

# Material de apoyo Teórica VII

## Temario

Seguimos trabajando con problemas combinatorios:

- Problemas de Cobertura de Conjuntos
  - Conjuntos a cubrir
  - Particionamiento
  - Packing
- Problemas de Secuenciamiento de tareas (Scheduling)

## **Problemas combinatorios**

- Son aquellos en los cuales se desea determinar combinaciones óptimas.
- Se caracterizan por tener un número finito de soluciones factibles.
- Generalmente este número es muy grande.

## **Problemas de cobertura de conjuntos**

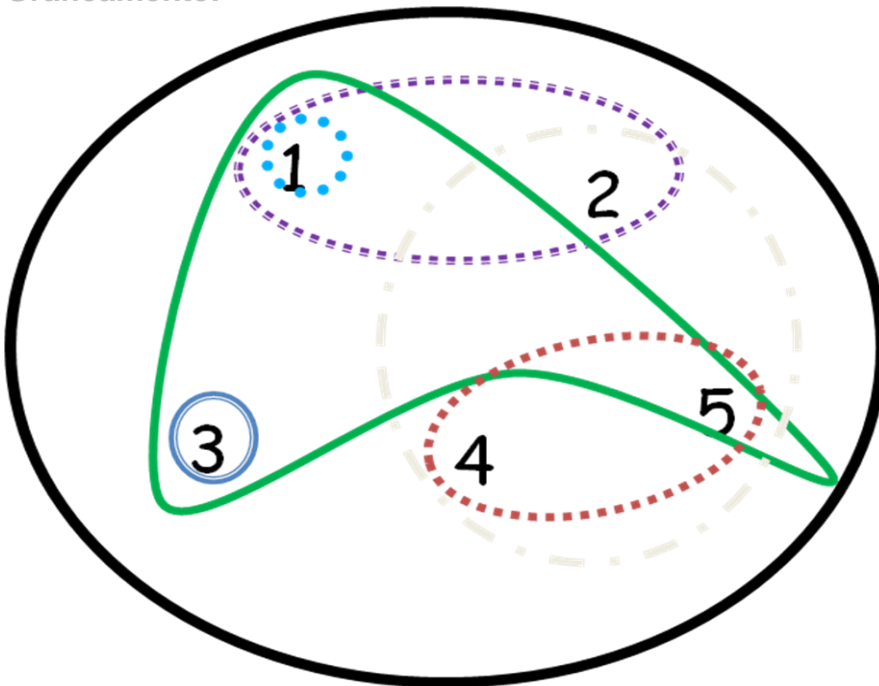
- Problemas de grupos que se deben cubrir.
- Problemas de grupos que se particionan.
- Problemas de "Packing"

### **Genéricamente**

- $S = \{1, 2, 3, \dots, n\}$  conjunto de elementos a cubrir.
- $L = \{(1, 2); (2); \dots\}$  conjunto formado por sub conjuntos de  $S$
- Elegir elementos de  $L$  tales que:
  - Cobertura: se cubran todos los elementos de  $S$  con solapamiento.
  - Partición: se cubran todos los elementos de  $S$  sin solapamiento.
  - Packing: se cubra la máxima cantidad de elementos de  $S$  que se pueda sin solapamiento

**Un ejemplo:**

- Una compañía aérea debe cubrir el vuelo a 5 ciudades (1,2,3,4,5) desde la ciudad cero
- Se han definido 6 posibles circuitos y para cada uno se indica qué ciudades toca: A - (1,2); B - (1,3,5); C - (2,4,5); D - (3); E - (1); F - (4,5)
- Cada tripulación está asignada a un circuito.
- Se busca minimizar la cantidad de tripulaciones.

**Gráficamente:**

En el conjunto S (ciudades) hemos marcado los subconjuntos (circuitos) que forman L

**Esquema general**

- Entre los subconjuntos que conforman L (los circuitos) tenemos que elegir algunos de manera de cubrir S, de acuerdo con los criterios del problema que estemos tratando (cobertura, particionamiento o packing)
- Para eso definimos una variable bivalente por cada circuito
- $Y_i$ : vale 1 si se realiza el circuito i,  
vale cero sino

**Problemas de grupos que se deben cubrir**

- ✓ Se debe determinar cuáles circuitos se realizarán, de modo tal que cada una de las cinco ciudades sea cubierta por al menos un circuito (puede haber una o más ciudades que sean cubiertas por más de un circuito, a esto se le llama solapamiento)

