Demostración Partition NP-Completo

¿Quiénes somos?

Javier Correa Marichal

alu0101233598@ull.edu.es

José Daniel Escánez Expósito

alu0101238944@ull.edu.es

Alejandro Peraza González

alu0101211770@ull.edu.es

Nerea Rodríguez Hernández

alu0101215693@ull.edu.es

Índice

PROBLEMA DEL EMPAREJAMIENTO TRIDIMENSIONAL

3DM

1

2

PROBLEMA DE LA PARTICIÓN

PARTITION

3

4

3DM → PARTITION

REDUCCIÓN

LENGUAJE DE PROGRAMACIÓN RUST

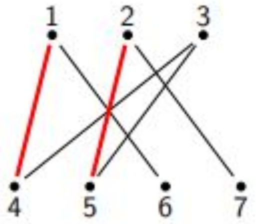
IMPLEMENTACIÓN

Problema del Emparejamiento Tridimensional

1

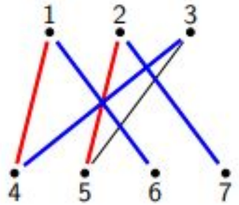
Problema del emparejamiento bidimensional

¿Es posible organizar un conjunto de forma que se evite que se repitan los elementos dentro de la t-upla?



$M_1 = \{(1,4),(2,5)\}$

$M_2 = \{(1,6),(2,7),(3,4)\}$



$M_1 = \{(1,4),(2,5)\}$

$M_2 = \{(1,6),(2,7),(3,4)\}$

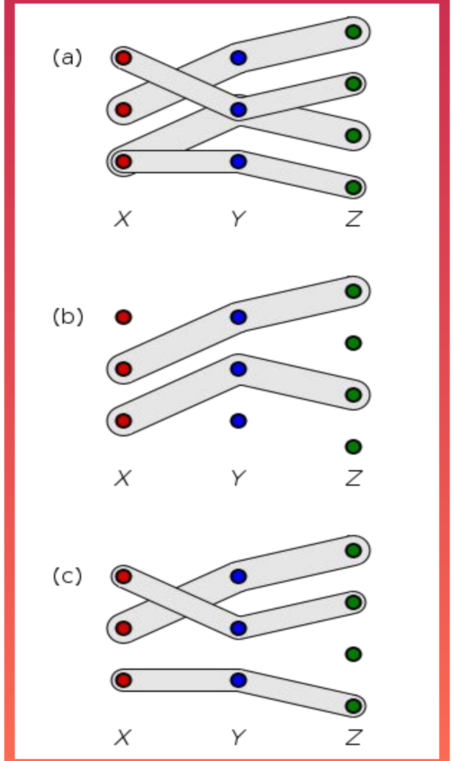
[The Bipartite Matching Problem - Math 482, Lecture 21 (illinois.edu)]

Problema del Emparejamiento Tridimensional

La figura (a) muestra el conjunto T.

La figura (b) muestra una coincidencia tridimensional M con | m | = 2.

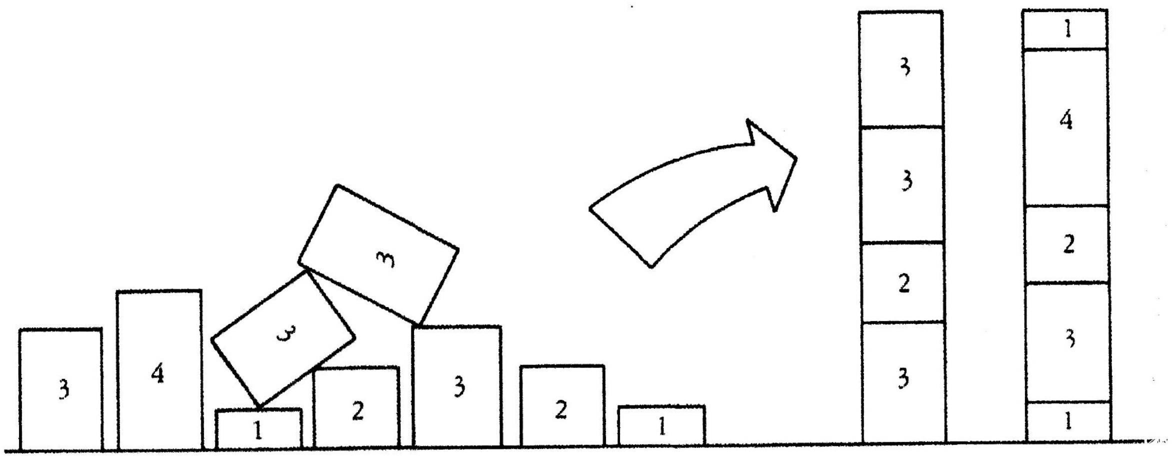
La figura (c) muestra una coincidencia tridimensional M con | m | = 3.



