Universidad de Oviedo

Intelligent Scheduling and Optimization

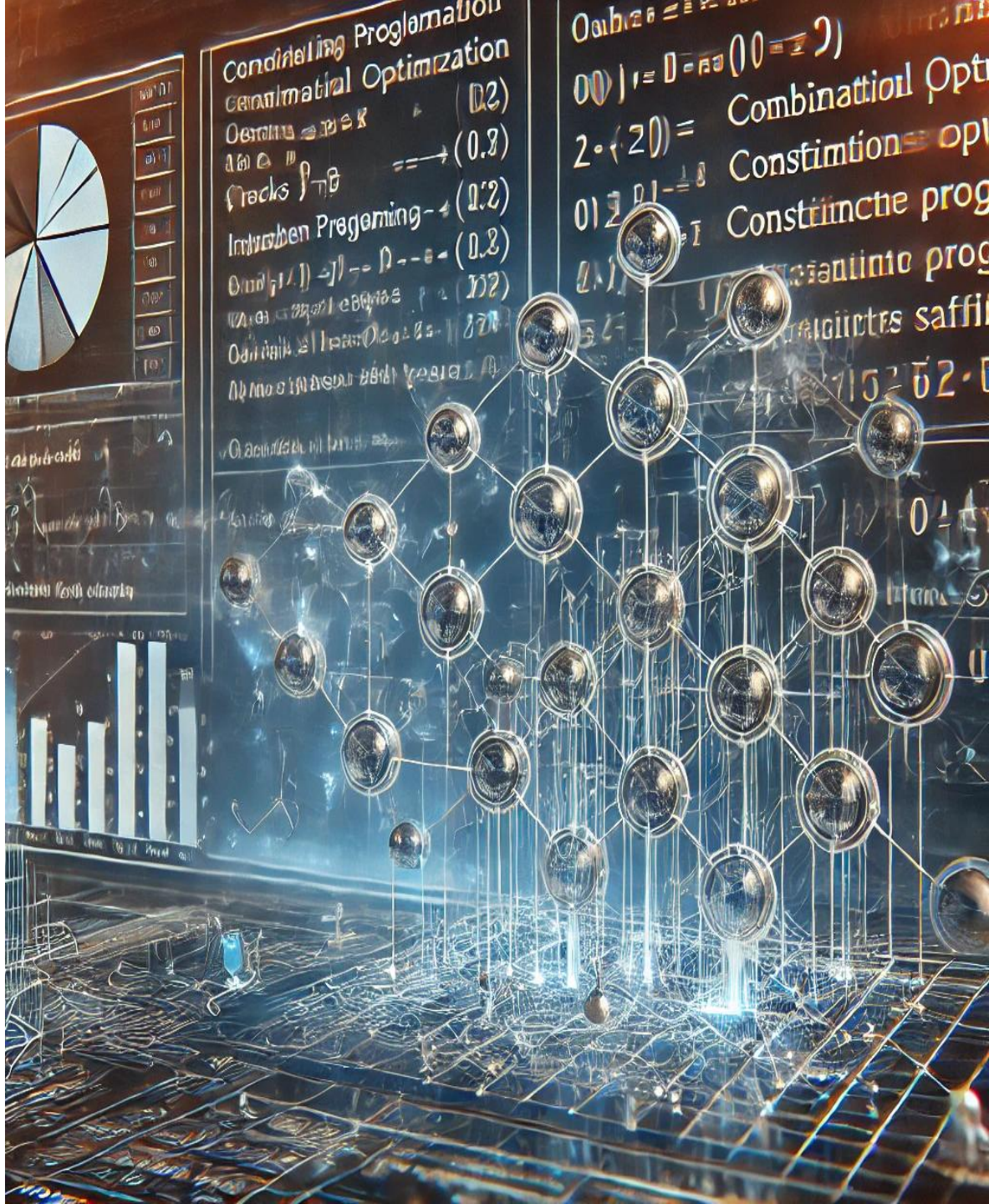
Técnicas de Inteligencia Artificial para la Optimización y Programación de Recursos

Tema 2: Introducción a los problemas de Scheduling

María Rita Sierra Sánchez
{sierramaria}@uniovi.es

Ciencia de la Computación e Inteligencia Artificial
Departamento de Informática

ceisIA Centro de Estudios sobre el Impacto Social de la Inteligencia Artificial | Universidad de Oviedo



Planificación vs Scheduling

La planificación se centra en la secuenciación de acciones para conseguir un objetivo

El Scheduling pone más énfasis en la utilización eficiente de los recursos disponibles por parte de las acciones

Contenidos

Problema de Scheduling

1. **Introducción al Scheduling**
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. Problema Job Shop Scheduling Problem
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

Introducción

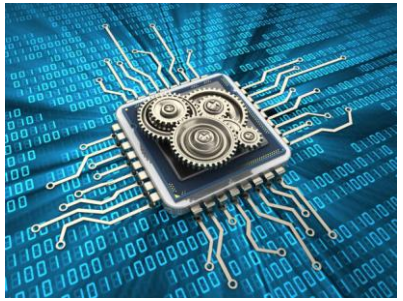
- *El Scheduling es un proceso de toma de decisiones que aparece frecuentemente en muchos entornos industriales y productivos.*
- *Aborda la asignación de recursos a trabajos (o tareas en las que estos se dividen) a lo largo de un periodo de tiempo, buscando optimizar uno o varios objetivos.*

Recursos (máquinas)

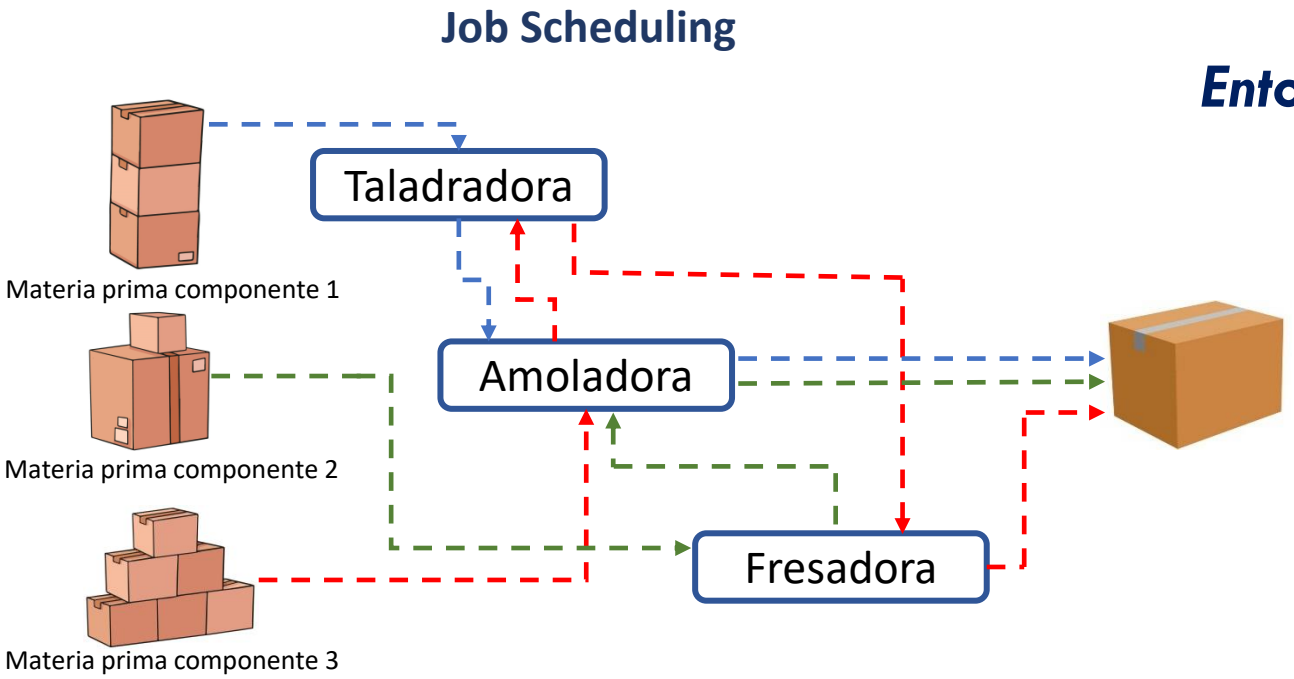
- Máquinas en un taller
- Pistas de un aeropuerto
- Cuadrillas en una obra
- Unidades de procesamiento en un entorno informático

Tareas (trabajos)

- Operaciones o trabajos en un taller
- Despegues y aterrizajes de aviones
- Tareas de un proyecto de construcción
- Programas informáticos