MIN  $YA + YB + YC + YD + YE + YF$

ST

!Deben estar cubiertas todas las ciudades al menos una vez, por lo tanto se debe

!elegir al menos un circuito de los que pasan por cada ciudad

CIUDAD1)  $YA + YB + YE \geq 1$

CIUDAD2)  $YA + YC \geq 1$

CIUDAD3)  $YB + YD \geq 1$

CIUDAD4)  $YC + YF \geq 1$

CIUDAD5)  $YB + YC + YF \geq 1$

- En este caso, como son pocas ciudades el problema se puede resolver a mano. Vemos que con solamente dos circuitos se cubren todas las ciudades (con el B y el C)
- Hay un solapamiento en la ciudad 5 (los dos circuitos la cubren) pero el planteo del problema lo permite
- Desde el punto de vista de conjuntos, sería elegir elementos de L tal que la unión de los subconjuntos de L elegidos me dé el conjunto S original

**Problemas de grupos que se particionan**

- ✓ Siguiendo el ejemplo anterior, se trata ahora de cubrir todas las ciudades utilizando los mismos 6 circuitos pero sin “solapamiento”.

MIN  $YA + YB + YC + YD + YE + YF$

ST

!Deben estar cubiertas todas las ciudades exactamente una vez, por lo tanto se

!debe elegir exactamente un circuito de los que pasan por !cada ciudad

CIUDAD1)  $YA + YB + YE = 1$

CIUDAD2)  $YA + YC = 1$

CIUDAD3)  $YB + YD = 1$

CIUDAD4)  $YC + YF = 1$

CIUDAD5)  $YB + YC + YF = 1$

- En este caso, como son pocas ciudades el problema se puede resolver a mano. Vemos que en este caso se necesitan tres circuitos (el problema al tener signo de igual es más restrictivo que el de grupos a cubrir que era de mayor o igual). Una posible solución es elegir los circuitos A, D y F ¿podés

encontrar otra con tres circuitos?

- Muchas veces este tipo de planteo no tiene solución, porque los datos no permiten cubrir sin solapamiento
- Desde el punto de vista de conjuntos, sería elegir elementos de  $L$  tal que la unión de los subconjuntos de  $L$  elegidos me dé el conjunto  $S$  original y que la intersección entre los subconjuntos de  $L$  elegidos sea el conjunto vacío

### Problemas de “Packing”

- ✓ Ahora se trata de “cubrir” la mayor cantidad de elementos que se pueda, sin solapamiento (no existe la obligación de cubrirlos todos).
- ✓ Muchas veces se plantea este problema porque el de partición no tiene solución (entonces, si no podemos cubrir todos los elementos, vamos a cubrir la máxima cantidad que podamos)

!En este caso hay que tener un funcional de máximo porque nos piden elegir la !mayor cantidad posible de elementos de  $L$

MAX  $Y_A + Y_B + Y_C + Y_D + Y_E + Y_F$

ST

!No es necesario que todas las ciudades estén obligatoriamente cubiertas, pero

!ninguna debe estar sobrecubierta (no hay solapamiento)

CIUDAD1)  $Y_A + Y_B + Y_E \leq 1$

CIUDAD2)  $Y_A + Y_C \leq 1$

CIUDAD3)  $Y_B + Y_D \leq 1$

CIUDAD4)  $Y_C + Y_F \leq 1$

CIUDAD5)  $Y_B + Y_C + Y_F \leq 1$

- En este caso, la solución es la misma que en el caso de conjuntos que se particionan (si el problema de conjuntos que se particionan tiene solución, la solución del problema de packing es la misma)
- Un ejemplo de este tipo de problema podría ser tratar de ocupar la máxima cantidad de Cocheras en un playa de estacionamiento (no se puede poner más de un auto por cochera y dependiendo de las condiciones del problema me pueden quedar algunas cocheras vacías).

