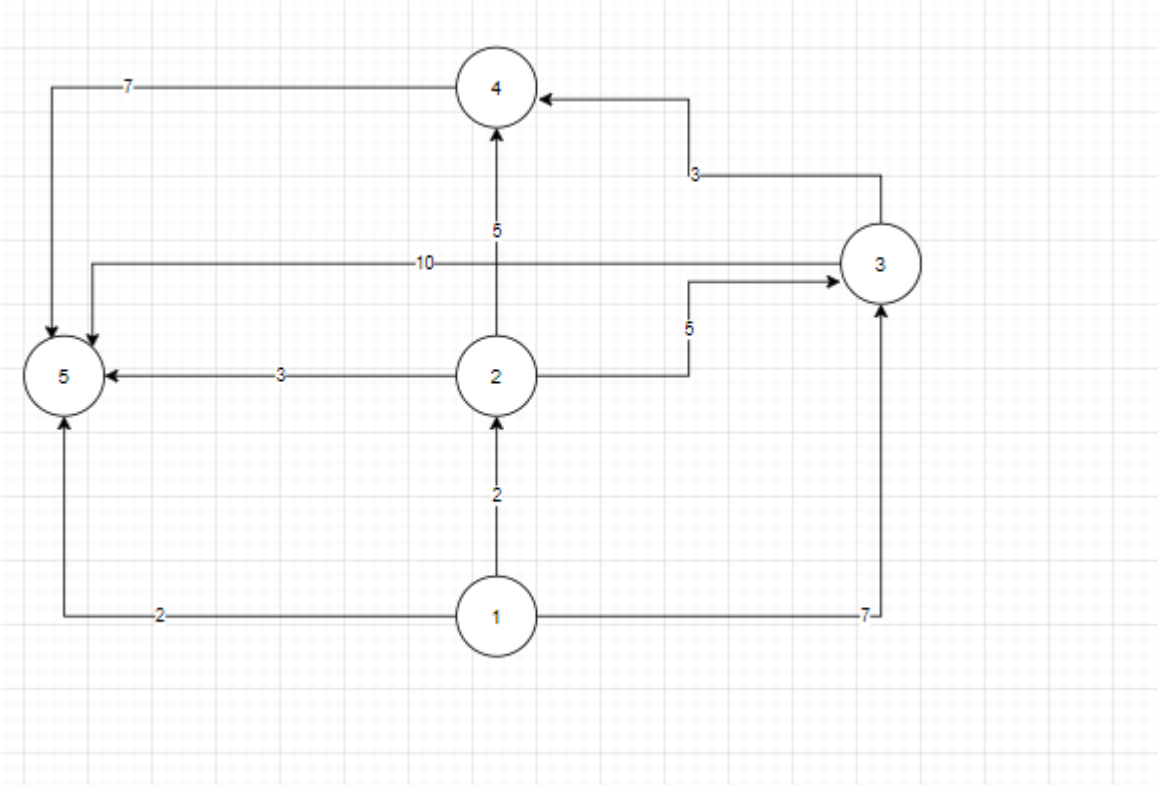
VIAJANTE DE COMERCIO

- Tenemos una red de nodos que pueden ser ciudades o lugares dentro de una misma ciudad.
- Conocemos la distancia entre cada nodo.
- Partimos de un nodo inicial, teniendo que recorrer todos los nodos sin pasar más de una vez por cada nodo, y volver al nodo inicial.



SOLUCIONES AL PROBLEMA DEL VIAJANTE DE COMERCIO

- Algoritmos ACO: Consiste en utilizar un sistema de hormigas para obtener el camino más corto que pasa por todos los nodos una sola vez y vuelve al nodo origen. Para ello, se utilizará una matriz de feromonas para comprobar por cuál camino cruzan un mayor número de

hormigas, siendo de este modo dicho camino el más corto.