Problema de La Partición

2

Problema de la Partición



Reducción

3DM → PARTITION

3

Pasos Iniciales

Tenemos los conjuntos:

$$W = \{w_1, w_2, \dots, w_q\}$$
$$X = \{x_1, x_2, \dots, x_q\}$$
$$Y = \{y_1, y_2, \dots, y_q\}$$
$$M = \{m_1, m_2, \dots, m_k\}$$

Donde:

$$k = |M|$$

$$q = |W| = |X| = |Y|$$

Objetivo

Queremos un conjunto **A** y determinar tamaños s(a) para cada elemento.

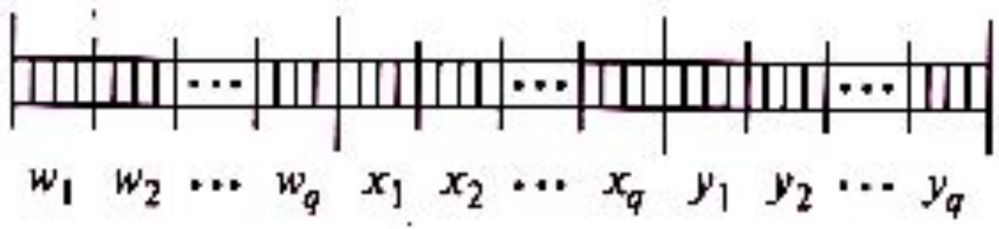
Tal que A tenga un subconjunto A’ donde:

$$\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$$

Representación en Binario

3q zonas de tamaño p

$$p = \lfloor log_2(k + 1) \rfloor$$



Ejemplo

	w1	w2	w3	w4	x1	x2	x3	x4	y1	y2	y3	y4	m_i
s(0)	X				X					X			{w1, x1, y2}
s(1)		X				X			X				{w2, x2, y1}
s(2)			X					X				X	{w3, x4, y4}
s(3)			X		X						X		{w3, x1, y3}
s(4)			X		X							X	{w3, x1, y4}
s(5)		X				X			X				{w2, x2, y1}
s(6)			X			X			X				{w3, x2, y1}

M = {{w1, x1, y2}, {w2, x2, y1}, {w3, x4, y4}, ..., {w3, x2, y1}}

Ejemplo

	w1	w2	w3	w4	x1	x2	x3	x4	y1	y2	y3	y4	Decimal
s(0)	00**1**	000	000	000	00**1**	000	000	000	000	00**1**	000	000	8592031808
s(1)	000	00**1**	000	000	000	00**1**	000	000	00**1**	000	000	000	1074004480
s(2)	000	000	00**1**	000	000	000	000	00**1**	000	000	000	00**1**	134221825
s(3)	000	000	00**1**	000	00**1**	000	000	000	000	000	00**1**	000	16809992
s(4)	000	000	00**1**	000	00**1**	000	000	000	000	000	000	00**1**	136314881
s(5)	000	00**1**	000	000	000	00**1**	000	000	00**1**	000	000	000	1074004480
s(6)	000	000	00**1**	000	000	00**1**	000	000	00**1**	000	000	000	134480384

