.