Consideramos t0=10 y la siguiente matriz de feromonas:

0	2	7	0	2
2	0	5	5	3
7	5	0	3	10
0	5	3	0	7
2	3	10	7	0

Calculamos la matriz de heurística:

∞	1/2	1/7	0	1/2
1/2	∞	1/5	1/5	1/3
1/7	1/5	∞	1/3	1/10
0	1/5	1/3	∞	1/7
1/2	1/3	1/10	1/7	∞

Vamos a utilizar 4 hormigas, las cuáles salen desde un nodo distinto

Hormigas	Nodo Salida
Hormiga1	Nodo1
Hormiga2	Nodo2

Hormiga3	Nodo3
Hormiga4	Nodo4
Hormiga5	Nodo5

Calculamos el camino de cada hormiga. Para ello, debemos aplicar la siguiente probabilidad:

$$p_k(r,s) = \begin{cases} \dfrac{[\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}{\sum_{u \in J_k(r)} [\tau_{ru}]^{\alpha} \cdot [\eta_{ru}]^{\beta}}, & \text{si } s \in J_k(r) \\ 0, & \text{en otro caso} \end{cases}$$

Para calcular el camino con un mayor número de feromonas, utilizaremos el valor 0,5528.

Calculamos el recorrido de la hormiga1

Nodo a continuación	$P_{1->2}$	$P_{1->3}$	$P_{1->5}$
Nodo3	7/16	1/8	7/16
	$P_{3->2}$	$P_{3->5}$	$P_{3->4}$
Nodo4	6/19	3/19	10/19
	$P_{4->2}$	$P_{4->5}$	
Nodo2	7/12	5/12	
Nodo5			

El camino recorrido por la hormiga1 es: 1->3->4->2->5->1

Calculamos el recorrido de la hormiga 2:

Nodo a continuación	$P_{2->1}$	$P_{2->3}$	$P_{2->4}$	$P_{2->5}$
3	15/37	6/37	6/37	10/37
	$P_{3->1}$	$P_{3->4}$	$P_{3->5}$	
4	30/121	70/121	21/121	
5				
1				

El camino que recorre la hormiga2 es: 2->3->4->5->1->2

Calculamos el camino de la hormiga3:

Nodo a continuación	$P_{3->1}$	$P_{3->2}$	$P_{3->4}$	$P_{3->5}$
4	30/163	42/163	70/163	21/163
	$P_{4->2}$	$P_{4->5}$		
2	7/12	5/12		
	$P_{2->1}$	$P_{2->5}$		
1	3/5	2/5		
5				

El camino que recorre la hormiga 3 es: 3->4->2->1->5->3

Calculamos el camino que recorre la hormiga 4:

Nodo a continuación	$P_{4->2}$	$P_{4->3}$	$P_{4->5}$
3	21/71	35/31	15/71
	$P_{3->1}$	$P_{3->2}$	$P_{3->5}$
2	10/31	14/31	7/31
	$P_{2->1}$	$P_{2->5}$	
1	3/5	2/5	
5			

El camino que recorre la hormiga4 es: 4->3->2->1->5->4

Calculamos el camino que recorre la hormiga 5:

Nodo a continuación	$P_{5->1}$	$P_{5->2}$	$P_{5->3}$	$P_{5->4}$
Nodo2	105/226	35/113	21/226	15/113
	$P_{2->1}$	$P_{2->5}$	$P_{2->4}$	
Nodo1	5/9	2/9	2/9	
Nodo4				
Nodo5				

El camino que recorre la hormiga 5 es: 5->2->1->3->4->5

Calculamos cuál es la mejor hormiga

Hormiga	Fitness	Mejor hormiga
1	20	1
2	19	2
3	22	2
4	19	2-4
5	22	2-4

Las mejores hormigas son la hormiga2 y la hormiga4, con un valor de fitness de 22. Por tanto, los caminos más cortos son [2->3->4->5->1->2] y [4->3->2->1->5->4]

- Enfriamiento simulado: Dada una solución considerada la mejor, obtenemos una nueva solución y comparamos ambas soluciones. Si la nueva solución tiene un valor de fitness inferior a la solución actual, la mejor solución será la nueva solución. En caso contrario, se comprueba si se acepta la nueva solución (la solución se rechaza si el valor de aceptación es superior al valor dado por la ecuación $e^{F(\text{mejor solución actual})-F(\text{nueva solución})/To}$, siendo To la temperatura inicial) y se calcula el valor dado por el tipo de enfriamiento utilizado.