Introducción

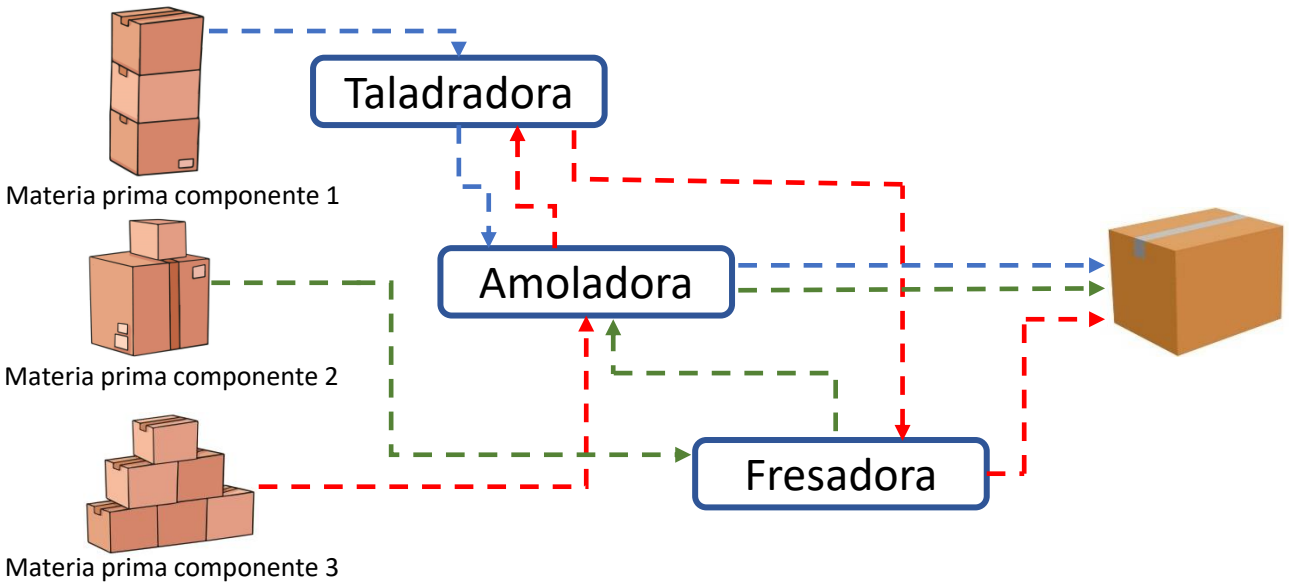


Entorno productivo

- ***Trabajo:*** *el item que se quiere producir (desde la materia prima hasta el producto terminado)*
- ***Recurso:*** *las máquinas.*
- *El* ***objetivo*** *del Scheduling es minimizar el tiempo y coste de la producción, diciéndole a la empresa cuándo fabricar, con qué personal y en qué equipo.*

Introducción

Job Scheduling



Trabajos

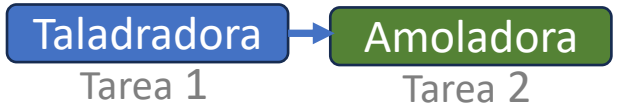
- Componente 1
- Componente 2
- Componente 3

Máquinas

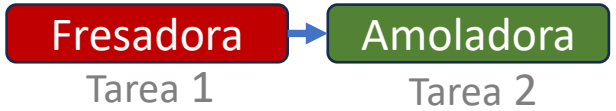
- Taladradora
- Fresadora
- Amoladora

Tareas

- Trabajo 1: Componente 1



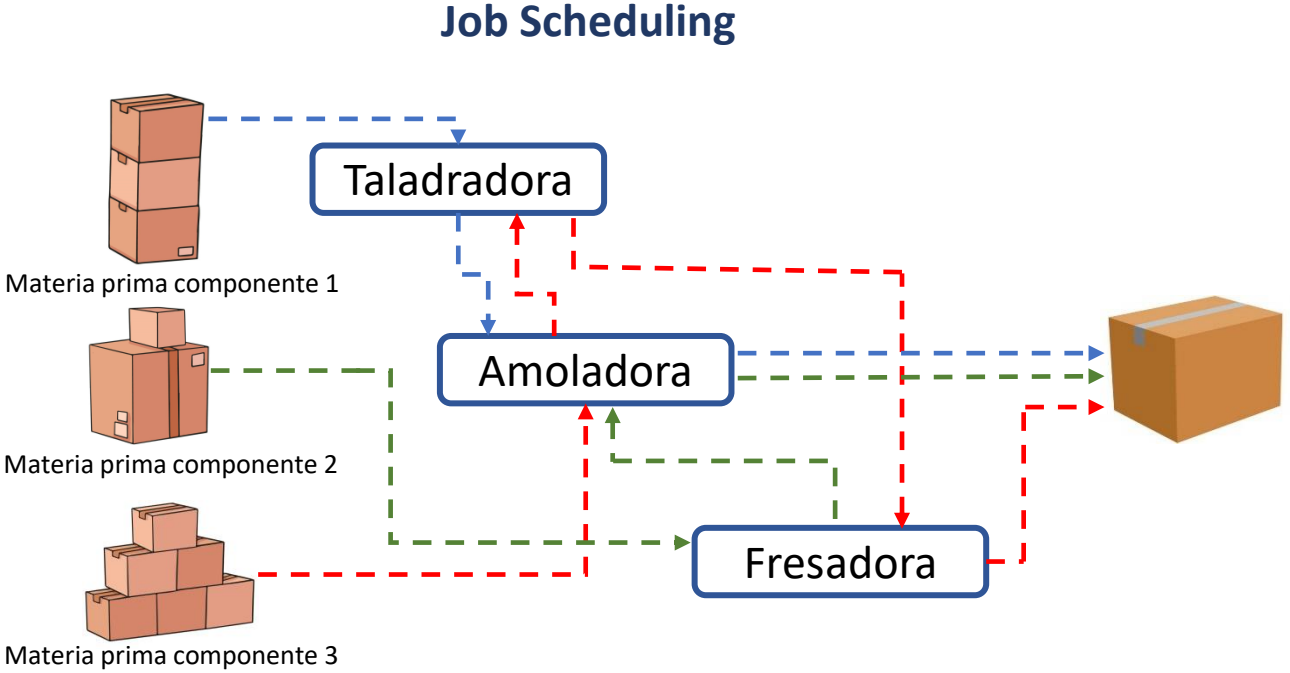
- Trabajo 2: Componente 2



- Trabajo 3: Componente 3

Introducción

Los problemas de Scheduling son problemas MUY complejos, cuya resolución, en general, requiere resolver tres subproblemas:



- ***Asignación de recursos a tareas.***
  - *Si cada tarea sólo puede ser procesada por un recurso específico se simplifica.*
- ***Secuenciación de tareas****, es decir determinar el orden en que se procesan las tareas, también dentro de cada recurso.*
- ***Scheduling****, establecer tanto la planificación como el uso de los recursos*

Contenidos

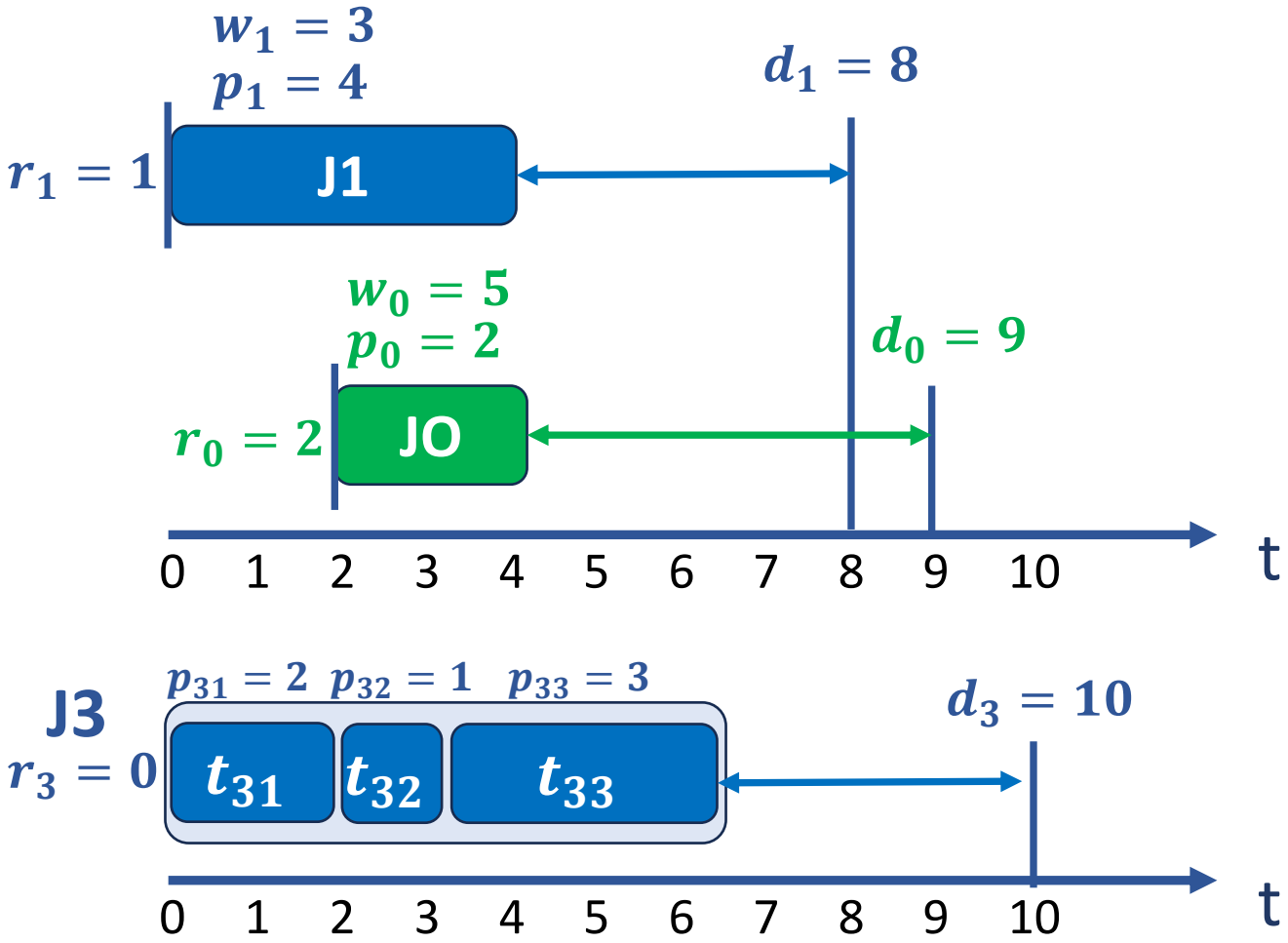
Problema de Scheduling

1. Introducción al Scheduling
2. **Conceptos Básicos**
3. Clasificación Problemas de Scheduling
4. Problema Job Shop Scheduling Problem
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

Conceptos Básicos

- En los problemas de Scheduling existen dos conceptos muy importantes:
  - ***Trabajo:*** Operación o conjunto de operaciones que han de ser procesadas para lograr una meta.
  - ***Máquina o recurso:*** Es cada uno de los entes encargados de realizar o procesar alguna de las tareas u operaciones pertenecientes a un trabajo.
- En los problemas de Scheduling, el número de trabajos y de máquinas son considerados finitos.
- En estos problemas:
  - Tenemos n trabajos, empleándose el subíndice i $(i = 1, \dots, n)$ para referirse a cada uno de ellos.
  - Cada trabajo i puede consistir en una única tarea o en una serie de tareas t_{ij} $(j = 1, \dots, n_i)$.
  - Tenemos m máquinas, $\{M_1, M_2, \dots, M_m\}$ en las que procesar los trabajos/tareas.
- Los problemas de Scheduling suelen expresar sus objetivos en términos de una función de coste que mide el coste que supone que un trabajo sea completado en un instante de tiempo t.