### Otro planteo

- Con el planteo del funcional que hicimos puede visitar muchos circuitos de pocas ciudades cada uno.
- Si queremos que visite la mayor cantidad de ciudades ahora sí tiene sentido tener una bivalente por ciudad
- $V_i$  vale 1 si se visita la ciudad  $i$  y vale 0 sino

$$\text{MAX } V1 + V2 + V3 + V4 + V5$$

ST

!No es necesario que todas las ciudades estén obligatoriamente cubiertas, pero

!ninguna debe estar sobrecubierta (no hay solapamiento)

$$\text{CIUDAD1) } YA + YB + YE = V1$$

$$\text{CIUDAD2) } YA + YC = V2$$

$$\text{CIUDAD3) } YB + YD = V3$$

$$\text{CIUDAD4) } YC + YF = V4$$

$$\text{CIUDAD5) } YB + YC + YF = V5$$

### Genéricamente:

- ▶  $S = \{1, 2, 3, \dots, n\}$  conjunto de elementos a cubrir.
- ▶  $L = \{(1, 2); (2); \dots\}$  conjunto formado por sub conjuntos de  $S$
- ▶ Elegir elementos de  $L$  tales que:
  - Cobertura: se cubran todos los elementos de  $S$  con solapamiento.
  - Partición: se cubran todos los elementos de  $S$  sin solapamiento.
  - Packing: se cubra la máxima cantidad de elementos de  $S$  que se pueda sin solapamiento

## Calendarización (Scheduling)

### Introducción

El estudio formal de los problemas de calendarización se remonta a la década del 50'. En ese entonces, industrias de diferente índole intentaban resolver el problema de cómo llevar adelante la realización de múltiples tareas teniendo recursos (máquinas) limitados. Oliver Wight identifica dos conceptos clave en lo que refiere a los problemas de calendarización orientados a la producción: prioridades y capacidades. En definitiva, qué deberá hacerse primero y quién debería hacerlo. La respuesta a estas dos preguntas podía marcar la diferencia entre una empresa competitiva y una que no lo era.

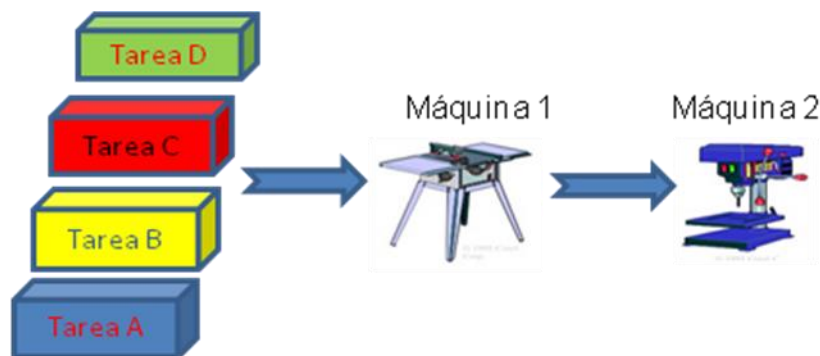
El hecho de que sea un problema al que se le encontraban tantas aplicaciones prácticas, impulsó su estudio y la aplicación de distintas técnicas para resolverlo. En la década del 70' se demostró que ciertos problemas de calendarización, pertenecen al grupo de problemas NP-Hard. Debido al costo (y en algunos casos incluso la imposibilidad) de obtener soluciones óptimas, se prepararon heurísticas que aunque no conseguían la solución óptima, encontraban soluciones suficientemente buenas para lo que se necesitaba.

### Ejemplo de problema de calendarización

Supongamos la existencia de  $N$  tareas a realizar, cada una con un tiempo de procesamiento que depende de la tarea. Cada una de las tareas se puede realizar en cualquiera de las  $M$  máquinas. No existen restricciones en lo que respecta a la precedencia de las tareas. El objetivo sería definir en qué momento se debe procesar cada tarea y en qué máquina, para poder completar todas las tareas lo antes posible. Es decir, que la tarea que finaliza último, termine lo antes posible. Una de las técnicas matemáticas que se puede utilizar para la resolución de este problema es la programación lineal.

### Un ejemplo práctico

Trabajaremos con un caso en el cual tenemos cuatro tareas (A, B, C y D) que tienen que pasar primero por la máquina 1 y luego por la máquina 2.



Los tiempos de procesamiento de cada tarea en las máquinas son:

Tarea	Tiempo en máquina 1	Tiempo en máquina 2
A	3 minutos	2 minutos
B	6 minutos	8 minutos
C	5 minutos	6 minutos
D	7 minutos	4 minutos

Cuando se empieza a procesar una tarea en una máquina no se interrumpe (sigue hasta que se termine)

Queremos tardar lo menos posible en terminar todas las tareas

¿Y si lo planteamos así?:

$$\text{Tiempo máquina 1} = 3 + 6 + 5 + 7 = 21 \text{ minutos}$$

$$\text{Tiempo máquina 2} = 2 + 8 + 6 + 4 = 20 \text{ minutos}$$

¡En 21 minutos terminamos!

