Ejercicio 3
El Problema de los Subuniversos

1. Ejerccio

Se tiene en un universo principal U con varias leyes.

$$U = \{l_1, l_2, \ldots, l_n\}$$

Sean $C_1, C_2, \ldots, C_k$ subuniversos donde cada uno presenta un subconjunto de leyes del universo principal. Se desea saber si existe un conjunto de estos subuniversos tal que su union formen el universo principal y que sean disjuntos entre si.

2. Solucion

2.1. Cobertura exacta de conjuntos / Exact Set Cover

Analizando el ejercicio podemos ver que es una instancia de un problema NP-Completo conocido, Exact Set Cover o Exact Cover.

Definicion (Exact Set Cover):

- Entrada:
  - Un conjunto finito X(universo)
  - Una coleccion S de subconjuntos de X
- Pregunta:
  - ¿Existe una subcoleccion $T \subseteq S$ tal que cada elemento de X pertence exactamente un subconjunto en T?

Mapeo de Instancias

- Universo:
  - Asignamos $U = X$
- Subconjuntos:
  - Asignamos $C = S$
- Objetivo:
  - Encontrar una subcoleccion de C que cubra cada elemento de U exactamente una vez.

Como podemos ver la estructura y pregunta de ambos problemas son identicas. Por lo que podemos afirmar que el problema de los subniversos es una instancia de Exact Set Cover.

2.2. Demostracion NP-Completo

Demostracion NP:
Para demostrar que el problema esta en NP, tenemos que verificar que en tiempo polinomial una solucion candidata sea valida.

Verificacion:
Cobertura Completa: Verificamos que la union de los subconjuntos seleccionados es igual a U. Podemos construir la union iterando sobre cada conjunto seleccionado y agregando sus elementos a un conjunto auxiliar.
Cobertura Exacta: Verificamos que no hay solapamientos entre los conjuntos seleccionados. Mantenemos un conjunto auxiliar y comprobamos que cada elemento se agrega exactamente una vez. Si encontramos un elemento que ya esta en el conjunto auxiliar al procesar un nuevo subconjunto, entonces hay solapamiento.
Tiempo Requerido: $O(\sum_{s \in S'} |s|)$ donde S' es la subcoleccion seleccionada.
La verificacion se pude realizar en tiempo polinomial respecto al tamaño del universo U y el total de elementos en los subconjuntos seleccionados.

Demostrar NP-Hard: Demostremos que el enigma de los subuniversos es NP-Hard mediante una reduccion desde el problema SAT que es conocido por ser NP-Completo.

Descripcion de SAT:

- Entrada:
  - Un conjunto de varaibles booleanas $B = \{b_1, b_2, \ldots, b_n\}$
  - Una formula f en forma normal conjuntiva (FNC), donde cada clausula C_i es la disyuncion literales(variables o su negacion).
- Pregunta:
  - ¿Existe una asignacion de valores de verdad a las variables en B que satisfaga todas las clausulas de f?

2.3. Reduccion

Direccion 1: Si existe una cobertura exacta entonces f es satisfacible
La idea principal es construir una fórmula booleana que codifique las restricciones del problema de Exact Set Cover. Para cada conjunto S_i en la familia S, introducimos una variable booleana x_i. La variable x_i será verdadera si y solo si el conjunto S_i es parte de la solución (es decir, si S_i está en S').

Cada elemento debe estar en exactamente un conjunto:
Para cada elemento $u \in U$, introducimos una cláusula que asegure que al menos un conjunto que contiene a u sea seleccionado:

$$x_{i1} \vee x_{i2} \vee \cdots \vee x_{ik}$$

donde $S_{i1}, S_{i2}, \ldots, S_{in}$ son todos los conjuntos que contienen a u. Esto asegura que cada elemento esté cubierto al menos una vez. Para cada par de conjuntos S_i y S_j que

intersectan, introducimos una cláusula que asegure que no ambos conjuntos puedan ser seleccionados simultáneamente

$$\neg x_i \wedge x_j$$

No se pueden seleccionar conjuntos vacíos:
Si algún conjunto S_i está vacío, simplemente no incluimos la variable x_i en la fórmula.