Conceptos Básicos



Para cada trabajo i podremos definir:

- Tareas
- Tiempo de procesamiento ("processing time"):
  - p_i si el trabajo sólo es procesado en una única máquina, o si el tiempo de procesamiento es independiente de la máquina.
  - p_{ij} tiempo de procesamiento de la tarea j del trabajo i, o también puede verse como el tiempo de procesamiento en la máquina requerida por la tarea i en el trabajo j.
- Fecha de lanzamiento ("release date"): r_j, instante en el que el trabajo j está disponible para ser procesado.
- Fecha de vencimiento ("due date"): d_j, instante de tiempo en el que el trabajo j debería, idealmente, haber terminado de ser procesado.
- Peso o prioridad: w_j, indica la importancia del trabajo j respecto al resto de trabajos.

En general, p_j , p_{ij} , r_j , d_j y w_j son variables enteras.

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. **Clasificación Problemas de Scheduling**
4. Problema Job Shop Scheduling Problem
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

Clasificación de problemas de Scheduling

- *Existen multitud de problemas de scheduling.*
- *Teniendo en cuenta los ambientes o configuraciones de las máquinas se pueden especificar como sistemas de una etapa o sistemas multietapa.*

Sistemas de una etapa

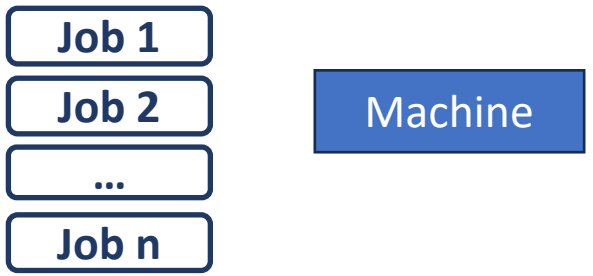
- Una única máquina $(m = 1)$, cada trabajo ha de ser procesado por la máquina exactamente una vez.
- Varias máquinas en paralelo $\mathrm{M} = \{M_1, M_2, \dots, M_m\}$, cada trabajo puede ser procesado por cualquier máquina, pero sólo una vez.
- El tiempo de procesamiento de un trabajo en una máquina puede tener relación o no con la velocidad de la misma.

Sistemas multietapa

- Cada trabajo debe ser procesado en cada máquina del conjunto de máquinas $\mathrm{M} = \{M_1, M_2, \dots, M_m\}$ al menos una vez.
- Todas las máquinas son diferentes.

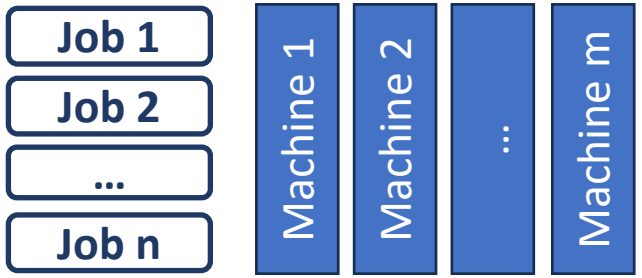
Clasificación de problemas de Scheduling

Sistemas de una etapa: 1 máquina o varias máquinas en paralelo



1, única máquina dedicada

- $p_{ij} = p_i$ =tiempo de procesamiento del trabajo i en la única máquina que hay.

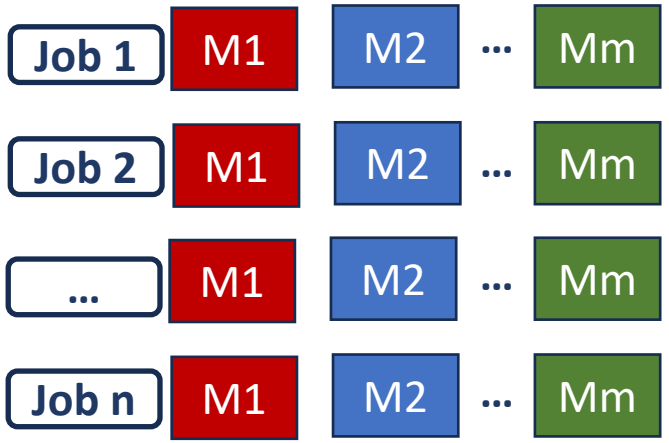


m, Máquinas en paralelo

- ***P***, máquinas idénticas
  - $p_{ij} = p_j \ (j = 1, \dots, m)$
- ***Q***, máquinas uniformes
  - $p_{ij} = \frac{p_i}{v_j}$, hay relación entre el tiempo de procesamiento del trabajo i y la velocidad de la máquina que procesa la tarea j.
- ***R***, máquinas no relacionadas
  - Cada máquina tiene una velocidad diferente de proceso para cada diferente trabajo. (El trabajo 1 puede ser más rápido que el trabajo 2 en la máquina 1, pero más lento en la máquina 2)

Clasificación de problemas de Scheduling

Sistemas multietapa: cada trabajo se procesa en cada máquina al menos una vez, y las máquinas son diferentes



Máquinas en serie

F, Flow Shop Scheduling Problem

- Hay m máquinas que tienen que procesar n trabajos.
- Todos los trabajos tienen el mismo orden de procesamiento en las máquinas.
- Cada máquina puede procesar los trabajos en un orden diferente.
  - La máquina 1 puede procesar antes el trabajo 2 que el 1, la máquina 2 puede ejecutar antes el trabajo 1 que el 2.
- $(n!)^m$ posibles soluciones

Permutation Flow Shop Scheduling Problem

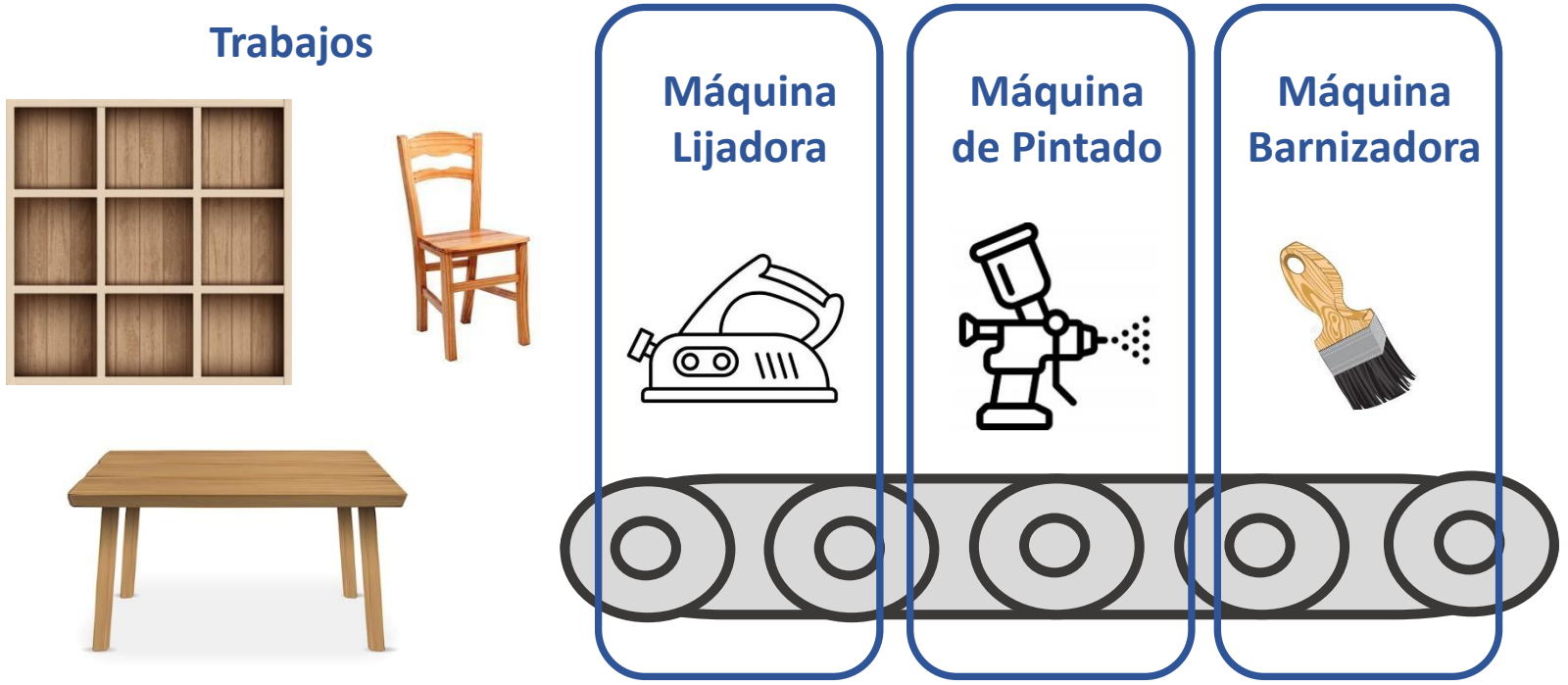
- Cada máquina procesa los trabajos en el mismo orden.
  - Todas las máquinas procesan primero el trabajo 1, luego el 2, etc…
- $(n!)$ posibles soluciones