Pero no estamos asegurando que una tarea se procese en la máquina 2 cuando ya se haya terminado de procesar en la máquina 1

Necesitamos saber en qué momento empieza cada tarea en cada máquina (y en qué momento termina, para que pueda pasar a la otra máquina):

**Iij:** Minuto en que empieza la tarea i en la máquina j

**Fij:** Minuto en el cual finaliza la tarea i en la máquina j

(como las tareas no se interrumpen  $Fij = Iij + \text{Tiempo de la tarea i en máquina j}$ )

Por ejemplo, para la tarea A:

$$FA1 = IA1 + 3$$

$$FA2 = IA2 + 2$$

Y así para las demás tareas (B, C y D)

Para que una tarea no empiece en la máquina 2 antes de haber finalizado su procesamiento en la máquina 1 podemos hacer:

Por ejemplo, para la tarea A:

$$FA1 \leq IA2$$

Y así para las demás tareas (B, C y D)

Para saber cuál es la tarea que termina más tarde podemos definir:

**FINAL:** Minuto en el cual finaliza la última tarea

$$FA2 \leq \text{FINAL}$$

$$FB2 \leq \text{FINAL}$$

$$FC2 \leq \text{FINAL}$$

$$FD2 \leq \text{FINAL}$$

Y la función objetivo es  $\text{MIN FINAL}$

Resumiendo, nuestro modelo sería:

$$Fi1 = li1 + \text{Tiempo en máquina } q1$$

$$Fi2 = li2 + \text{Tiempo en máquina } q2$$

Para  $i = A, B, C, D$

$$Fi1 \leq li2$$

Para  $i = A, B, C, D$

$$FA2 \leq \text{FINAL}; \quad FB2 \leq \text{FINAL}; \quad FC2 \leq \text{FINAL}; \quad FD2 \leq \text{FINAL}$$

$$\text{MIN FINAL}$$

Vamos a resolverlo con software y vemos cuál es la solución óptima:



MIN FINAL	LP OPTIMUM FOUND AT STEP 5
ST	OBJECTIVE FUNCTION VALUE
FA1 - IA1 = 3	1) 14.000000
FA2 - IA2 = 2	
FB1 - IB1 = 6	VARIABLE VALUE
FB2 - IB2 = 8	FINAL 14.000000
FC1 - IC1 = 5	FA1 3.000000
FC2 - IC2 = 6	IA1 0.000000
FD1 - ID1 = 7	FA2 5.000000
FD2 - ID2 = 4	IA2 3.000000
FA1 - IA2 < 0	FB1 6.000000
FB1 - IB2 < 0	IB1 0.000000
FC1 - IC2 < 0	FB2 14.000000
FD1 - ID2 < 0	IB2 6.000000
FA2 - FINAL < 0	FC1 5.000000
FB2 - FINAL < 0	IC1 0.000000
FC2 - FINAL < 0	FC2 14.000000
FD2 - FINAL < 0	IC2 8.000000
END	FD1 7.000000
	ID1 0.000000
	FD2 14.000000
	ID2 10.000000

### ¡INICIA TODAS LAS TAREAS AL MISMO TIEMPO EN LA MÁQUINA 1!

¿Cómo podemos hacer para que no haga dos tareas al mismo tiempo en la misma máquina?

Por ejemplo, en la máquina 1, o hace la tarea A antes que la B, o hace la tarea B antes que la A

$$FA1 \leq IB1$$

ó

$$FB1 \leq IA1$$

¿Cómo hacemos para que de las dos restricciones el modelo anule una de las dos y deje la otra activa? Lo vimos hace dos semanas

$$FA1 \leq IB1 + M \text{YanuloAB}$$

$$FB1 \leq IA1 + M \text{YanuloBA}$$

$$\text{YanuloAB} + \text{YanuloBA} = 1$$

Entonces tenemos que agregar en el modelo el siguiente grupo de restricciones:

Para todas las tareas en la máquina 1:

$$Fi1 \leq Ik1 + M \text{Yanuloik}$$

$$Fk1 \leq li1 + M \text{Yanuloki}$$

$$\text{Yanuloik} + \text{Yanuloki} = 1$$

Para todo par de tareas i k

Y lo mismo en la máquina 2:

$$Fi2 \leq Ik2 + M \text{Yanulo2ik}$$

$$Fk2 \leq li2 + M \text{Yanulo2ki}$$

$$\text{Yanulo2ik} + \text{Yanulo2ki} = 1$$

Para todo par de tareas i k

**El modelo, en el software, quedaría así (recordemos que como no existe el signo de mayor o menor estricto, cuando ponemos menor el software interpreta que es menor o igual):**