- Heurística de Greedy, el cual consiste en obtener el camino que pasa por todos los nodos una sola vez (volviendo al nodo inicial) y con el menor coste.

Para ello, debemos aplicar los siguientes pasos:
Paso 0: Aplicamos una codificación entera de las posibles soluciones. Por ejemplo, la posible solución [1->2->3->4->5->1]

representa el recorrido de un posible camino por los nodos indicados.

Paso 1: Obtenemos la solución inicial. Para ello, debemos obtener un camino cualquiera, que sea una posible solución al problema.

Paso 2: Obtenemos la solución óptima. Para ello, debemos obtener una nueva posible solución no obtenida del problema. Una vez obtenida dicha solución, calculamos su coste y, si dicho coste es menor a la solución considerada como óptima, entonces la nueva solución será la óptima y repetimos el paso 2.

Este proceso se repetirá mientras exista una nueva posible solución al problema.

- Búsqueda tabú: Para ello, se clasifica un número reciente de movimientos como movimientos tabú, de modo que los movimientos ya realizados no pueden volver a darse.

- Algoritmos genéticos de codificación entera: Para ello, debemos aplicar los siguientes pasos mientras hayan generaciones por producirse:

Paso 0: Generamos la población inicial de manera aleatoria, la cual consiste en un conjunto de individuos formados por cromosomas que representan posibles soluciones al problema. Por ejemplo, un posible individuo de la población inicial sería el camino [1->2->3->4->5->1].
Paso 1: Obtenemos los padres de la nueva generación mediante una selección por torneo, mediante el cual se comparan k individuos de la población actual y se selecciona como padre a los mejores individuos de la población.

Paso 2: En función de una probabilidad de cruce (Pc), realizamos el cruce de 2 padres y obtenemos 2 individuos hijos mediante un operador de cruce de permutación en 2 puntos. Este tipo de cruce consiste en elegir un subconjunto de cada progenitor y cruzarlos preservando el orden y la posición de la mayor cantidad de genes posible del otro, manteniendo la convergencia.

Paso 3: En función de una probabilidad de mutación (Pm), realizamos una mutación de un individuo aplicando una mutación estándar. Dicha mutación consiste en seleccionar al azar 2 vértices del individuo e intercambiar sus posiciones.

Paso 4: Finalmente, nos quedamos con la mejor solución aplicando elitismo, mediante el cual nos quedamos con las mejores poblaciones generadas del total de poblaciones exploradas.

- ¿Búsqueda local Iterativa? Consiste en, dada una solución inicial, se aplica una búsqueda local de dicha solución y se repiten los siguientes pasos mientras no se cumpla una condición de parada (por ejemplo: no existen más caminos por comprobar):

1. Se aplica una modificación de la solución obtenida mediante la búsqueda local.
2. Se aplica una búsqueda local sobre la solución modificada.
3. Se comprueba si la solución obtenida en el paso anterior cumple un criterio de aceptación (por ejemplo: el valor de fitness de la nueva solución es inferior al valor de fitness de la mejor solución actual). En caso de que se cumpla dicho criterio de aceptación, la mejor solución será la nueva solución.

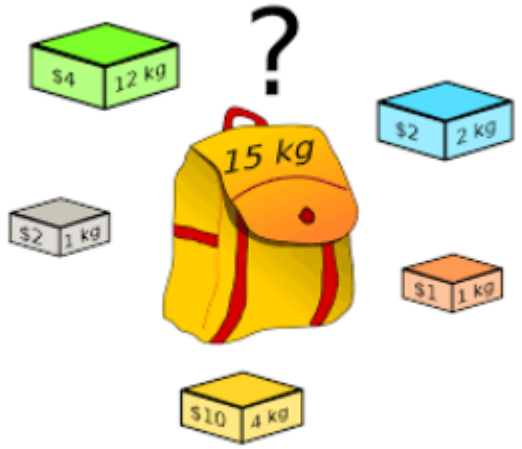
- ¿Búsqueda por entornos variables?
- Método GRASP: Consiste en aplicar los siguientes pasos mientras no se cumpla una condición de parada (en este caso, mientras no queden caminos sin comprobar):

1. Construimos la solución Greedy
2. Aplicamos una búsqueda local sobre la solución Greedy
3. Comprobamos si la solución obtenida por la búsqueda actual es la mejor solución.
4. Repetimos el proceso con una nueva solución.