Clasificación de problemas de Scheduling

Sistemas multietapa: cada trabajo se procesar en cada máquina al menos una vez, y las máquinas diferentes

$$(n!) = (3!) = 6$$

Permutation Flow Shop Scheduling Problem



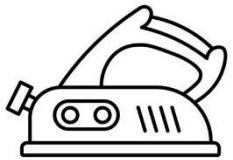
Clasificación de problemas de Scheduling

Sistemas multietapa: cada trabajo se procesar en cada máquina al menos una vez, y las máquinas son diferentes

Flow Shop Scheduling Problem

Trabajos

Máquina Lijadora



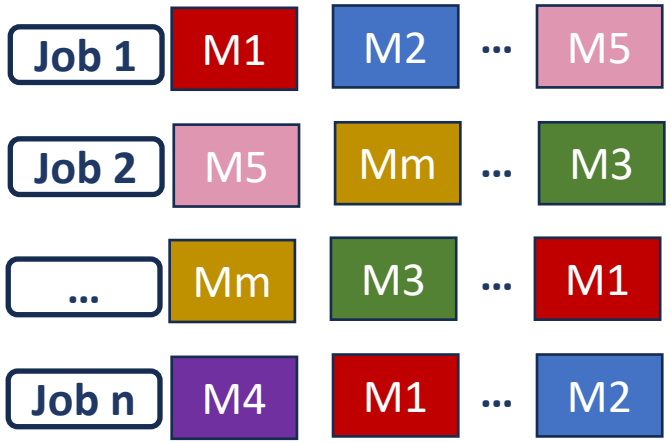
Máquina de Pintado

Máquina Barnizadora

$$(n!)^m = (3!)^3 = (6)^3 = 216$$

Clasificación de problemas de Scheduling

Sistemas multietapa: cada trabajo se procesar en cada máquina al menos una vez, y las máquinas son diferentes



J, Job Shop Scheduling Problem

- Hay m máquinas y n trabajos.
- Cada trabajo tiene predeterminada su ruta a seguir por las máquinas.
- Se distingue entre Job Shops donde cada trabajo visitar cada máquina al menos una vez y aquellos donde un trabajo puede visitar una máquina más de una vez.
- $\frac{(n*m)!}{(m!)^n}$

O, Open Shop Scheduling Problem

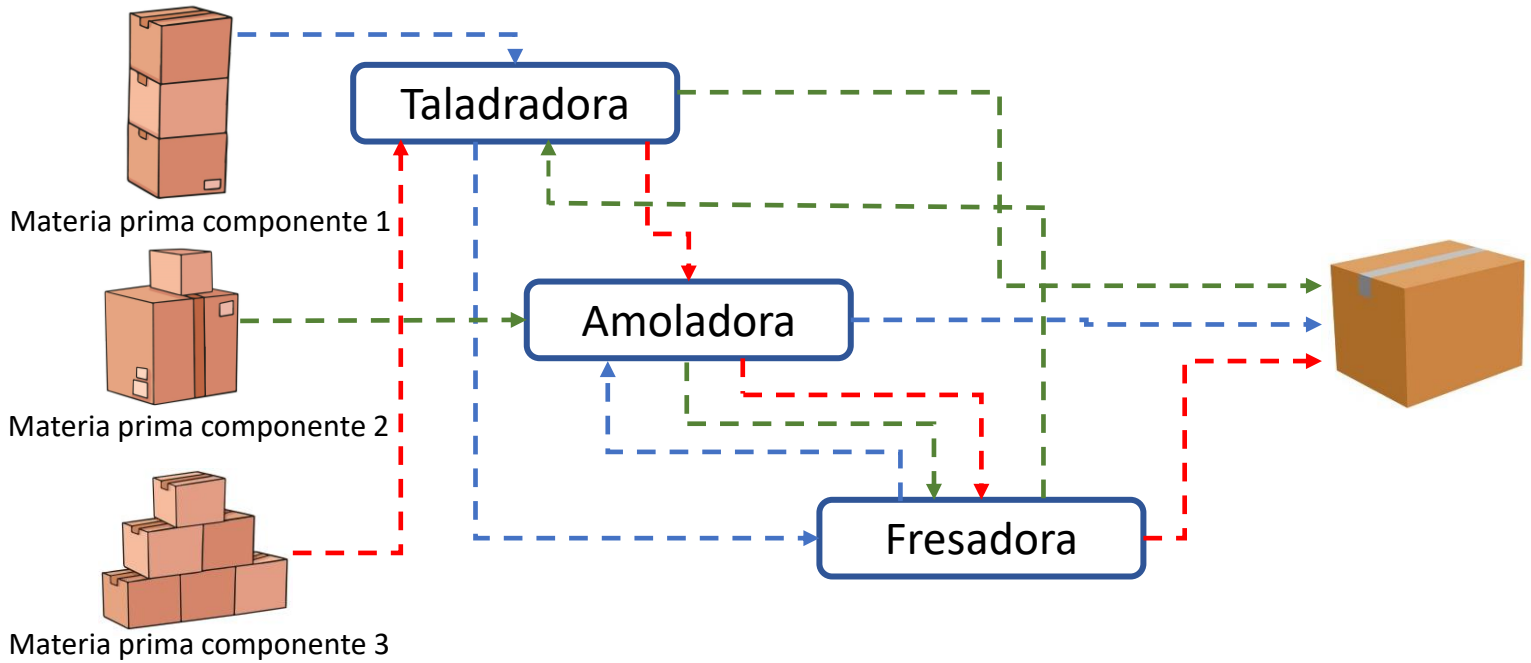
- Hay m máquinas y n trabajos.
- Cada trabajo debe ser procesado en cada una de las máquinas, sin embargo, alguno de los tiempos de procesamiento puede ser cero.
- Los trabajos no tienen rutas predeterminadas a seguir por las máquinas. Se ejecutan en las máquinas en un orden arbitrario.
- $\frac{(n*m)!}{(m!)^n(n!)^m}$

Clasificación de problemas de Scheduling

Sistemas multietapa: cada trabajo se procesar en cada máquina al menos una vez, y las máquinas son diferentes

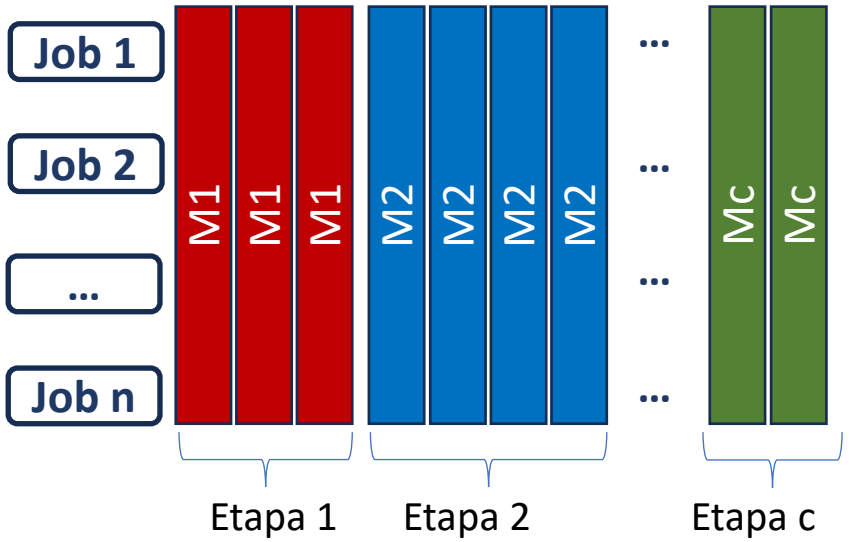
Componente 1	Taladradora	Fresadora	Amoladora
Componente 2	Amoladora	Fresadora	Taladradora
Componente 3	Taladradora	Amoladora	Fresadora

Job Shop Scheduling Problem (máquinas en serie)



Clasificación de problemas de Scheduling

Sistemas multietapa: cada trabajo se procesar en cada máquina al menos una vez, y las máquinas son diferentes



Otras variantes

- **X, Mixed Shop**
  - Unos trabajos tienen establecido un orden de ejecución en las máquinas (como en el Flow o el Job Shop), pero otros son ejecutados en un orden arbitrario (como en el Open Shop).
- **Máquinas en Serie y Paralelo. c** etapas en serie, cada una con una determinada cantidad de máquinas en paralelo.
  - Cada etapa c funciona como un banco de máquinas paralelas. En cada etapa un trabajo necesita sólo una máquina y cualquier máquina puede procesar cualquier trabajo. Colas entre etapas (FIFO)
  - **FFc, Flexible Flow Shop**
    - Generalización del Flow Shop (F) y del de Máquinas en Paralelo (P).
    - Todos los trabajos pasan en el mismo orden por todas las etapas, es decir todos los trabajos se procesan primero en la etapa 1, luego en la etapa 2, y así sucesivamente.
  - **FJc, Flexible Job shop**
    - Generalización del Job Shop (J) y del de Máquinas en Paralelo (P).
    - Cada trabajo pasa en un orden concreto por las etapas, es decir cada trabajo tiene su orden de procesamiento específico en las etapas.

