




## Teoría de Algoritmos I 75.29

### Trabajo Práctico N° 3

Grupo Rosita Forever

Apellido y Nombre	Padrón	Correo electrónico
Jamilis, Netanel David	99093	njamilis@fi.uba.ar
Del Torto, Agustín	98867	adeltorto@fi.uba.ar
Daverede, Agustín	98540	agusdaverede@yahoo.com.ar
Betz, Joaquín	104348	

Github: 

<https://github.com/netaneldj/7529-TDA/TP3>

# Indice

<b>1</b>	<b>Introducción</b>	<b>3</b>
1.1	Objetivos . . . . .	3
1.2	Resumen . . . . .	3
<b>2</b>	<b>Manifestaciones seguras</b>	<b>4</b>
2.1	Solución . . . . .	4
2.2	Demostración NP-Completo . . . . .	4
2.2.1	Demostración NP . . . . .	4
2.2.2	Demostración NP-Hard . . . . .	5
<b>3</b>	<b>División de Bienes</b>	<b>9</b>
3.1	Solución . . . . .	9
3.2	Demostración NP-C . . . . .	9
<b>4</b>	<b>Un poco de teoría</b>	<b>12</b>
4.1	Definiciones . . . . .	12
4.1.1	Problemas P y NP . . . . .	12
4.2	NP-completo y NP-hard . . . . .	12
4.3	Complejidad . . . . .	12
4.4	Ejemplos . . . . .	12
4.5	Problema . . . . .	12
4.5.1	I . . . . .	13
4.5.2	II . . . . .	13
4.5.3	III . . . . .	13
<b>5</b>	<b>Conclusión</b>	<b>14</b>
5.1	Manifestaciones seguras . . . . .	14
5.2	División de Bienes . . . . .	14
5.3	Un poco de teoría . . . . .	14
<b>6</b>	<b>Bibliografía y Referencias</b>	<b>15</b>
<b>7</b>	<b>Anexo Manifestaciones seguras</b>	<b>16</b>
7.1	Solución . . . . .	16
7.2	Demostración NP-Completo . . . . .	16
7.2.1	Demostración NP . . . . .	16
7.2.2	Demostración NP-Hard . . . . .	17

<b>8</b>	<b>Anexo División de Bienes</b>	<b>19</b>
8.1	Demostración NP . . . . .	19
8.2	Reducción del problema SUBSET-SUM . . . . .	19
8.2.1	$\{A, k\} \in SUBSET-SUM \Rightarrow \{A'\} \in NUMBER-PARTITION$ . . . .	20
8.2.2	$\{A, k\} \in SUBSET-SUM \Leftarrow \{A'\} \in NUMBER-PARTITION$ . . . .	20
8.2.3	Corolario . . . . .	21
<b>9</b>	<b>Anexo Un poco de teoría</b>	<b>22</b>
9.1	Definiciones . . . . .	22
9.1.1	Problemas P y NP . . . . .	22
9.2	NP-completo y NP-hard . . . . .	22
9.3	Complejidad . . . . .	22
9.4	Ejemplos . . . . .	22
9.5	Problema . . . . .	22
9.5.1	I . . . . .	23
9.5.2	II . . . . .	23
9.5.3	III . . . . .	23

# 1 Introducción

## 1.1 Objetivos

Los objetivos del presente trabajo son los siguientes:

- Aplicar los conceptos aprendidos en clase en problemas NP
- Preparar un informe técnico

En las siguientes secciones se vera plasmado el cumplimiento de los objetivos.

## 1.2 Resumen

El presente trabajo se centrará en la presentación y consecuente resolución de los problemas planteados.

En primer lugar, se tratará el problema de manifestaciones seguras, perteneciente a la familia de problemas *Node-disjoint directed path problem*. Se demostrará que la solución pedida es NP-completa. Luego, se repetirá el mismo análisis para el problema de división de bienes, perteneciente a la familia de problemas *Number Partitioning*. Se demostrará que la solución pedida es NP-completa. Por último se realizarán una serie de demostraciones teoricas de forma generica.

Para concluir, se analizaran los resultados obtenidos, donde se tendrán en cuenta los conocimientos adquiridos en el desarrollo del trabajo y la bibliografía consultada para la realización del presente informe.

## 2 Manifestaciones seguras

En una decisión temeraria una ciudad decidió autorizar un conjunto de  $n$  manifestaciones el mismo día y horas. Cada manifestación comienza en un punto de reunión y tiene un destino final. Para evitar enfrentamientos y confusiones desean que cada ruta sea aislada de las otras. Contamos con el mapa de la ciudad que incluye todos los caminos e intersecciones por los que pueden ir las marchas. Nos piden que elaboremos un algoritmo que retorne los caminos a seguir para cada manifestación de modo que no haya riesgo de un cruce (si es posible).

Debemos demostrar que es un problema NP-Completo.

### 2.1 Solución

El problema que precede se puede modelar con un grafo en donde los nodos son las intersecciones de las calles y las aristas son las calles de la ciudad. En un principio todas las calles están disponibles para efectuar las manifestaciones siempre y cuando no se superpongan las manifestaciones.

### 2.2 Demostración NP-Completo

Para demostrar que es NP – Completo debemos demostrar que nuestro problema pertenece a NP y a NP – Hard simultáneamente.