PROBLEMA DE LA MOCHILA DE ELECCIÓN SIMPLE

- Disponemos de una mochila y un conjunto de objetos.
- Cada objeto tiene un peso y un valor.
- Deseamos llenar la mochila con aquellos objetos que produzcan un mayor beneficio sin sobrepasar el peso máximo (M).
- Solución al problema:
    - Debemos utilizar una función objetivo con las siguientes características:
        - Maximizar $\sum_{i=1}^{N} Pi * Xi$, donde Xi es cada posible solución del problema y Pi es el valor de cada objeto.
        - Restricción $\sum_{i=1}^{N} Wi * Xi, \leq M$, donde Wi es el peso de cada objeto.

SOLUCIONES AL PROBLEMA DE LA MOCHILA DE ELECCIÓN SIMPLE



Para resolver este problema, aplicaremos una codificación binaria y las siguientes funciones objetivo:

- Maximizar $\sum_{i=1}^{N} Wi * Xi$, siendo Wi el valor del objeto i y Xi la posible solución del problema.
- Restricción $\sum_{i=1}^{N} Pi * Xi, \leq 15$, siendo Pi es peso del objeto i.

Paso 0: Aplicamos una codificación binaria al problema. Por ejemplo, la posible solución [0 0 1 0 1] indica que se han utilizado los objetos 3 y 5.
Paso1: Aplicamos la función de restricción y eliminamos las soluciones con un peso superior a 15 Kg.
Paso2: Calculamos el valor de las soluciones, siendo la solución óptima aquella solución con el mayor valor.

- Algoritmo genético: Consiste en aplicar los siguientes pasos mientras existan generaciones por producir:

1. Población inicial: Generamos una población inicial formada por n individuos codificados mediante

codificación binaria (por ejemplo, el individuo1 sería [10010].

2. Selección de padres: Aplicamos una selección por torneo de k individuos, mediante la cual comparamos k individuos y consideramos como el individuo padre al mejor individuo.
3. Cruce de padres: En función de un operador de cruce, realizaremos un cruce estándar en n puntos mediante el cuál seleccionaremos n genes de cada padre y utilizaremos dichos genes seleccionados para generar un individuo hijo (el resto de los genes de los padres se utilizarán para generar otro individuo hijo).
4. Mutación de los hijos: En función de un operador de mutación, realizaremos una mutación estándar de n genes mediante la cual se intercambiarán los genes del cromosoma del hijo.
5. Elitismo: Una vez obtenida la nueva generación, aplicaremos elitismo para almacenar las mejores generaciones.
6. Finalmente, la nueva generación sustituirá a la generación actual.

PROBLEMA DE LA MOCHILA DE ELECCIÓN MÚLTIPLE

- Disponemos de una mochila y un conjunto de productos.
- Cada producto tiene un peso y un valor.
- Se desea llenar la mochila con 0 o N productos que produzcan un mayor beneficio sin sobrepasar el peso máximo de la mochila (M).
- Solución al problema:
    - Cada posible solución al problema (Xi) es un vector discreto.
    - Debemos utilizar la siguiente función objetivo:

- Maximizar $\sum_{i=1}^{N} Pi * Xi$, donde Pi es el peso del objeto i.
- Restricción $\sum_{i=1}^{N} Wi * Xi \leq M$, donde Wi es el valor del objeto i.

SOLUCIONES DEL PROBLEMA DE LA MOCHILA DE ELECCIÓN MÚLTIPLE

Para resolver este problema, debemos utilizar una codificación entera de las posibles soluciones al problema. Por ejemplo, la posible solución [1 4 3 2 0] indica que hemos utilizado el objeto1 una vez, el objeto2 4 veces,etc.
Una vez codificado el problema, aplicamos la función de restricción para obtener las soluciones válidas al problema. Finalmente, la solución óptima será aquella cuyo valor total sea el mayor.

- Algoritmo genético: Igual al algoritmo genético usado para solucionar el problema de la mochila de elección simple pero utilizando una codificación entera.