Notación de Graham $\alpha|\beta|\gamma$

- Las diferentes características de los trabajos, las máquinas y las planificaciones se pueden reflejar a través de la clasificación $\alpha|\beta|\gamma$.
- Dada la multitud de variaciones que hay en los problemas de Scheduling, Graham propuso en [Graham 1979] la notación $\alpha|\beta|\gamma$. Notación estándar, para diferenciar y referirse a las diversos modelos de los problemas de Scheduling.
  - α : especifica el entorno de la máquina (una o dos entradas)
  - β : especifica las características de los trabajos (ninguna o múltiples entradas)
  - γ : especifica el criterio de optimalidad o función objetivo (una entrada)

$$\alpha_1\alpha_2|\beta_1;\beta_2;\ldots|\gamma$$

[Graham 1979] *Graham, R. L.; Lawler, E. L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. (1979).* Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the* Discrete *Optimization Symposium. Elsevier. pp. (5) 287–326.*

Calculadora Notación $\alpha|\beta|\gamma$: The Scheduling Zoo. http://schedulingzoo.lip6.fr/

Notación de Graham $\alpha|\beta|\gamma$

Entorno de máquina $\alpha = \alpha_1\alpha_2$

- Sistemas de 1 etapa
  - 1 única máquina dedicada $(\alpha_1 = \circ, \alpha_2 = 1)$
  - m, máquinas en paralelo
    - $\boldsymbol{P}$, máquinas idénticas $(\alpha_1 = P,\ \alpha_2 = m|\circ)$
    - $\boldsymbol{Q}$, máquinas uniformes $(\alpha_1 = Q,\ \alpha_2 = m|\circ)$
    - $\boldsymbol{R}$, máquinas no relacionadas $(\alpha_1 = R,\ \alpha_2 = m|\circ)$
- Sistemas multietapa
  - $\boldsymbol{F}$, Flow Shop $(\alpha_1 = F,\ \alpha_2 = m|\circ)$
  - $\boldsymbol{O}$, Open Shop $(\alpha_1 = O,\ \alpha_2 = m|\circ)$
  - $\boldsymbol{J}$, Job Shop $(\alpha_1 = J,\ \alpha_2 = m|\circ)$
  - **Otras variantes.**
    - $\boldsymbol{X}$, Mixed Shop $(\alpha_1 = X,\ \alpha_2 = m|\circ)$.
    - $\boldsymbol{FF_c}$, Flexible Flow Shop con c etapas $(\alpha_1 = FF_c\ \alpha_2 = m|\circ)$.
    - $\boldsymbol{FJ_c}$, Flexible Job Shop con c etapas $(\alpha_1 = FJ_c\alpha_2 = m|\circ)$.

Entorno trabajos $\beta_1; \beta_2; \ldots$ *múltiples entradas*

- r_i, presencia de "reléase dates".
- d_i, presencia de "due dates", $d_i = d$.
- $p_i = p$, tiempos de procesamiento iguales.
- s_{ik}, s_{ijk} tiempos de setup dependientes de la secuencia.
- $prmp$ ó $pmtn$, se permite la interrupción.
- *Restricciones de precedencia* entre los trabajos:
  - $prec, chains, intree, outtree, tree, sp-graph, \ldots$
- $brkdwn$, caídas y averías de las máquinas. Las máquinas no están continuamente disponibles. Por ejemplo, $m(t)$ máquinas idénticas paralelas están disponibles en el instante t.
- M_i, restricciones de elegibilidad de máquinas. M_i denota el conjunto de máquinas paralelas que pueden procesar el trabajo i.
- $prmu$, permutación Flow Shop.
- $block$, presencia de bloqueo en Flow Shop (cola limitado). Un trabajo terminado no puede pasar de una máquina a la siguiente debido al espacio limitado en la cola. Por lo tanto, bloquea a la máquina.
- nwt, no espera en Flow Shop. No se permite que un trabajo espere entre dos ejecuciones sucesivas en máquinas diferentes.
- $recrc$, recirculación en Job Shop. Cuando un trabajo visita una máquina más de una vez.
- $s-batch, p-batch$, cuando hay un grupo de trabajos (batch) que pueden ser procesados (en serio o paralelo) al mismo tiempo en una única máquina. La presencia de estos grupos permite reducir los tiempos de configuración ("setup") de la máquina.
- …

Notación de Graham $\alpha|\beta|\gamma$

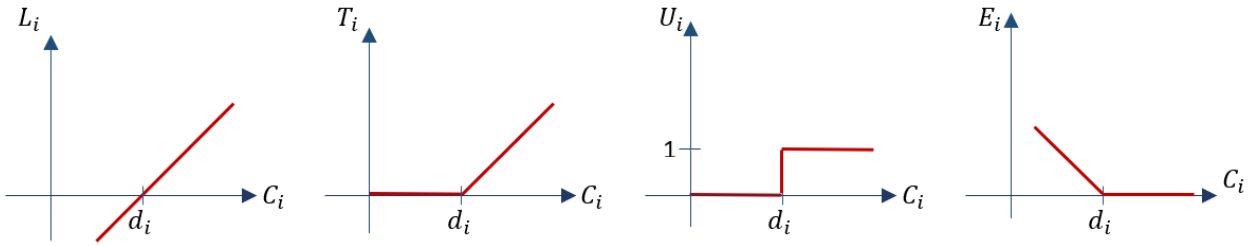
Función Objetivo γ

- Objetivo construir un Schedule factible que optimice una función objetivo.

- Frecuentemente depende de los **tiempos de completud** de los trabajos: C_i.
  - Máximo tiempo de completud (**Makespan**): $C_{max} = max\{C_i | \ i = 1, \dots, n\}$ $(\gamma = C_{max})$
  - Tiempo de flujo Total (**Total Flow Time**): $\sum C_i = \sum_{i=1}^{n} C_i$ $(\gamma = \sum C_i)$
  - Tiempo de flujo total ponderado (**Total Weighted Flow Time**): $\sum w_i C_i = \sum_{i=1}^{n} w_i C_i$ $(\gamma = \sum w_i C_i)$

- Otras dependen de los **tiempos de vencimiento** de los trabajos: d_i. Conceptos relacionados:
  - Máximo retraso (**Maximum Lateness**): $L_{max} = max\{L_i | \ i = 1, \dots, n\}$ $(\gamma = L_{max})$
  - Tardanza Total (**Total Tardiness**) : $\sum T_i = \sum_{i=1}^{n} T_i$ $(\gamma = \sum T_i)$
  - Tardanza Total Ponderada (**Total weighted Tardiness**) : $\sum w_i T_i = \sum_{i=1}^{n} w_i T_i$ $(\gamma = \sum w_i T_i)$
  - Número total de trabajos retrasados (**Total number of late jobs**): $\sum U_i = \sum_{i=1}^{n} U_i$ $(\gamma = \sum U_i)$
  - Número total ponderado de trabajos retrasados (**Total weighted number of late jobs**):

    $\sum w_i U_i = \sum_{i=1}^{n} w_i U_i$ $(\gamma = \sum w_i U_i)$



$L_i = C_i - d_i$, Retraso del trabajo i (**Lateness**). $L_i < 0$ ó $L_i \geq 0$.

$T_i = max(C_i - d_i, 0) = max(L_i, 0)$, Tardanza (**Tardiness**) del trabajo $i, T_i \geq 0$.

$E_i = max(0, \ d_i - C_i)$, Puntualidad (**Earliness**) del trabajo i. $E_i \geq 0$.

$U_i = \begin{cases} 1, & si \ C_i > d_i \\ 0, & en \ otro \ caso \end{cases}$, Penalización unitaria (**Unit Penalty**). Trabajo j retrasado o no.

Algunos ejemplos

- $1|r_i; pmtn|L_{max}$
  - Problema de scheduling en el que se han de planificar una serie de trabajos, con unas fechas de lanzamiento determinadas (relesase dates) y que pueden ser interrumpidos, en una sola máquina, con el objetivo de minimizar el máximo retraso (Lateness).
- $P|p_i = 1|C_{max}$
  - Problema de Scheduling en el que se han de planificar una serie de trabajos con duración unitaria, en m máquinas paralelas idénticas, con el objetivo de minimizar el máximo tiempo de completud (makespan).
- $J3|p_i = p|\sum T_i$

  Problema Job Shop Scheduling en el que se han de planificar una serie de trabajos con la misma duración en 3 máquinas, con el objetivo de minimizar el Total Tardiness.

Practiquemos un poco!!!

Identifica las máquinas, tareas y el objetivo, e indica la notación $\alpha|\beta|\gamma$ para los siguientes problemas de Scheduling

a) En la industria editorial, la publicación de un libro pasa por las siguientes fases: composición tipográfica, impresión, encuadernación y empaquetado. Los tiempos de procesamiento de estas fases son diferentes dependiendo del tamaño del libro, el número de copias, etc. El objetivo es la publicación de los libros lo antes posible.

b) En la industria de la confección, la elaboración de una prenda consta de las siguientes etapas: corte, confección, planchado, empaquetado. El objetivo es producir todas las prendas lo antes posible.

| | | | |

Máquinas	Trabajos	Objetivo

Practiquemos un poco!!!

c) En la empresa siderúrgica, diferentes barras o vigas pasan por el conjunto de rodillos en su propio orden con sus propias temperaturas y ajustes de presión. El objetivo es producir todos los elementos lo antes posible.

Máquinas	Trabajos	Objetivo

d) En un taller de automóviles se realizan las siguientes actuaciones: cambiar neumáticos, reparar la caja de cambios, revisar frenos, reparar faros, etc. El objetivo es realizar la revisión de todos los coches. para todas estas actuaciones, lo antes posible.