#### 2.2.1 Demostración NP

Para demostrar que el problema es NP debemos poder corroborar la solución propuesta en tiempo polinomial. Usando la analogía de la cerradura y la llave, esto sería equivalente a dada la llave probar que esta abre la cerradura en "tiempo polinomial".

A continuación se propone un algoritmo que verifica la solución propuesta en tiempo polinomial:

Llamar  $L$  a la lista de manifestaciones(representadas por tuplas de nodos)

Funcion esManifestacionValida( $L$ ):

    Llamar visitados a la lista que contendrá a los nodos visitados

    visitados es lista vacia

    Para cada manifestacion en  $L$ :

        Para cada esquina en manifestacion:

            Si esquina en visitados:

                Devolver falso

        Sino:

visitados agregar esquina  
Devolver verdadero

Se puede apreciar que la complejidad es de  $\mathcal{O}(km)$  siendo  $k$  la cantidad de manifestaciones y  $m$  la cantidad máxima de esquinas atravesadas por una manifestación.

### 2.2.2 Demostración NP-Hard

Para demostrar que el problema es **NP – Hard**, consideremos una instancia del problema conocido como conjunto independiente que consiste en un grafo  $G = (V, E)$  con  $|V| = n$  y un valor entero  $k$ .

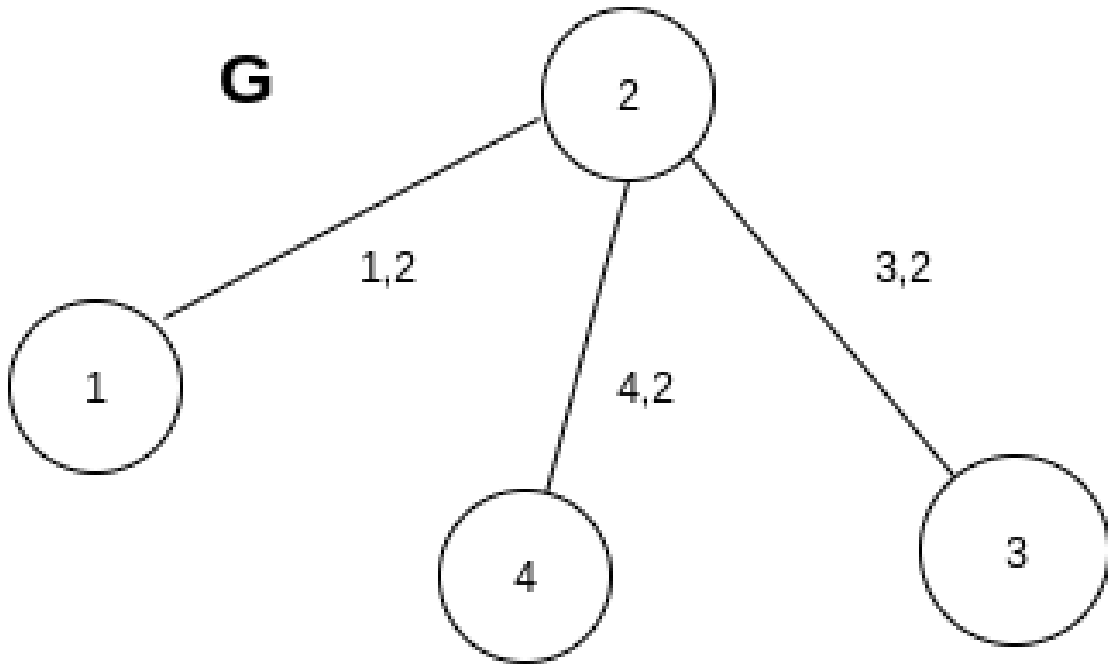


Figure 1: Instancia conjunto independiente

Sea el grafo  $M = (V', E')$  donde  $V' = V \cup \{x_{u,v} : (u, v) \in E\}$  y  $E' = V' \times (V' - 1)$ .