OTROS PROBLEMAS

Imagina que, tras la generación de la población inicial, la distribución de los individuos con respecto a la función de ajuste es la que muestra la siguiente tabla (el máximo de la función es 99.8).

Rango fitness	Número individuos
0 - 1	30
1 - 5	10
5-10	8
10-20	1
20-30	-
30-50	-
50-100	1

¿Qué método de selección de padres te parece más adecuado, la selección por torneo (t=20), la selección por rangos o la selección por ruleta?

El método de selección de padres más adecuado será aquel con una probabilidad de selección de padres mayor.
En la selección por torneo, la probabilidad de seleccionar el mejor padre será:

P (Mejor padre) = 1 - P (No mejor padre) = 1 - $(\frac{49}{50} * \frac{48}{49} * etc)$

, repitiendo la obtención de la probabilidad de no obtener el mejor padre durante 20 repeticiones.
En la selección por ruleta, se asigna a cada individuo una parte proporcional a su función de aptitud, de modo que los mejores individuos tendrán una proporción de ruleta mayor que los peores individuos. De este modo, la probabilidad de obtener el mejor padre será $\frac{f\ (Mejor\ padre)}{\sum_{i=1}^{N} fi}$, siendo esta probabilidad mayor que la probabilidad obtenida por la selección por torneo.
En la selección por rangos, la población se ordena de acuerdo a su función de aptitud y las probabilidades de selección se asignan en base a su ranking. De este modo, se obtiene una probabilidad de obtener al mejor padre superior a la probabilidad obtenida por la selección por ruleta.
Por tanto, el mejor método de selección de padres es la selección por rangos.

Imagina que quisiéramos encontrar los valores máximos de la función

$$f(x) = 3e^{-(x-5)^2} + 4e^{-(x-7)^2}$$

En el intervalo (0, 12). Considerando que esta función tiene dos máximos, propón la forma más apropiada de obtener todos los máximos (absoluto y relativo) en este intervalo.

Para obtener los máximos (absoluto y relativo) en dicho intervalo podemos utilizar los siguientes métodos:

1. Algoritmo genético de codificación real.
2. Algoritmo de nichos: Consiste en aplicar una función multimodal que produzca una convergencia hacia los puntos óptimos en el intervalo indicado.

Presenta un problema que sea fácil de resolver mediante un algoritmo de ACO, y justifica por qué el problema es apropiado para este tipo de algoritmos.

Un algoritmo ACO permite obtener el mejor camino de un conjunto de caminos. Por ello, este algoritmo es apropiado para problemas TSP, los cuáles consisten en obtener el camino con menor valor de fitness para recorrer un grafo.

Justifica las ventajas de utilizar un cruce BLX-α frente al cruce aritmético en términos del binomio explotación vs. exploración.

El binomio de explotación busca encontrar mejores soluciones en profundidad a una solución dada, mientras que el binomio de exploración busca soluciones más prometedoras alrededor de una solución.
El cruce aritmético sólo es explotador, mientras que el cruce BLX-α tiene parte de explotación como de exploración.

Imagina que queremos hallar el máximo la función $f(x) = x^2 - 2x + 1$ en el intervalo (0,4) con una precisión de centésimas mediante un algoritmo genético con codificación **binaria**. Explica cómo llevarías a cabo dicha codificación y cuáles serían los parámetros de diseño más importantes a definir para encontrar la solución a dicho problema.

Para llevar a cabo la decodificación binaria, debemos dividir el intervalo (0,4) en 2^n iteraciones, donde la anchura entre cada iteración debe ser menor o igual a 10^{-2}.
Los parámetros de diseño más importantes a definir para resolver el problema son:

1. Tamaño de la población: Representa el número de individuos de la población.
2. Número de generaciones: Representa el número total de poblaciones de nuevas generaciones que se van a generar.
3. Selección de padres: Se utilizará una selección por rangos de los individuos de la población.
4. Cruce y mutación.

5. Tipo de algoritmo genético: Se empleará un algoritmo genético generacional mediante el cuál la nueva generación sustituye a la generación anterior.