Máquinas	Trabajos	Objetivo

Practiquemos un poco!!!

e) Planificación de la preparación de una asignatura. Se ha de preparar la materia de n módulos de una asignatura compuestos por diversos temas (T1, T2, T3,…). Los módulos son independientes por lo que se puede interrumpir el estudio de un tema para abordar el estudio de otro tema de otro módulo. Dentro de un módulo los temas han de ser estudiados en el instante en el que ya hayan sido impartidos en clase. Cada módulo tiene un tiempo de estudio diferente y una fecha tope en la que debería haber sido estudiado. El objetivo es terminar el estudio de los módulos, minimizando el tiempo de tardanza total en el estudio de los temas de todos los módulos.

| | |

Máquinas	Trabajos	Objetivo

Practiquemos un poco!!!

f) La revisión bibliográfica para el Trabajo Fin de Grado requiere la lectura de n libros de la biblioteca. Cada libro requiere un número diferente de días para su lectura y debe ser devuelto antes de la fecha de vencimiento del préstamo de la biblioteca. La biblioteca cobra 50 céntimos por día por cada libro atrasado. El objetivo es minimizar la multa total.

Máquinas	Trabajos	Objetivo

||

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

El Problema Job Shop Scheduling (JSSP)

"El problema general de job shop es un reto fascinante. Si bien es fácil plantearlo y visualizarlo, es extremadamente difícil resolverlo. El problema continúa atrayendo investigadores, quienes no pueden creer que un problema tan simplemente estructurado, pueda ser tan difícil, hasta que lo intentan."

Richard W. Conway, et al. (1967)

El Problema Job Shop Scheduling (JSSP)

- *Es uno de los **problemas más difíciles** que hay.*
- *Está catalogado como **NP-duro**, lo que indica que no todas las instancias del problema pueden ser resueltas, y que las que se resuelven requieren de un algoritmo no determinista.*

Para darnos cuenta de la dificultad del problema JSSP, un problema con 20 trabajos y 10 tareas cada uno (20×10) *tiene* 7.261×10^{183} *soluciones posibles, lo que en principio supera la edad del universo expresada en microsegundos, es más, muchas de estas soluciones pueden no ser factibles al incumplir algunas de las restricciones del problema.*

El Problema Job Shop Scheduling (JSSP)

- Asignación temporal de recursos para la ejecución de las tareas de un conjunto de trabajos, bajo ciertas restricciones, para optimizar uno o más objetivos
  - Recursos: máquinas de un taller, operarios…
  - Tareas: operaciones a realizar en las máquinas…
  - Restricciones: limitaciones de capacidad, no-interrupción…
  - Objetivos: minimizar el tiempo de finalización del proyecto..
- Organizar la ejecución de las tareas en los recursos

El Problema Job Shop Scheduling (JSSP)

EJEMPLO: Fábrica de coches

Pedir	Cortar	Soldar	Pintar	Ensamblar
Hatchback	2 hours	1.5 hours	1 hours	4 hours
Sport car	2 hours	1.5 hours	1 hours	10 hours
Sedan	2.5 hours	1.5 hours	1.5 hours	5 hours
4x4 car	2.5 hours	1.5 hours	2 hours	8 hours

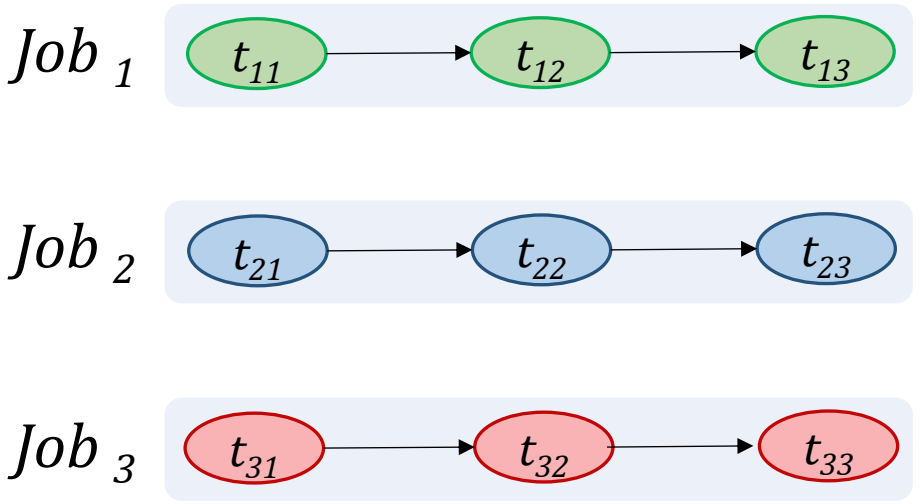
Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - **Definición del JSSP**
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensiones del JSSP
5. Referencias Bibliográficas

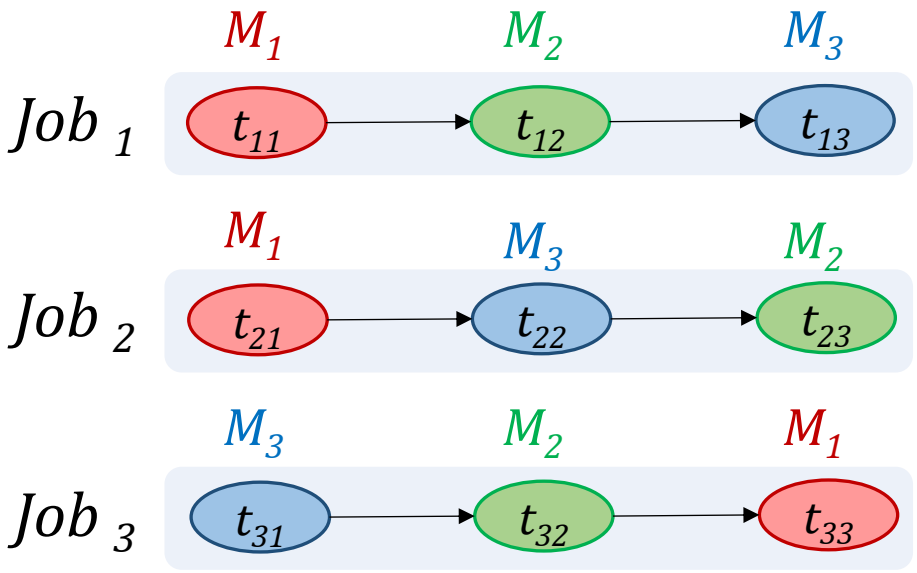
Definición del JSSP

- Datos
  - Conjunto de trabajos, $J = \{J_1, \dots, J_n\}$
  - Cada trabajo consiste en una secuencia de tareas: $J_i = \{t_{i1}, \dots, t_{in_i}\}$



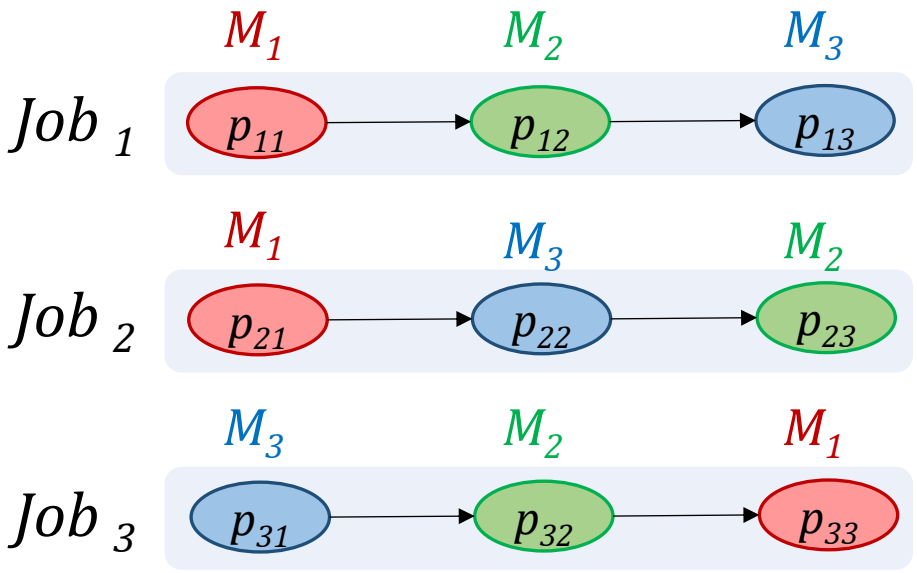
Definición del JSSP

- Datos
  - Conjunto de máquinas, $M = \{M_1, \dots, M_m\}$
  - Cada tarea ha de ejecutarse en una máquina concreta



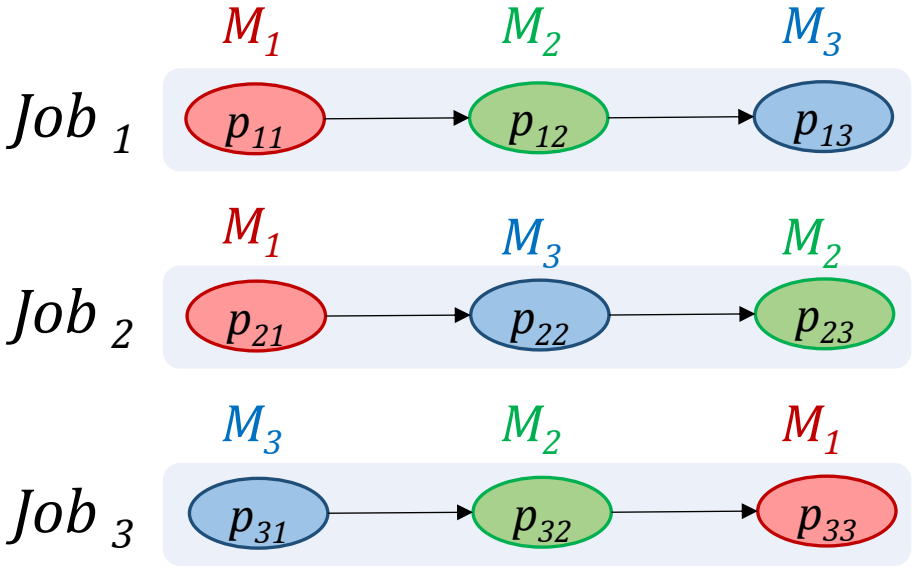
Definición del JSSP

- Datos
  - La ejecución de cada tarea t_{ij} en tiene una duración conocida p_{ij}