**M**

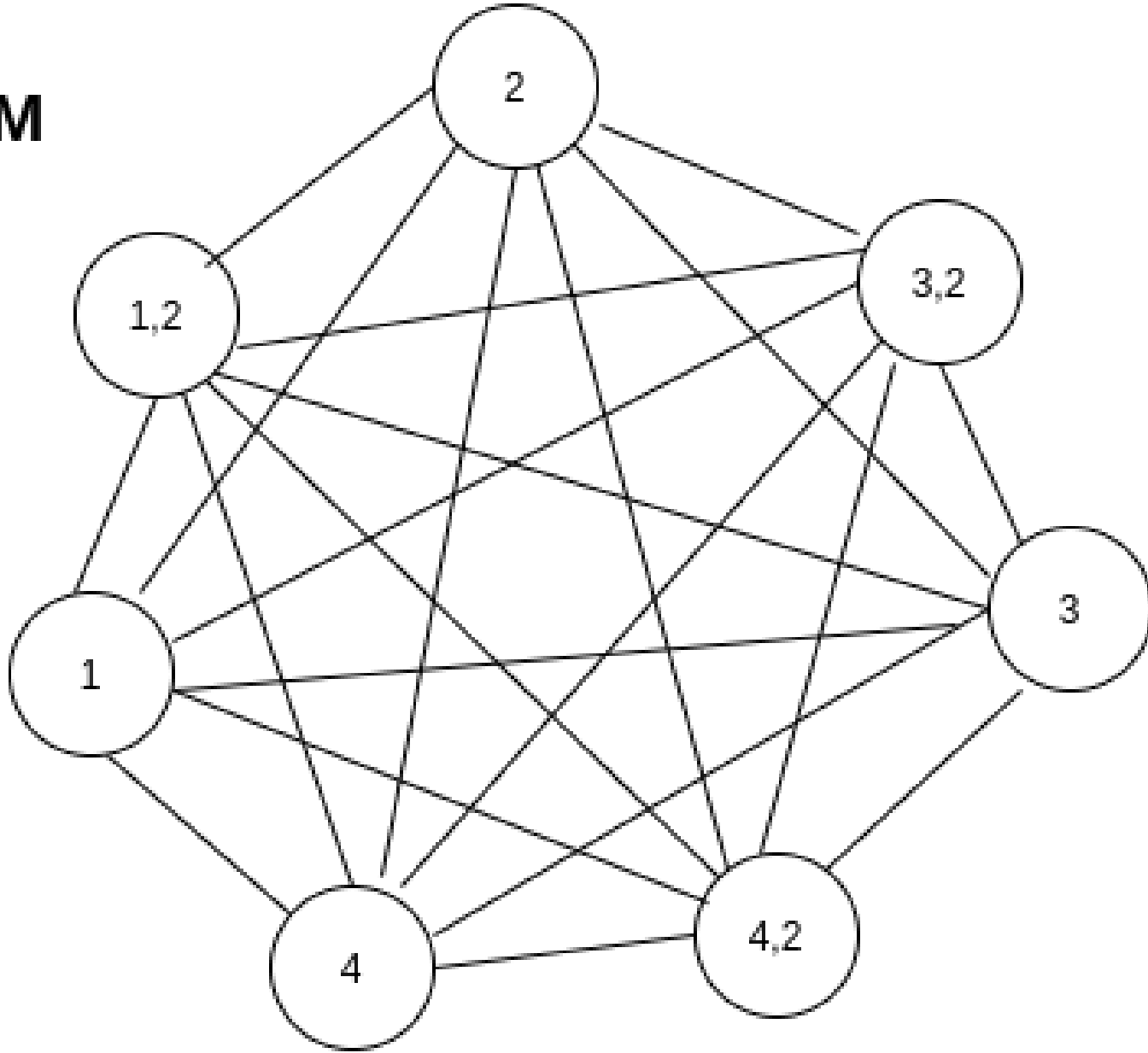


Figure 2: Grafo manifestaciones

Para cada  $u \in V$  siendo  $v_1, v_2, \dots, v_m$  los vecinos de  $u$  en  $G$  y definamos la ruta  $P_u = \langle u, x_{u,v_1}, x_{u,v_2}, \dots, x_{u,v_m} \rangle$  en  $M$ . Siendo  $\mathcal{P} = \{P_u : u \in V\}$ .

Hay un conjunto independiente de tamaño como máximo  $k$  en  $G$ , si y solo si hay un subconjunto  $\mathcal{P}'$  con  $|\mathcal{P}'| \leq k$  tal que los caminos en  $\mathcal{P}'$  son vértices disjuntos.

Para visualizarlo, construimos los siguientes Paths en  $M$ :

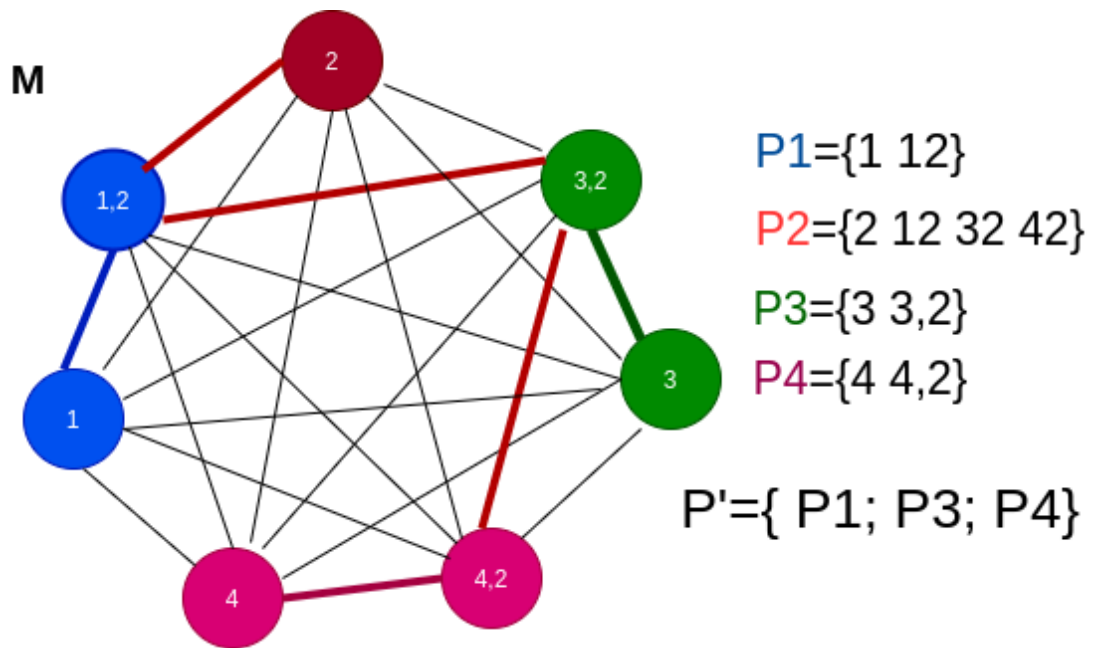


Figure 3: Rutas

Ahora queda formado un  $\mathcal{P}' = \{ P1; P3; P4 \}$  con  $|\mathcal{P}'| = 3$  que representa en nuestro grafo  $G$ :