El problema de buscar una función que se ajuste a una nube de puntos, sin hacer hipótesis sobre la estructura de la función, se denomina regresión simbólica. Supongamos que tenemos la siguiente tabla y queremos buscar la función que se ajuste a estos datos

X	-2	-1	0	1	2
Y	-6	-1	0	3	14

¿Cómo resolverías dicho problema mediante programación genética **canónica**? Explica cuáles serían los nodos terminal y función que usarías, la profundidad máxima necesaria y otros parámetros relevantes.

Se utilizará un árbol jerárquico formado por individuos (una o más expresiones de la función de ajuste a los datos) y nodos terminales (variables, constantes, etc) y funcionales (operadores).
La función de ajuste a los datos es: $f = x^3 + x^2 + x = x * [(x+1)*x+1]$
Para calcular la profundidad máxima necesaria, debemos crear un individuo y calcular su profundidad máxima.
Una vez obtenida la profundidad máxima, creamos nuevos individuos de la función. Para ello, podemos utilizar el método full (las hojas del árbol tienen la misma profundidad), el método grow (se detiene la creación de cada rama del árbol al alcanzar la profundidad máxima) o ambos.
Una vez obtenidos, aplicamos un cruce de ramas entre 2 individuos al azar y realizamos una mutación sobre los nuevos individuos generados tras el cruce.

Codifica los atributos de la siguiente base de datos en su versión binarizada, para que podamos extraer reglas mediante un algoritmo genético. La variable de salida sería la denominada **Crash Severity**.

Weather Condition	Driver's Condition	Traffic Violation	Seat Belt	Crash Severity
Good	Alcohol-impaired	Exceed speed limit	No	Major
Bad	Sober	None	Yes	Minor
Good	Sober	Disobey stop sign	Yes	Minor
Good	Sober	Exceed speed limit	Yes	Major
Bad	Sober	Disobey traffic signal	No	Major
Good	Alcohol-impaired	Disobey stop sign	Yes	Minor
Bad	Alcohol-impaired	None	Yes	Major
Good	Sober	Disobey traffic signal	Yes	Major
Good	Alcohol-impaired	None	No	Major
Bad	Sober	Disobey traffic signal	No	Major
Good	Alcohol-impaired	Exceed speed limit	Yes	Major
Bad	Sober	Disobey stop sign	Yes	Minor

Para realizar la codificación de los atributos, debemos transformar los posibles valores de los atributos en series de valores lógicos (1 y 0).

- El atributo Weather Condition puede ser Good o Bad. Por ello, la codificación de dicho atributo sería de 1 bit, siendo 1 para Good y 0 para Bad.
- El atributo Driver's Condition puede ser Alcohol-impaired o Sober. Por ello, la codificación de dicho atributo sería de 1 bit, siendo 1 para Alcohol-impaired y 0 para Sober.
- El atributo Traffic Violation puede ser Exceed speed limit, None, Disobey stop sign o Disobey traffic signal. Por ello, la codificación de dicho atributo sería de 2 bits, siendo 00 para Exceed speed limit, 01 para None, 10 para Disobey stop sign y 11 para Disobey traffic signal.
- El atributo Seat Bealt puede ser NO o YES. Por ello, la codificación de dicho atributo sería de 1 bit, siendo 1 para NO y 0 para YES.
- El atributo Crash Severity puede ser Major o Minor. Como este atributo representa la variable de salida, su codificación sería de + para Major y - para Minor.

Imagina que quieres construir un clasificador mediante programación genética para los datos contenidos en la siguiente tabla. Indica cómo codificarías los atributos para construir reglas, considerando que la variable de salida es la última columna de la tabla.

RID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31 . . . 40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31 . . . 40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31 . . . 40	medium	no	excellent	yes
13	31 . . . 40	high	yes	fair	yes
14	>40	medium	no	excellent	no

- El atributo RID no se puede codificar pues sus posibles valores no son atributos.
- El atributo age puede ser <= 30, 31 … 40 o >40. Por ello, su codificación sería A para <=30, B para [31,40] y C para >40.
- El atributo income puede ser high, medium o low. Por ello, su codificación será 0 para high, 1 para medium y 2 para low.
- El atributo student puede ser no o yes. Por ello, su codificación sería 1 para no y 0 para yes.
- El atributo credit_rating puede ser fair o excellent. Por ello, su codificación sería 0 para fair y 1 para excellent.
- El atributo Class: buys_computer puede ser No o yes. Como representa la variable de salida, su codificación sería + para No y - para Yes.