Definición del JSSP

- Datos
  - Número de trabajos y de máquinas
  - Secuencia de máquinas que usan las tareas de cada trabajo
  - Tiempos de procesamiento (duración) de las tareas, p_{ij}



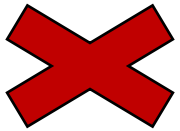
3 trabajos
3 máquinas
Secuencias de máquinas
1 2 3
1 3 2
3 2 1
Duraciones
3 2 4
4 2 3
2 1 3

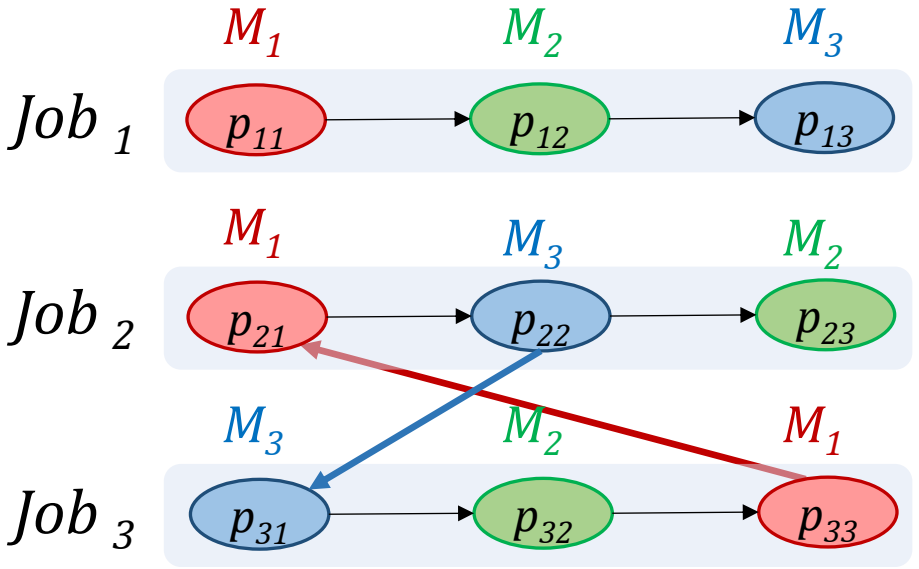
Definición del JSSP

- Restricciones
  - Secuenciales (de trabajo)
    - Una tarea no puede empezar antes de que termine la anterior en su trabajo
  - Capacidad (de máquina):
    - No se pueden solapar la ejecución de dos tareas en una misma máquina
  - No-interrupción:
    - No puede interrumpirse la ejecución de las tareas

Definición del JSSP

- Meta
  - Construir un *schedule*, es decir, el tiempo de inicio, st_{ij}, de todas las tareas o, simplemente, un orden de procesamiento de las tareas
    - Órdenes de procesamiento de las tareas en las máquinas, que sean "compatibles"





Definición formal del JSSP

- Datos
  - Trabajos: $J = \{J_1, \dots, J_n\}$
  - Máquinas: $M = \{M_1, \dots, M_m\}$
  - Tareas de $J_i = (t_{i1}, \dots, t_{in_i})$
  - Duración de $t_{ij} : p_{ij}$
  - Máquina para $t_{ij} : \mu_{ij}$
- Variables de decisión:
  - Tiempo de inicio de t_{ij}: st_{ij}
  - Tiempo de fin de t_{ij}: C_{ij}

- Restricciones:
  - Secuenciales: $st_{ij} + p_{ij} \leq st_{i(j+1)},\ \forall t_{ij}, 1 \leq j < n_i$
  - Capacidad: $st_{ij} + p_{ij} \leq st_{kl} \vee st_{kl} + p_{kl} \leq st_{ij},$
    $$\forall t_{ij}, t_{kl} : \mu_{ij} = \mu_{kl}$$
  - No-interrupción: $C_{ij} = st_{ij} + p_{ij}$

- Funciones objetivo:
  - $(J|\ \ |C_{max})$ Minimizar el makespan $C_{max} = \max C_{ij}$
  - $(J|\ \ |\Sigma C_i)$ Minimizar el flujo total $C_{sum} = \Sigma C_i$
  - $(J|\ \ |\Sigma T_i)$ Minimizar el retraso respecto a fechas de entrega de los trabajos (Tardiness),
  - $(J|\ \ |L_{max})$ Minimizar el máximo tiempo de retraso (Lateness)
  - …

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - **Representación JSSP**
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

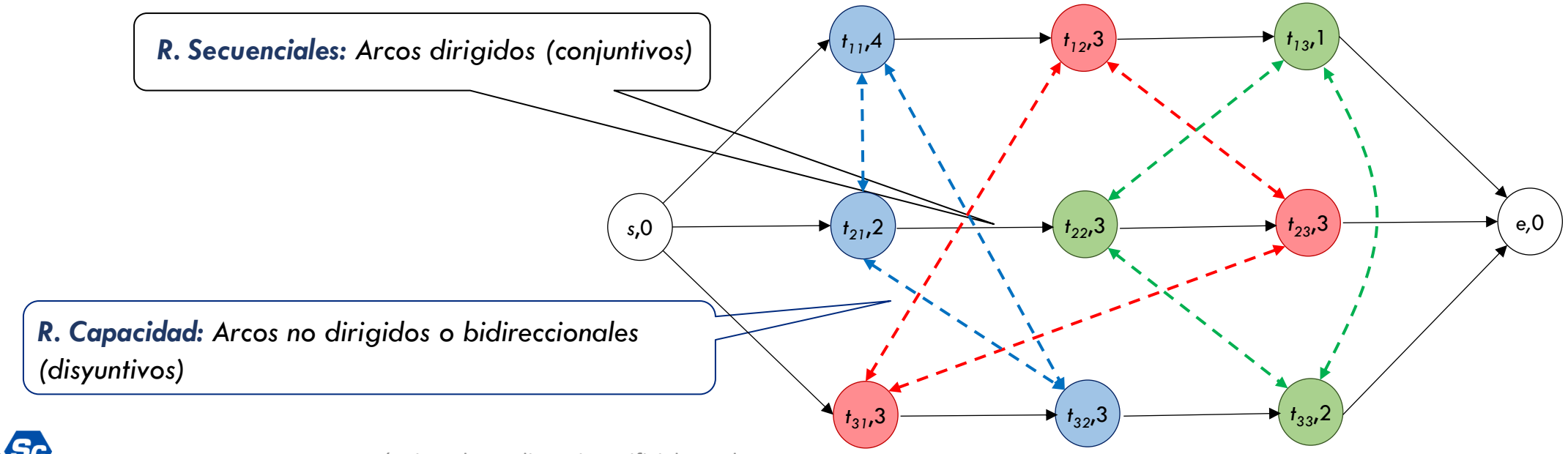
Representación. Modelo Grafo Disyuntivo

- Mediante un **Grafo Dirigido** $G=(V,A \cup E)$, dónde:
  - V: Conjunto de nodos finito que representan a las tareas del problema, con la excepción de los nodos ficticios inicial (s) y final (e).
    - El nodo ficticio inicial está conectado con la primera tarea de cada trabajo, y las últimas tareas de cada trabajo están conectadas con el nodo ficticio final. El tiempo de procesamiento de los nodos ficticios es 0.
  - A: Conjunto de arcos conjuntivos finito. Reflejan las relaciones de precedencia entre las tareas de un mismo trabajo, restricciones secuenciales.
  - E: Conjunto de arcos disyuntivos finito. Reflejan las relaciones de precedencia entre las tareas que emplean el mismo recurso, es decir, las restricciones de capacidad.
    - Se descompone en m subconjuntos E_i, uno por cada máquina, de forma que $E = \cup_{i=1..m} E_i$. El subconjunto E_i incluye un arco (v, w) por cada par de tareas v y w que requieren la máquina M_i.

Representación. Modelo Grafo Disyuntivo

Grafo de restricciones

- Representa el problema JSS
  - Vértices: tareas con sus costes. Dos vértices adicionales $\{s, e\}$
  - Arcos: restricciones