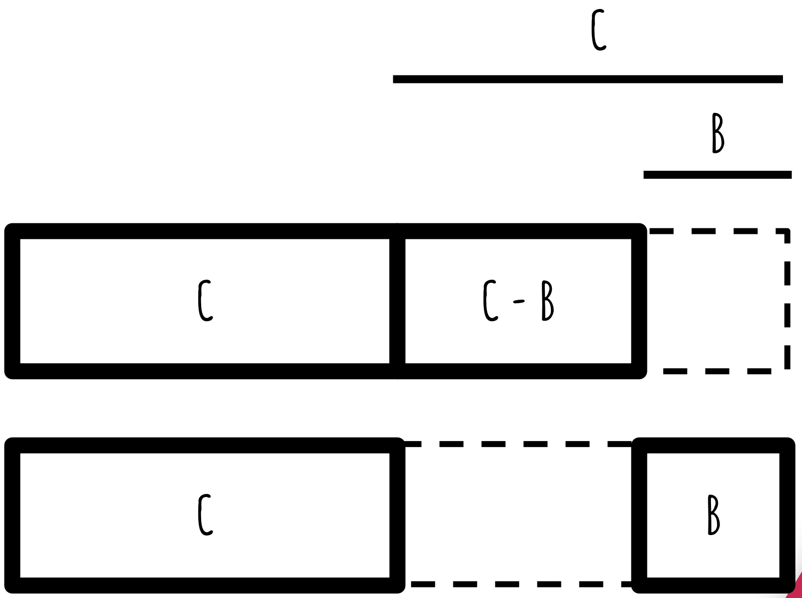
Ejemplo

	w1	w2	w3	w4	x1	x2	x3	x4	y1	y2	y3	y4	Decimal
s(0)	00**1**	000	000	000	00**1**	000	000	000	000	00**1**	000	000	8592031808
s(1)	000	00**1**	000	000	000	00**1**	000	000	00**1**	000	000	000	1074004480
s(2)	000	000	00**1**	000	000	000	000	00**1**	000	000	000	00**1**	134221825
s(3)	000	000	00**1**	000	00**1**	000	000	000	000	000	00**1**	000	16809992
s(4)	000	000	00**1**	000	00**1**	000	000	000	000	000	000	00**1**	136314881
s(5)	000	00**1**	000	000	000	00**1**	000	000	00**1**	000	000	000	1074004480
s(6)	000	000	00**1**	000	000	00**1**	000	000	00**1**	000	000	000	134480384
C	001	010	011	001	010	011	001	001	011	001	001	010	11161867850
B	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	00**1**	9817068105

Cálculo de b1 y b2

$$s(b1) = 2 * C - B$$

$$s(b2) = C + B$$



Implementación

4

Clase Partition

class Partition

```rust
use serde::{Deserialize, Serialize};

#[derive(Debug, Deserialize, Serialize)]
pub struct Partition {
    pub values: Vec<usize>,
}
```

partition.ron

```
(
    values: [
        16781313,
        2129984,
        262664,
        16810048,
        2101256,
        266241,
        266304,
        58061659,
        57791771,
    ],
)
```

Clase TDM

class TDM

```rust
use super::utility;
use serde::{Deserialize, Serialize};

#[derive(Debug, Deserialize, Serialize)]
pub struct TDM {
    cardinality: usize,
    m: Vec<(usize, usize, usize)>,
}
```

tdm.ron

```
TDM(
    cardinality: 3,
    m: [
        (1, 2, 3),
        (2, 1, 1),
        (3, 3, 2),
        (1, 1, 1),
        (2, 2, 2),
        (3, 2, 3),
        (3, 2, 1),
    ]
)
```

Reducción

class TDM

```
#[derive(Debug)]
struct BinaryVector {
    values: Vec<u8>,
    p: usize,
}

impl BinaryVector {
    fn new(n: usize, p: usize, (x, y, z): (usize, usize, usize)) -> Self {
        let mut values = vec![0; 3 * n * p];
        let groups = [x, y, z];
        for (index, item) in groups.iter().enumerate() {
            values[((item * p) - 1) + p * n * index] = 1;
        }
        Self { values, p }
    }

    fn empty(n: usize, p: usize) -> Self {
        let values = vec![0; 3 * n * p];
        Self { values, p }
    }

    fn create_b(n: usize, p: usize) -> Self {
        let mut values = vec![0; 3 * n * p];
        for index in 0..values.len() {
            if index % p == p - 1 {
                values[index] = 1;
            }
        }
        Self { values, p }
    }

    fn to_decimal(&self) -> usize {
        self.values.iter().fold(0, |acc, x| acc * 2 + *x as usize)
    }
}

fn add(self, second_summand: &BinaryVector) -> BinaryVector
```