Explica cómo construir un clasificador basado en reglas que permita determinar qué familia comprará un nuevo cortacésped a partir de sus ingresos y el tamaño de su parcela

Table 7.1: Lot Size, Income, and Ownership of a Riding Mower for 24 Households

Household number	Income ($ 000's)	Lot Size (000's ft^2)	Ownership of, riding mower
1	60	18.4	Owner
2	85.5	16.8	Owner
3	64.8	21.6	Owner
4	61.5	20.8	Owner
5	87	23.6	Owner
6	110.1	19.2	Owner
7	108	17.6	Owner
8	82.8	22.4	Owner
9	69	20	Owner
10	93	20.8	Owner
11	51	22	Owner
12	81	20	Owner
13	75	19.6	Non-Owner
14	52.8	20.8	Non-Owner
15	64.8	17.2	Non-Owner
16	43.2	20.4	Non-Owner
17	84	17.6	Non-Owner
18	49.2	17.6	Non-Owner
19	59.4	16	Non-Owner
20	66	18.4	Non-Owner
21	47.4	16.4	Non-Owner
22	33	18.8	Non-Owner
23	51	14	Non-Owner
24	63	14.8	Non-Owner

- El atributo Household number no se puede codificar pues no es un atributo.
- El atributo Income puede ser menor a 60, (60,80) o >80.
- El atributo Ownership of, riding mower representa el valor de salida.

Para construir el clasificador basado en regla, debemos construir un árbol sintáctico con los siguientes nodos:

- Nodos terminales: Atributos, operadores (<,<=,>,>=,=,!=).
- Nodos función: AND, NOT, OR.

CUESTIONES

- **Ventaja e inconveniente de utilizar agregación de objetivos frente a la búsqueda del frente de Pareto en los algoritmos de optimización multiobjetivo**

En los algoritmos de optimización multiobjetivo, la agregación de objetivos conduce a la obtención de un único punto en la frontera de Pareto por cada iteración, pero no permite la obtención de soluciones en regiones cóncavas.

- **Razona el motivo por el cuál los algoritmos que utilizan una población auxiliar obtienen mejores frente de Pareto que los demás**
  Porque al almacenar las soluciones no-denominadas encontradas a lo largo de la búsqueda en poblaciones externas, permiten al algoritmo cubrir de forma más adecuada el Frente de Pareto.
- **Enumera las diferencias más importantes entre la escalada, el enfriamiento simulado y la búsqueda tabú**
  La escalada puede quedar atrapada en un óptimo local al realizar la búsqueda, a diferencia del enfriamiento simulado y la búsqueda tabú.
  El enfriamiento simulado obtiene el óptimo global mediante una función, mientras que la búsqueda tabú utiliza la memoria para obtener el óptimo global.
- **Enumera las ventajas que tienen la escalda, el enfriamiento simulado y la búsqueda tabú con respecto a las demás.**
  La escalada se dirige siempre a un estado mejor que el actual.
  El enfriamiento simulado utiliza una función que disminuye la probabilidad de movimiento hacia soluciones peores al avanzar la búsqueda.
  La búsqueda tabú no repite las trayectorias de búsqueda.
- **Explica en qué consiste la presión selectiva, su utilidad en los algoritmos evolutivos, su mayor inconveniente y cómo se soluciona**
  La presión selectiva consiste en obtener el grado de reproducción que está dirigida por los mejores individuos de la población.

En los algoritmos evolutivos, la presión selectiva se utiliza para determinar el grado de diversidad de la población.
El mayor inconveniente de la presión selectiva es que puede provocar la convergencia prematura del proceso evolutivo.
Este problema puede resolverse aplicando técnicas para preservar la diversidad de la población (restricciones al cruzar los padres,etc).

- **Razona si crees o no que la inicialización de la matriz de feromonas es crítica en el buen funcionamiento de los algoritmos ACO.**
  Es crítica, porque permite consolidar la cantidad de feromonas que se almacena entre cada par de ciudades (i,j), de modo que el mejor camino entre 2 ciudades es aquel que tenga un mayor número de feromonas.