MIN FINAL	FA1 - IB1 - 30 Y1AB < 0	INT Y1AB
ST	FB1 - IA1 - 30 Y1BA < 0	INT Y1BA
FA1 - IA1 = 3	Y1AB + Y1BA = 1	INT Y1AC
FA2 - IA2 = 2	FA1 - IC1 - 30 Y1AC < 0	INT Y1CA
FB1 - IB1 = 6	FC1 - IA1 - 30 Y1CA < 0	INT Y1AD
FB2 - IB2 = 8	Y1AC + Y1CA = 1	INT Y1DA
FC1 - IC1 = 5	FA1 - ID1 - 30 Y1AD < 0	INT Y1BC
FC2 - IC2 = 6	FD1 - IA1 - 30 Y1DA < 0	INT Y1CB
FD1 - ID1 = 7	Y1AD + Y1DA = 1	INT Y1BD
FD2 - ID2 = 4	FB1 - IC1 - 30 Y1BC < 0	INT Y1DB
FA1 - IA2 < 0	FC1 - IB1 - 30 Y1CB < 0	INT Y1CD
FB1 - IB2 < 0	Y1BC + Y1CB = 1	INT Y1DC
FC1 - IC2 < 0	FB1 - ID1 - 30 Y1BD < 0	INT Y2AB
FD1 - ID2 < 0	FD1 - IB1 - 30 Y1DB < 0	INT Y2BA
FA2 - FINAL < 0	Y1BD + Y1DB = 1	INT Y2AC
FB2 - FINAL < 0	FC1 - ID1 - 30 Y1CD < 0	INT Y2CA
FC2 - FINAL < 0	FD1 - IC1 - 30 Y1DC < 0	INT Y2AD
FD2 - FINAL < 0	Y1CD + Y1DC = 1	INT Y2DA
	FA2 - IB2 - 30 Y2AB < 0	INT Y2BC
	FB2 - IA2 - 30 Y2BA < 0	INT Y2CB
	Y2AB + Y2BA = 1	INT Y2BD
	FA2 - IC2 - 30 Y2AC < 0	INT Y2DB
	FC2 - IA2 - 30 Y2CA < 0	INT Y2CD
	Y2AC + Y2CA = 1	INT Y2DC
	FA2 - ID2 - 30 Y2AD < 0	
	FD2 - IA2 - 30 Y2DA < 0	
	Y2AD + Y2DA = 1	
	FB2 - IC2 - 30 Y2BC < 0	
	FC2 - IB2 - 30 Y2CB < 0	
	Y2BC + Y2CB = 1	
	FB2 - ID2 - 30 Y2BD < 0	
	FD2 - IB2 - 30 Y2DB < 0	
	Y2BD + Y2DB = 1	
	FC2 - ID2 - 30 Y2CD < 0	
	FD2 - IC2 - 30 Y2DC < 0	
	Y2CD + Y2DC = 1	
	END	

Y ahora lo vamos a volver a resolver para ver si ahora nos coloca las tareas en las máquinas de manera que no se superpongan dos tareas en la misma máquina:

OBJECTIVE FUNCTION VALUE		FINAL	
1)	25.000000	FA1	14.000000
VARIABLE	VALUE	IA1	11.000000
Y1AB	1.000000	FA2	21.000000
Y1BA	0.000000	IA2	19.000000
Y1AC	1.000000	FB1	11.000000
Y1CA	0.000000	IB1	5.000000
Y1AD	0.000000	FB2	19.000000
Y1DA	1.000000	IB2	11.000000
Y1BC	1.000000	FC1	5.000000
Y1CB	0.000000	IC1	0.000000
Y1BD	0.000000	FC2	11.000000
Y1DB	1.000000	IC2	5.000000
Y1CD	0.000000	FD1	21.000000
Y1DC	1.000000	ID1	14.000000
Y2AB	1.000000	FD2	25.000000
Y2BA	0.000000	ID2	21.000000
Y2AC	1.000000		
Y2CA	0.000000		
Y2AD	0.000000		
Y2DA	1.000000		
Y2BC	1.000000		
Y2CB	0.000000		
Y2BD	0.000000		
Y2DB	1.000000		
Y2CD	0.000000		
Y2DC	1.000000		
FINAL	25.000000		
FA1	14.000000		
IA1	11.000000		
FA2	21.000000		
IA2	19.000000		
FB1	11.000000		
IB1	5.000000		
FB2	19.000000		
IB2	11.000000		
FC1	5.000000		
IC1	0.000000		
FC2	11.000000		
IC2	5.000000		
FD1	21.000000		
ID1	14.000000		
FD2	25.000000		
ID2	21.000000		