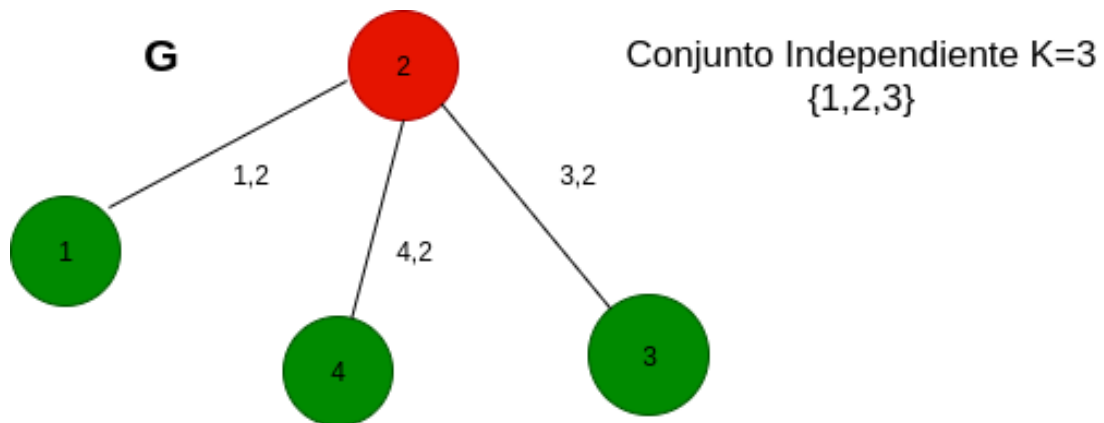


Figure 4: Nodos Independientes

Más precisamente, si  $S$  es un conjunto independiente de  $G$ , entonces  $\{P_u : u \in S\}$  es una colección de caminos separados de vértices  $M$  y, si  $\mathcal{P}'$  es una colección de caminos separados de vértices en  $M$ , entonces  $\{u : P_u \in \mathcal{P}'\}$  es un conjunto independiente de  $G$ .





### 3 División de Bienes

Una de las parejas más ricas del mundo está pasando por un proceso de divorcio. Entre sus bienes cuentan con propiedades, autos, motos, estampillas raras y otros coleccionables. Como no se ponen de acuerdo en la manera de dividirlos, el juez ha dictaminado que un tasador ponga valor a cada bien y luego se haga una partición por valores iguales. El juez nos pide que elaboremos un algoritmo que en forma eficiente haga este trabajo.

#### 3.1 Solución

El problema que precede se puede pensar como una representación de un conjunto  $C = \{w_1, w_2, \dots, w_n\}$  donde cada  $w_i$  está asociado al precio de cada bien. Luego, se desea determinar si existe un subconjunto de  $C$  tal que:

$$W = \sum w_i \quad (1)$$

Donde  $w_i$  representa el precio de cada bien.

Utilizando conceptos de programación dinámica es posible llegar a una solución. No obstante, dicha solución estaría caracterizada por ser  $O(W \cdot n)$ , donde  $n$  es la cantidad de bienes y  $W$  es el valor de la ecuación (1) al que se desea llegar. Esto puede llegar a ser de orden casi exponencial si considera un  $W$  muy grande, y se tiene en cuenta la cantidad de bits empleados para representarlo y su crecimiento.

Sin embargo, como se describirá en el siguiente apartado, el problema es NP-Completo, de manera que no es posible hallar una solución que no sea exponencial. Al menos así ha sido descripto hasta ahora.

#### 3.2 Demostración NP-C

Para poder demostrar que el problema es NP-Completo, se decidió proceder realizando la demostración a partir de dos hipótesis:

- El problema es NP
- El problema es NP-Hard

Si ambas se cumplen, podemos afirmar que el problema es NP-Completo.

En primer lugar, se debe demostrar que es NP. Utilizando la notación previa, se puede decir que el subconjunto  $C$  puede tener tantos elementos como el conjunto original. Para poder hallar a  $W$  es preciso realizar las sumas asociadas a las iteraciones de los diferentes  $w_i$ . Esto se puede resolver en tiempo polinomial, luego el problema es NP.

Ahora se debe probar que el problema es NP-Hard.

Para eso se va a demostrar:

$$3DM \leq_p SUBSET - SUM \quad (2)$$

La demostración se sustenta en que es conocido que el problema 3 dimensional matching es NP-C.

Para proceder a demostrar que el problema SUBSET SUM se puede reducir a 3DM se planteará que el conjunto  $C \subseteq X, Y, Z$ , siendo estos conjuntos ordenados disjuntos de tamaño  $n$  cada uno. Se desea determinar si existe el conjunto  $C$ .