Ejemplo:
Consideremos el siguiente ejemplo de Exact Set Cover:
$U = \{a, b, c\}$
$S = \{\{a, b\}, \{b, c\}, \{c\}\}$
La fórmula booleana correspondiente sería:

$$(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge \neg(x_1 \wedge x_2) \wedge \neg(x_2 \wedge x_3)$$

donde x_1, x_2, y x_3 corresponden a los conjuntos $\{a, b\}, \{b, c\}, y\{c\}$ respectivamente.

Tomando como solucion del Exact Set Cover a x_1 y x_3. Obtenemos ver que satisface la formula, tomando como verdadero los literales de los conjuntos seleccionados de S y como falso los no seleccionados. x_1 verdadero, x_2 falso, x_3 verdadero.

La reducción de Exact Set Cover a SAT es polinomial. Esto significa que el tamaño de la fórmula booleana producida está acotado por un polinomio en el tamaño de la instancia de Exact Set Cover.

Direccion 2: Si f es satisfacible entonces existe una cobertura exacta.

Dada una instancia de SAT:

f = $C_1 \wedge C_2 \wedge \ldots \wedge C_n$ donde $C_i = \{\ x_{i1} \vee x_{i2} \vee \ldots \vee x_{ik}\}$ y $x_{i,j}$=\{ True or False, de la forma $\neg x_{i,j}$ o $x_{i,j}\}$

Crearemos una instancia de Exact Set Cover

Definamos un conjunto E y una familia de subconjuntos F, tal que existe una cobertura exacta de E en F si la formula es satifacible.

El conjunto universo E consistira de 3 tipos de elementos:

- x_i: por cada variable i
- C_j:: por cada clausula j
- $p_{i,j}$ por cada aparicion de la variable i en la clausula j

Para el conjunto F tendremos:

Por cada variable i creamos dos conjuntos, el conjunto verdadero x_i^T y el conjunto falso x_i^F. El conjunto verdadero contiene x_i en conjunto a las apariciones de i en cada clausula j donde aparezca negada $p_{i,j}$ y el conjunto falso contiene x_i en conjunto a las apraciciones de i en cada clausula j donde no aparezca negada p_{ij}.

Ejemplo: Si x_1 aparece en la clausula 2 y $\neg x_1$ aparece en las clausulas 3 y 4 creamos los conjuntos:
$x_1^T = \{x_1, p_{1,2}\}$ y $x_1^F = \{x_1, p_{1,3}, p_{1,4}\}$

Por cada elemento $p_{i,j}$ creamos un conjunto $\{C_j, p_{i,j}\}$ y otro solo con $\{p_{i,j}\}$.

Como los conjuntos verdaderos y falsos son los unicos que contienen el elemento x_i, por cada variable i, una solucion de Exact Set Cover debe contener exactamente uno

de los correspondientes conjuntos verdaderos o falsos. Esto da una relacion entre los conjuntos verdaderos/falsos seleccionandos en las soluciones del Exact Set Cover y las variables asignadas a las instancia de SAT.

Además, cualquier solución de Exact Set Cover debe, para cada cláusula j, contener para algunos i el conjunto $\{C_j, p_{i,j}\}$.

Si i es un literal positivo en la cláusula j, esto significa el conjunto falso para la variable i no puede estar en la solución, ya que también contiene $p_{i,j}$. De manera similar, si i es un literal negado en la cláusula j, no podemos haber seleccionado el conjunto verdadero para la variable i. Por tanto, la variable i satisface la cláusula j en la asignación de variables correspondientes a los conjuntos verdadero/falso de variables seleccionadas. Por el contrario, dada una asignación satisfactoria a la instancia SAT, podemos simplemente seleccionar los conjuntos de variables verdadero/falso correspondientes a la asignación.

Para cada cláusula podemos seleccionar cualquier conjunto $\{C_j, p_{i,j}\}$ por lo cual i es un literal satisfactorio de la cláusula j. Esto cubre los x_i y los C_j. El resto de los $p_{i,j}$ pueden estar cubiertos por los conjuntos independientes de estos.