Reducción

class TDM

```
pub fn tdm_to_partition(tdm: &TDM, verbose: bool) -> Partition {
    let p = ((tdm.get_m().len() + 1) as f64).log2().ceil() as usize;
    let mut binary_rows = vec![];
    for (index, tuple) in tdm.get_m().iter().enumerate() {
        binary_rows.push(BinaryVector::new(tdm.get_cardinality(), p,
*tuple));
    }
    let c = binary_rows
        .iter()
        .fold(BinaryVector::empty(tdm.get_cardinality(), p), |acc, x| {
            &acc + &x
        });

    let b = BinaryVector::create_b(tdm.get_cardinality(), p);
    let b1 = c.to_decimal() * 2 - b.to_decimal();
    let b2 = c.to_decimal() + b.to_decimal();

    let mut values: Vec<usize> = binary_rows.iter().map(|x|
x.to_decimal()).collect();
    values.push(b1);
    values.push(b2);
    Partition { values }
}
```

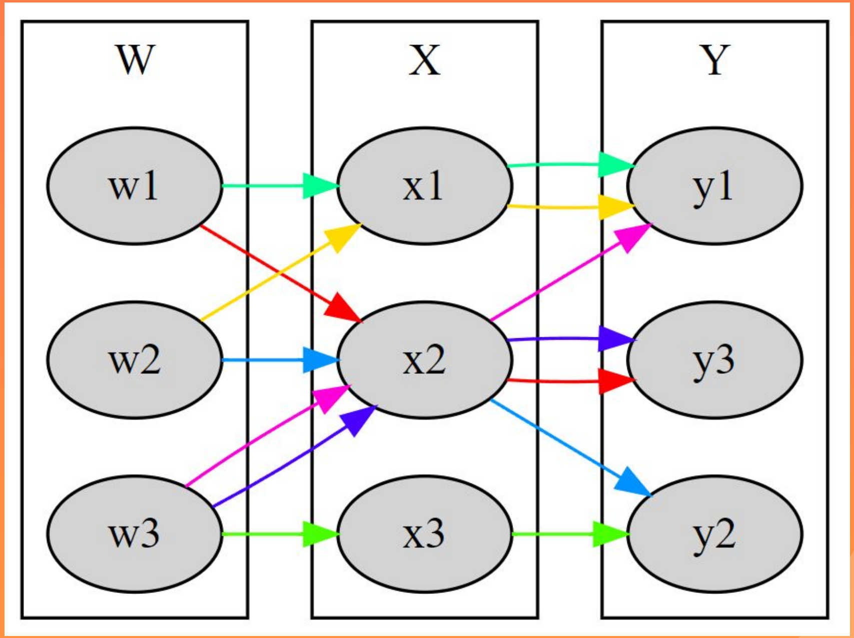
Representación de 3DM

```
digraph G {
    rankdir = LR;
    subgraph cluster_0 {
        node [style=filled];
        w1 w2 w3;
        label = "W";
    }

    subgraph cluster_1 {
        node [style=filled];
        x1 x2 x3;
        label = "X";
    }

    subgraph cluster_2 {
        node [style=filled];
        y1 y2 y3;
        label = "Y";
    }

    w1 -> x2 -> y3 [color="#ff0000"];
    w2 -> x1 -> y1 [color="#ffdb00"];
    w3 -> x3 -> y2 [color="#49ff00"];
    w1 -> x1 -> y1 [color="#00ff92"];
    w2 -> x2 -> y2 [color="#0092ff"];
    w3 -> x2 -> y3 [color="#4900ff"];
    w3 -> x2 -> y1 [color="#ff00db"];
}
```



Referencias

Lavrov, M. (2020, 25 marzo). The Bipartite Matching Problem. University of Illinois Urbana-Champaign, College of Liberal Arts & Sciences, Department of Mathematics. Recuperado 18 de enero de 2022, de https://faculty.math.illinois.edu/~mlavrov/slides/482-spring-2020/slides21.pdf

Wikipedia contributors. (2021a, octubre 16). Partition problem. Wikipedia. Recuperado 18 de enero de 2022, de https://en.wikipedia.org/wiki/Partition_problem

Wikipedia contributors. (2021b, noviembre 6). *3-dimensional matching*. Wikipedia. Recuperado 18 de enero de 2022, de https://en.wikipedia.org/wiki/3-dimensional_matching

Garey, M. R., & Johnson, D. S. (2000). Computers and Intractability: A Guide to the Theory of NP-completeness (22.a ed.). W. H. Freeman and Company.

¡Gracias!

¿Alguna pregunta?