Luego, se procede a representar a una tripla  $t = x_i, y_i, z_i$  como una serie de bits de tamaño  $3n$ . A los efectos de resolver el problema se plantea que el conjunto  $C$  se puede representar como se demuestra a continuación:

$$\begin{array}{r}
 \begin{array}{c}
 \boxed{0} \boxed{1} \boxed{0} \dots \boxed{1} \boxed{0} \boxed{0} \dots \boxed{0} \boxed{1} \boxed{0} \\
 \boxed{0} \boxed{1} \boxed{0} \dots \boxed{0} \boxed{1} \boxed{0} \dots \boxed{1} \boxed{0} \boxed{0} \\
 + \quad \vdots \\
 \boxed{1} \boxed{0} \boxed{0} \dots \boxed{0} \boxed{1} \boxed{0} \dots \boxed{0} \boxed{0} \boxed{1} \\
 \vdots \\
 \boxed{0} \boxed{0} \boxed{1} \dots \boxed{0} \boxed{0} \boxed{1} \dots \boxed{1} \boxed{0} \boxed{0}
 \end{array} \\
 \hline
 W = \boxed{1} \boxed{1} \boxed{1} \dots \boxed{1} \boxed{1} \boxed{1} \dots \boxed{1} \boxed{1} \boxed{1}
 \end{array}$$

Figure 6: Representación de tripla

No obstante, esta representación trae aparejado un posible problema de overflow. Esto se puede observar en la siguiente representación:

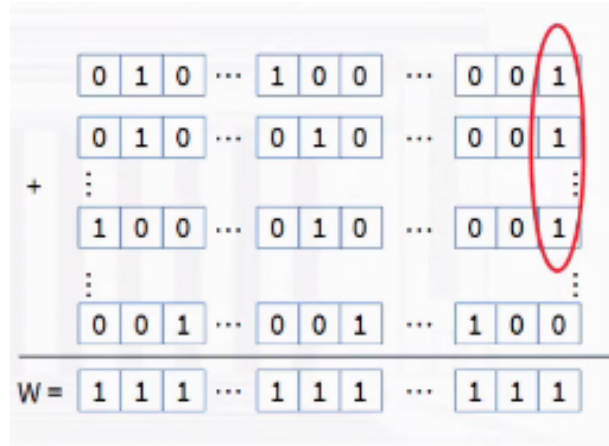


Figure 7: Problema de overflow

Para evitar esta situación se debe elegir una representación en una base al menos un valor mayor de la cantidad de triplas, ya que esto imposibilita que se presente el overflow.

Al llegar a una representación del problema que se puede caracterizar por 3DM, podemos afirmar que el mismo representa a un problema similar a 3DM.

Por lo tanto, como 3DM es NP-C, se probó la ecuación (2) y también que el problema es NP, se puede afirmar que se trata de un problema NP-C.

## 4 Un poco de teoría

### 4.1 Definiciones

#### 4.1.1 Problemas P y NP

Un problema P es todo aquel problema de decisión que se puede resolver eficientemente (es decir, en tiempo polinomial), mientras que un problema NP es el que, contando con una posible solución del mismo, ésta se puede verificar eficientemente.

### 4.2 NP-completo y NP-hard

Si todo problema NP se puede reducir al problema A, A es un problema NP-hard. Y los NP-complete son aquellos problemas NP-hard que además son NP.

### 4.3 Complejidad

Si quisiéramos ordenar estos problemas de menor a mayor complejidad, en términos generales:

- P
- NP
- NP-Completo
- NP-Hard

### 4.4 Ejemplos

- Problema P, NP: Camino más corto.
- Problema NP-completo: Problema de la mochila.
- Problema NP-hard: Problema de la parada.

### 4.5 Problema

Tenemos un problema A, un problema B y una caja negra NA y NB que resuelven el problema A y B respectivamente. Sabiendo que B es NP

#### **4.5.1 I**

Qué podemos decir de A si utilizamos NA para resolver el problema B (asumimos que la reducción realizada para adaptar el problema B al problema A es polinomial)

Podemos decir que A es igual o menos complejo que B.

#### **4.5.2 II**

Qué podemos decir de A si utilizamos NB para resolver el problema A (asumimos que la reducción realizada para adaptar el problema A al problema B es polinomial)

No podemos decir nada de A porque como NB resuelve NP, le podemos pasar tanto problemas P como NP

#### **4.5.3 III**

Qué pasa con los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es P?

1. Decimos que A es igual o más complejo que B y ambos se pueden resolver de forma polinómica.
2. Decimos que B es igual o más complejo que A y no necesariamente B se resuelve de forma polinómica.

## 5 Conclusión

Habiendo culminado el trabajo podemos efectuar las siguientes conclusiones:

### 5.1 Manifestaciones seguras

- El problema se simplifica al modelar la ciudad como un grafo.
- Cada vértice independiente en *Independent set* equivale a un camino independiente en nuestro problema

### 5.2 División de Bienes

- El problema puede ser resuelto empleando una simplificación que puede ser útil desde el punto de vista computacional, ya que la resolución empleando bits dá la posibilidad de realizar operaciones de modo más eficiente.
- Se logró demostrar que el problema era NP-C. Luego, se llegó a la certeza, al menos por ahora, de que no es posible plantear una solución que no sea de orden superior al polinomial.