Entonces no superpone y termina en 25 minutos.

## Notación

Tan grande es el conjunto de problemas de calendarización, que incluso se desarrolló una notación para identificarlos. Los problemas de calendarización se pueden especificar de la forma

$$\alpha | \beta | \gamma$$

El término  $\alpha$  representa el entorno de las máquinas. Por ejemplo, si existiera una sola máquina se indicaría con un **1**. Si existieran 3 máquinas idénticas, se marcaría **P3**.

El segundo término,  $\beta$ , indica las características de las tareas. Si el trabajo en una tarea se pudiera interrumpir para que pueda ser continuado más adelante, agregaríamos **pmtn** en este término. Si existieran precedencias también serían detalladas en este campo. Es necesario mencionar que las características no son excluyentes. Es decir, podría armarse por ejemplo un problema de calendarización con precedencias y con interrupción de tareas.

Por último, en el término  $\gamma$  se incluye la función objetivo a resolver. La función objetivo no necesariamente será completar la última tarea lo antes posible (denominada **Cmax**). Constituyen otros ejemplos de funciones objetivo:

- Minimizar la finalización de todas las tareas ponderadas por un peso.
- Si se estableciera una fecha de finalización esperada para cada tarea, podría minimizarse la demora o la cantidad de tareas

La notación correspondiente al problema identificado en la sección 1 es:

$$Pm \mid \mid Cmax$$

## Aplicaciones

Ya se describió que la industria fue la que impulsó el estudio de los problemas de calendarización, aplicándolo principalmente para el caso en el cual los recursos eran máquinas. No obstante, con el tiempo el problema fue extendiéndose a nuevas áreas. El libro Handbook of Scheduling compila varias de estas aplicaciones[1].

Algunos de los ejemplos son:

- Asignación de horarios: El problema de asignar horarios a docentes que dan clases en un organismo educativo puede ser considerado como un problema de calendarización. Además de la complejidad intrínseca

asociada al aspecto matemático del problema, existen temas que tienen que ver con el hecho de que los recursos, no son máquinas sino que son humanos. No siempre se podrá satisfacer a todos los docentes con los horarios que ellos soliciten. En consecuencia, el hecho de que puede ser necesario llevar adelante una especie de negociación es un factor que se debe tener en cuenta.

- Industria aérea: Varios de los problemas que tienen que resolver las aerolíneas podrían catalogarse dentro del mundo de scheduling. Lo que distingue a esta industria es que muchas de las decisiones que debe tomar están interrelacionadas entre sí. Tiene que definir vuelos para satisfacer la demanda de pasajeros y asignar tripulaciones a estos vuelos, siempre teniendo en consideración las regulaciones legales y otras restricciones existentes. A diferencia del estudio de calendarización en máquinas donde se pudo por ejemplo definir una notación, el estudio en la industria de las aerolíneas esta menos estandarizado debido a la imposibilidad de generar un modelo común a estos problemas.

## Referencias Scheduling

1. LEUNG, Joseph et al. Handbook of Scheduling. 2004.
2. HERRMANN, Jeffrey et al. Handbook of Production Scheduling. 2006.
3. GRAHAM, Ronald et al. Optimization and approximation in deterministic sequencing and scheduling. 1979.
4. SOTSKOV, Yu y SHAKHLEVICH N.V. NP-hardness of shop-scheduling problems with three jobs. 1989.

¿Qué nos queda de esta clase?

- ☐ Seguimos trabajando con problemas combinatorios que vamos a encontrar en la Guía 3 (enteras)
- ☐ Cómo modelizar problemas combinatorios de la familia de Cobertura de Conjuntos:
  - ☐ Conjuntos a cubrir
  - ☐ Partición de conjuntos
  - ☐ Packing
- ☐ Secuenciamiento de tareas – Calendarización - Scheduling