Ejemplo:

$f = \{ x_1 \vee x_2 \vee \neg x_3\} \wedge \{ \neg x_1 \vee x_2 \vee x_3\}$

Creamos los conjuntos verdaderos y falsos

$x_1^T = \{x_1, p_{1,2}\}$ $x_1^F = \{x_1, p_{1,1}\}$ $x_2^T = \{x_2\}$ $x_2^F = \{x_2, p_{2,1}, p_{2,2}\}$ $x_3^T = \{x_3, p_{3,1}\}$ $x_3^F = \{x_3, p_{3,2}\}$

Creamos los conjuntos de clausulas con sus $p_{i,j}$

$\{C_1, p_{1,1}\}$, $\{C_1, p_{2,1}\}$, $\{C_1, p_{3,1}\}$ $\{C_2, p_{1,2}\}$, $\{C_2, p_{2,2}\}$, $\{C_2, p_{3,2}\}$ $\{C_3, p_{1,3}\}$, $\{C_3, p_{2,3}\}$, $\{C_3, p_{3,3}\}$

y los conjuntos independientes de $p_{i,j}$

$p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}$,$p_{2,3}, p_{3,1}, p_{3,2}$,$p_{3,3}$

Sea

$$F = \{\{p_{1,1}\}, \{p_{2,1}\}, \{p_{3,1}\}, \{p_{1,2}\}, \{p_{2,2}\}, \{p_{3,2}\}, \{p_{1,3}\}, \{p_{2,3}\}, \{p_{3,3}\}, x_1^T, x_1^F, x_2^T, x_2^F, x_3^T, x_3^F, \{C_1, p_{1,1}\}, \{C_1, p_{2,1}\}, \{C_1, p_{3,1}\}\{C_2, p_{1,2}\}, \{C_2, p_{2,2}\}, \{C_2, p_{3,2}\}\{C_3, p_{1,3}\}, \{C_3, p_{2,3}\}, \{C_3, p_{3,3}\}\} \tag{1}$$

Sea el universo

$$A = \{x_1, x_2, x_3, C_1, C_2, p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}, p_{3,1}, p_{3,2}, p_{3,3}\}$$

Dada una solucion cualquiera de SAT sea esta

$$s = \neg x_1 \wedge x_2 \wedge x_3$$

cubriremos el universo, dando como resultado el subconjunto

$$S = \{x_2^T, x_1^F, x_3^T, \{C_1, p_{2,1}\}, \{C_2, p_{1,2}\}, \{p_{2,2}\}, \{p_{3,2}\}\}$$

Podemos observar que son disjuntos y hacen un cubrimiento completo del universo. Con estas dos reducciones podemos demostrar que estos problemas son equivalentes. Hemos demostrados que el enigma de los subuniversos es NP-Completo.

2.4. Algoritmo

Backtrack

Para la solucion de este problema, planteamos un algoritmo de backtrack que iteramos por cada conjunto y vamos agregando si no existe solapamiento. Termina si encuentra una solucion y la devuelve, si no devuelve vacio.

```
def exact_set_cover(universe, subsets):
    used_subsets = []

    def backtrack(covered, index):
        if covered == universe:
            return True
        if index >= len(subsets):
            return False

        for i in range(index, len(subsets)):
            subset = subsets[i]

            if not covered & subset:
                used_subsets.append(subset)
                if backtrack(covered | subset, i + 1):
                    return True
                used_subsets.pop()
        return False

    if backtrack(set(), 0):
        return used_subsets
    else:
        return None
```

Podemos ver que es un algoritmo bastante sencillo pero ineficiente ya que recorre todos los posibles subconjuntos.

Greedy Conjunto de Mayor Tamaño

Como el anterior es muy ineficiente para mejorar la eficiciencia del algoritmo usamos un enfoque greedy donde siempre a la hora de agregar tomamos el conjunto con mayor cantidad de elementos. Para lograr es ordenamos el conjunto S de mayor a menor en cuanto a longitud y realizamos backtrack.

Resultados:

Para un universo U de 30 elementos y un conjunto S de 75 subuniversos.

Backtrack: 0.03481864929199219 ms.
Greedy Conjunto Mayor Tamaño: 0.006000041961669922ms.
Podemos observar que el algoritmo greedy es mas eficiente al de backtrack.