### 5.3 Un poco de teoría

- Un problema solo puede ser resuelto por otro igual o más complejo que el primero

## 6 Bibliografía y Referencias

- J. Kleinberg, E. Tardos, Algorithm Design
- T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms



## 7 Anexo Manifestaciones seguras

En una decisión temeraria una ciudad decidió autorizar un conjunto de  $n$  manifestaciones el mismo día y horas. Cada manifestación comienza en un punto de reunión y tiene un destino final. Para evitar enfrentamientos y confusiones desean que cada ruta sea aislada de las otras. Contamos con el mapa de la ciudad que incluye todos los caminos e intersecciones por los que pueden ir las marchas. Nos piden que elaboremos un algoritmo que retorne los caminos a seguir para cada manifestación de modo que no haya riesgo de un cruce (si es posible).

Debemos demostrar que es un problema NP-Completo.

### 7.1 Solución

El problema que precede se puede modelar con un grafo en donde los nodos son las intersecciones de las calles y las aristas son las calles de la ciudad. En un principio todas las calles están disponibles para efectuar las manifestaciones siempre y cuando no se superpongan las manifestaciones.

### 7.2 Demostración NP-Completo

Para demostrar que es NP – Completo debemos demostrar que nuestro problema pertenece a NP y a NP – Hard simultáneamente.

#### 7.2.1 Demostración NP

Para demostrar que el problema es NP debemos poder corroborar la solución propuesta en tiempo polinomial. Usando la analogía de la cerradura y la llave, esto sería equivalente a dada la llave probar que esta abre la cerradura en "tiempo polinomial".

A continuación se propone un algoritmo que verifica la solución propuesta en tiempo polinomial:

Llamar  $L$  a la lista de manifestaciones(representadas por tuplas ordenadas de nodos)

Funcion esManifestacionValida( $L$ ):

    Llamo  $C$  al grafo que representa a la ciudad

    Llamar visitados a la lista que contendrá a los nodos visitados

    visitados es lista vacia

    Para cada manifestacion en  $L$ :

        Para cada esquina en manifestacion:

            Llamo  $S$  a siguiente esquina

            Si esquina en visitados:

```

        Devolver falso
    Si existe S y no existe camino(esquina,S):
        Devolver falso
    Sino:
        visitados agregar esquina
Devolver verdadero

```

Se puede apreciar que la complejidad es de  $\mathcal{O}(km)$  siendo  $k$  la cantidad de manifestaciones y  $m$  la cantidad máxima de esquinas atravesadas por una manifestación.

### 7.2.2 Demostración NP-Hard

Para demostrar que el problema es NP – Hard, consideremos una instancia del problema 3-SAT compuesto por  $m$  clausuras y  $n$  variables:

$$E = (X_1 + X_2 + X_3) * (\overline{X_1} + \overline{X_2} + \overline{X_4}) * (\overline{X_2} + \overline{X_3} + X_4)$$

Para crear el grafo en el que se resolverá el problema de las manifestaciones definimos para cada variable ( $i$  de 1 a  $n$ ) y para cada clausura ( $j$  de 1 a  $m$ ) dos vértices:  $X_{ij}$  (*true*) y  $\overline{X_{ij}}$  (*false*); creando así 6 vértices por clausura. A su vez definimos un par de vértices  $I$  (inicio) y  $F$  (fin) por cada clausura y por cada variable quedándonos así formados los pares  $I_{jc}, F_{jc}$  (para los pares de las clausuras) y  $I_{iv}, F_{iv}$  (para los pares de las variables).

Por cada clausura conectaremos  $I_{jc}$  a  $X_{ij}$  si la variable  $X_{ij}$  esta sin negar y por consiguiente a  $\overline{X_{ij}}$  si la variable esta negada y luego conectaremos la variable  $X_{ij}$  usada al  $F_{jc}$  correspondiente.

Por cada variable conectaremos  $I_{iv}$  al vértice  $X_{ij}$  cuya  $j$  sea la clausura de menor índice en la que aparezca la variable  $i$ , después conectaremos dicho  $X_{ij}$  al  $X_{ij}$  cuya  $j$  sea la segunda clausura de menor índice en el que aparezca la variable  $i$ , crearemos aristas hasta que ya no queden clausuras que utilicen la variable  $i$  y procederemos a unir el ultimo  $X_{ij}$  al  $F_{iv}$ . Realizaremos exactamente el mismo procedimiento con las variables  $\overline{X_{ij}}$ .

De esta forma nos queda el siguiente grafo:

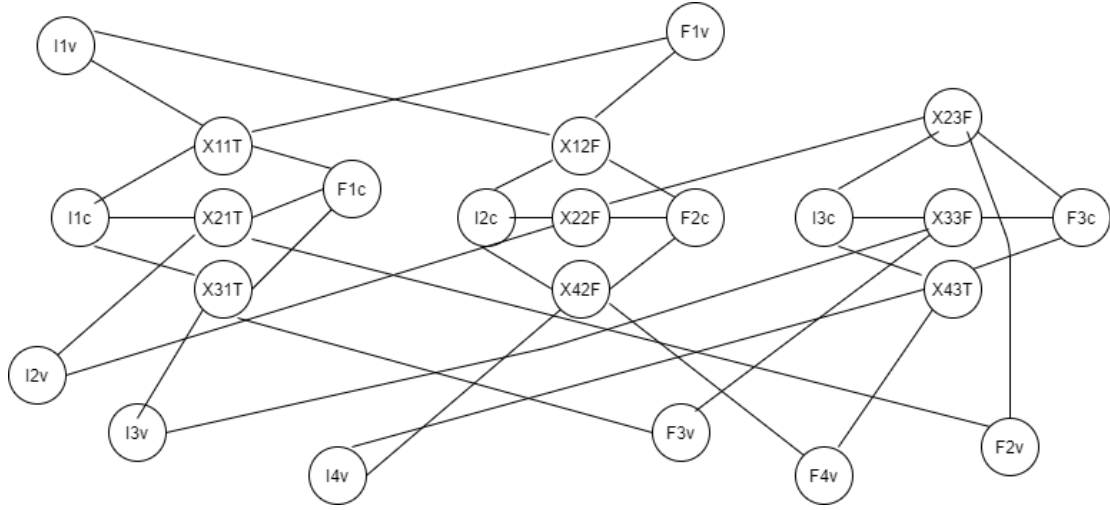


Figure 8: Grafo manifestaciones

Nota: No se muestran en el grafo los  $X_{ij}$  que se encuentran aislados del resto para no sobrecargar al mismo y facilitar su comprensión.

Por ultimo definiremos las manifestaciones. Existirá una manifestación por clausura, esta iniciara en el nodo  $I_{jc}$  y finalizara en el  $F_{jc}$  y otra manifestación por cada variable la cual iniciara en el nodo  $I_{iv}$  y finalizara en el  $F_{iv}$ .

Entonces en relación al problema de 3-SAT el camino que tomen las manifestaciones de las clausuras será la evaluación literal de *true* de la variable y el camino que no usemos de las manifestaciones de las variables corresponderá a la asignación de *true* a la variable.

## 8 Anexo División de Bienes

Una de las parejas más ricas del mundo está pasando por un proceso de divorcio. Entre sus bienes cuentan con propiedades, autos, motos, estampillas raras y otros coleccionables. Como no se ponen de acuerdo en la manera de dividirlos, el juez ha dictaminado que un tasador ponga valor a cada bien y luego se haga una partición por valores iguales. El juez nos pide que elaboremos un algoritmo que en forma eficiente haga este trabajo.

El pedido se puede plantear como un problema *NUMBER-PARTITION*; dado un conjunto  $C = \{w_1, w_2, \dots, w_n\}$ , donde cada  $w_i$  está asociado al precio de cada bien, se desea determinar si existe un subconjunto  $W$  de  $C$  tal que la suma de sus elementos equivalga a la suma de los elementos restantes en  $C$ .

$$\sum_{w \in W} w = \sum_{w \in \overline{W}} w \quad (3)$$

Para que un problema  $X$  sea NP-Completo, debe cumplirse que:

- (1)  $X$  sea NP
- (2) cualquier problema NP-Completo  $Y$  puede reducirse a  $X$ , en tiempo polinomial
- (3)  $X$  tenga una solución si y sólo si  $Y$  tiene solución

### 8.1 Demostración NP

Corroborar que dos particiones de un conjunto sean solución de este problema es muy simple. Sólo hay que verificar que la suma total de sus elementos sea equivalente. Por lo tanto, verificarlo es  $\mathcal{O}(n)$ , siendo  $n$  la cantidad de elementos del subconjunto más poblado. Entonces, al poder verificar la solución en tiempo polinomial, se puede asegurar que este problema es NP.

### 8.2 Reducción del problema SUBSET-SUM

Un conocido problema NP-Completo es el problema de la suma de los subconjuntos. Éste se puede describir como: Dado un conjunto de enteros  $A$ , ¿existe un subconjunto  $S \subset A$ , tal que la suma de sus elementos sea exactamente  $k$ ?

En términos matemáticos:

$$A = \{w_1, w_2, \dots, w_n\}, \sum_{w \in A} w_i = s$$
$$\exists S \subset A / \sum_{w \in S} w_i = k?$$

Lo que se desea corroborar ahora es si  $SUBSET-SUM \leq_p NUMBER-PARTITION$ . A tales efectos, se procederá a adaptar el problema *SUBSET-SUM* para que sea resoluble a través de *NUMBER-PARTITION*.

Sea  $s$  la suma de los miembros de  $A$  y  $A' = A \cup \{s - 2k\}$  un conjunto definido, donde  $k$  es el valor objetivo previamente mencionado. Se aceptará que  $A$  tiene solución si y sólo si *NUMBER-PARTITION* acepta  $A'$ .

$$A' = \{w_1, w_2, \dots, w_n, s - 2k\}, \sum_{w \in A'} w = s + (s - 2k) = 2s - 2k \quad (4)$$

Se desea probar que  $\{A, k\} \in \text{SUBSET-SUM} \Leftrightarrow \{A'\} \in \text{NUMBER-PARTITION}$ . Se hará esto en ambos sentidos.

### 8.2.1 $\{A, k\} \in \text{SUBSET-SUM} \Rightarrow \{A'\} \in \text{NUMBER-PARTITION}$

Si existe un conjunto de números en  $A$  que sumen  $k$ , entonces el resto de los números sumarán  $s - k$ .

$$\begin{aligned} S \subset A / \sum_{w \in S} w &= k \\ \Rightarrow \sum_{w \in \bar{S}_A} w &= \sum_{w \in A} w - \sum_{w \in S} w = s - k \end{aligned}$$

Se llamará  $\bar{S}$  al complemento de  $S$  en  $A$ . Entonces, es posible descomponer  $A'$  de la siguiente forma (haciendo abuso de notación, pues  $A$  y  $A'$  son conjuntos casi idénticos).

$$\begin{aligned} A' &= A \cup \{s - 2k\} \\ &= \{\bar{S} \cup S\} \cup \{s - 2k\} \\ &= \bar{S} \cup \{S \cup \{s - 2k\}\} \\ &= B \cup \bar{B}_{A'} \end{aligned}$$

Como  $B \equiv \bar{S}$ , la suma de sus elementos es " $s - k$ ". Luego, es posible calcular cuánto suma el conjunto de números restantes en  $A'$  (complemento de  $B$  en  $A'$ ).

$$\begin{aligned} \sum_{w \in \bar{B}_{A'}} w &= \sum_{w \in A'} w - \sum_{w \in B} w \\ &= 2s - 2k - (s - k) = s - k \end{aligned}$$

Por lo tanto, existe una partición en dos de  $A'$  ( $B$  y  $\bar{B}_{A'}$ ) tal que cada conjunto sume " $s - k$ " (exactamente la mitad de la suma total de  $A'$ ).

### 8.2.2 $\{A, k\} \in \text{SUBSET-SUM} \Leftarrow \{A'\} \in \text{NUMBER-PARTITION}$

Si existe una partición en dos de  $A'$  tal que cada una sume  $s - k$  (la mitad de su suma), entonces una de ellas necesariamente contiene el número  $s - 2k$  que se agregó previamente. Al descartar del conjunto este valor, se obtiene un subconjunto de  $A'$  cuya suma sea  $k$ .

### 8.2.3 Corolario

Un problema NP-Completo, como lo es el SUBSET-SUM, es reducible al problema del enunciado, NUMBER-PARTITION. Esta reducción es claramente en tiempo polinomial; para adaptar el conjunto "de entrada" sólo hay que incluir un valor  $(s-2k)$  para que pueda ser resuelto como una instancia de NUMBER-PARTITION. Además, para adaptar su salida sólo habría que identificar la partición con el valor que se agregó y descartarlo.

Teniendo todo esto en cuenta, se cumplen las 3 condiciones enunciadas y queda demostrado que NUMBER-PARTITION es NP-Completo porque, para poder resolverlo, se debe resolver el problema SUBSET-SUM, que es NP-completo.

## 9 Anexo Un poco de teoría

### 9.1 Definiciones

#### 9.1.1 Problemas P y NP

Un problema P es todo aquel problema de decisión que se puede resolver eficientemente (es decir, en tiempo polinomial), mientras que un problema NP es el que, contando con una posible solución del mismo, ésta se puede verificar eficientemente.

### 9.2 NP-completo y NP-hard

Si todo problema NP se puede reducir al problema A, A es un problema NP-hard. Y los NP-complete son aquellos problemas NP-hard que además son NP.

### 9.3 Complejidad

Si quisiéramos ordenar estos problemas de menor a mayor complejidad, en términos generales:

- P
- NP
- NP-Completo
- NP-Hard

### 9.4 Ejemplos

- Problema P, NP: Camino más corto.
- Problema NP-completo: Problema de la mochila.
- Problema NP-hard: Problema de la parada.

### 9.5 Problema

Tenemos un problema A, un problema B y una caja negra NA y NB que resuelven el problema A y B respectivamente. Sabiendo que B es NP

### 9.5.1 I

Qué podemos decir de A si utilizamos NA para resolver el problema B (asumimos que la reducción realizada para adaptar el problema B al problema A es polinomial)

Podemos decir que A es igual o menos complejo que B.

### 9.5.2 II

Qué podemos decir de A si utilizamos NB para resolver el problema A (asumimos que la reducción realizada para adaptar el problema A al problema B es polinomial)

¿Por qué no podemos decir nada de A? ¿A quién le podemos pasar? ¿A qué clases puede (y no puede) pertenecer A? ¿Cómo es la relación de A con respecto a B?

Ya que A se puede resolver con NB sabemos que no es un problema mas complejo que NP. Sin embargo A puede pertenecer a cualquier complejidad menor a NP porque el que lo podamos resolver como resolveriamos un problema NP no significa que no exista un metodo mas simple y de menor complejidad algoritmica para llegar al mismo resultado. Lo que si podemos asegurar es que A es de igual o menor complejidad que B.

### 9.5.3 III

Qué pasa con los puntos anteriores si no conocemos la complejidad de B, pero sabemos que A es P?

Si “no necesariamente B se resuelve de forma polinómica”, ¿a qué clases puede pertenecer?

1. Decimos que A es igual o más complejo que B y ambos se pueden resolver de forma polinómica.
2. Decimos que B es igual o más complejo que A y no necesariamente B se resuelve de forma polinómica. B podría pertenecer a NP (o mayor) y que utilicemos un algoritmo de mayor complejidad para resolver un problema P.