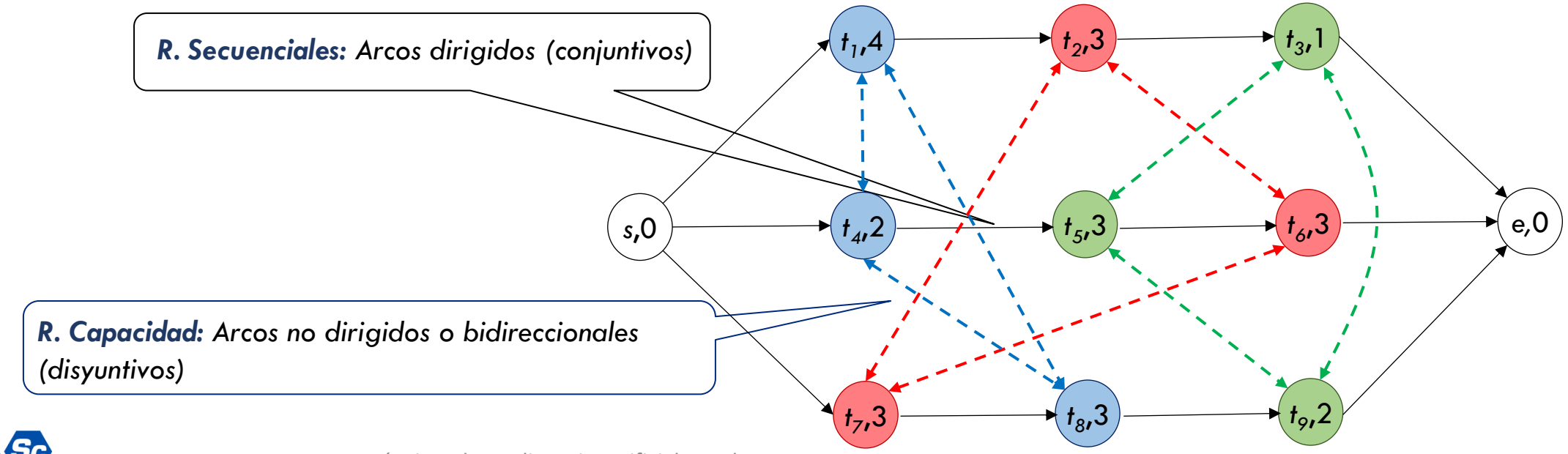


Representación. Modelo Grafo Disyuntivo

Grafo de restricciones

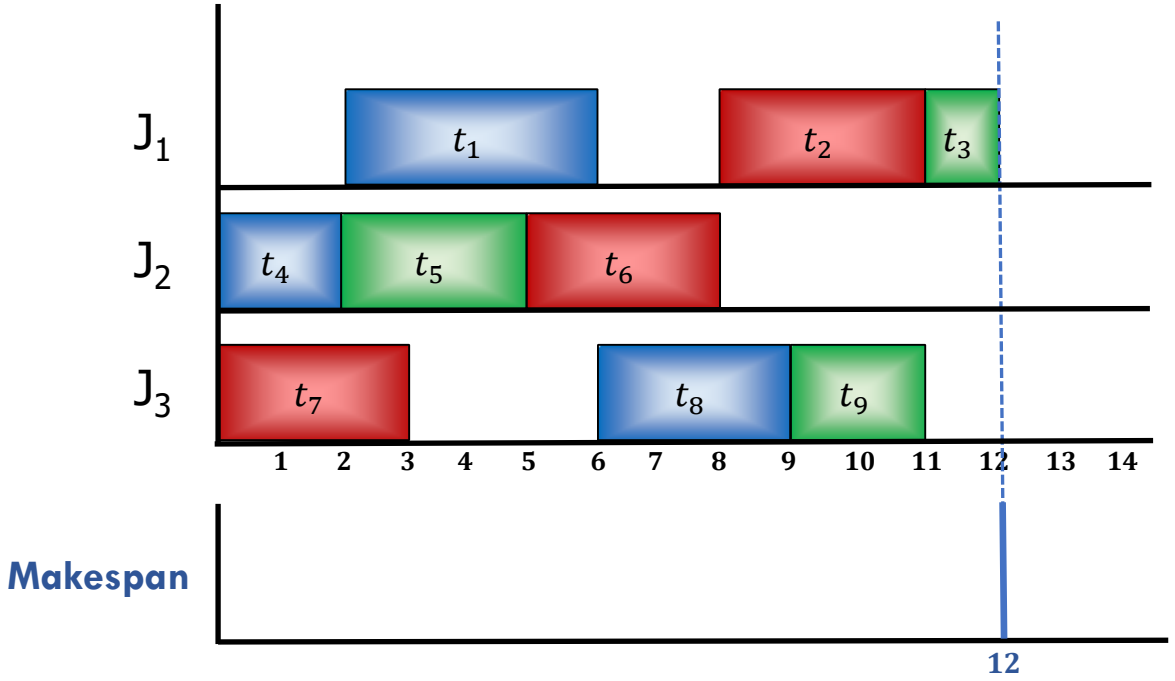
- Representa el problema JSS
  - Vértices: tareas con sus costes. Dos vértices adicionales $\{s, e\}$
  - Arcos: restricciones

También se pueden enumerar los nodos de las tareas con un solo subíndice.



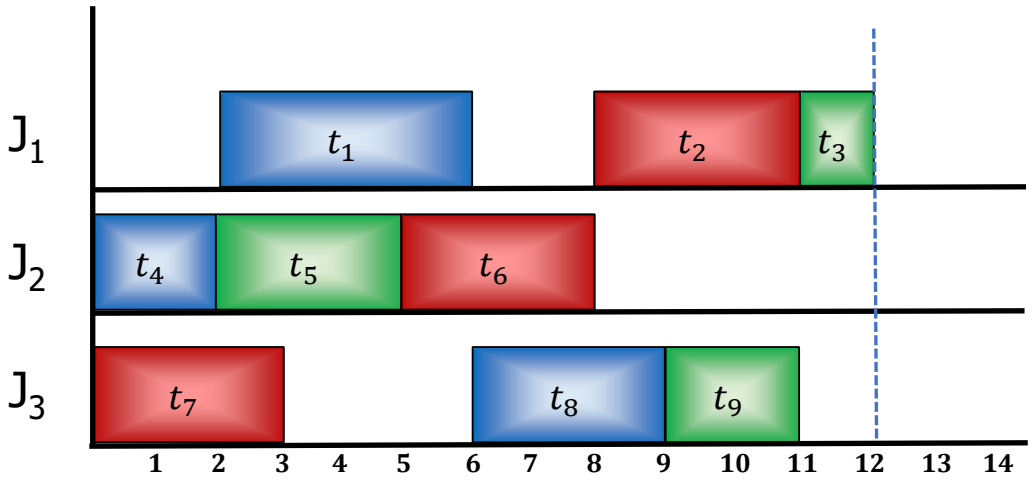
Representación. Gráfico de Gannt

- Representa un Schedule mediante los tiempos de inicio y las duraciones de las tareas

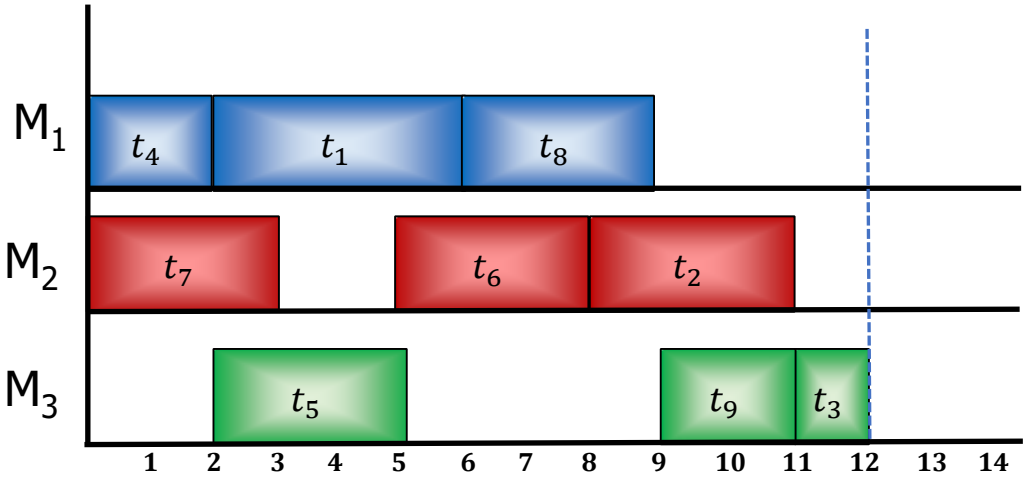


Representación. Gráfico de Gannt

Orientado al trabajo

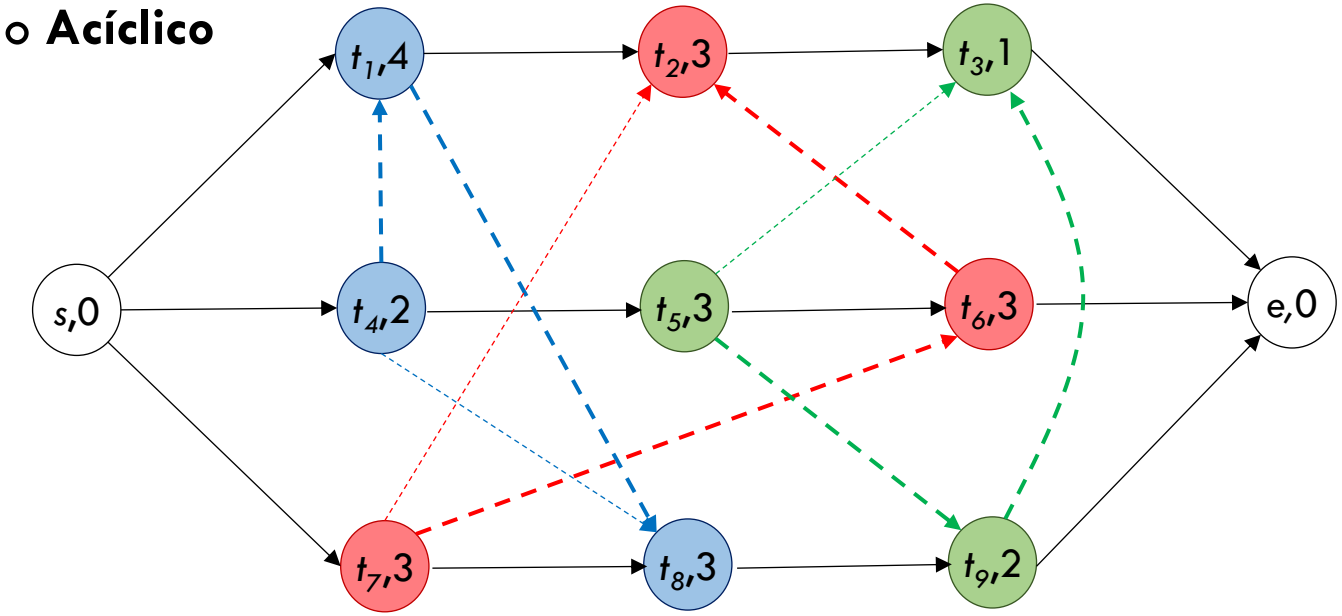


Orientado a la máquina



Representación. Grafo Solución

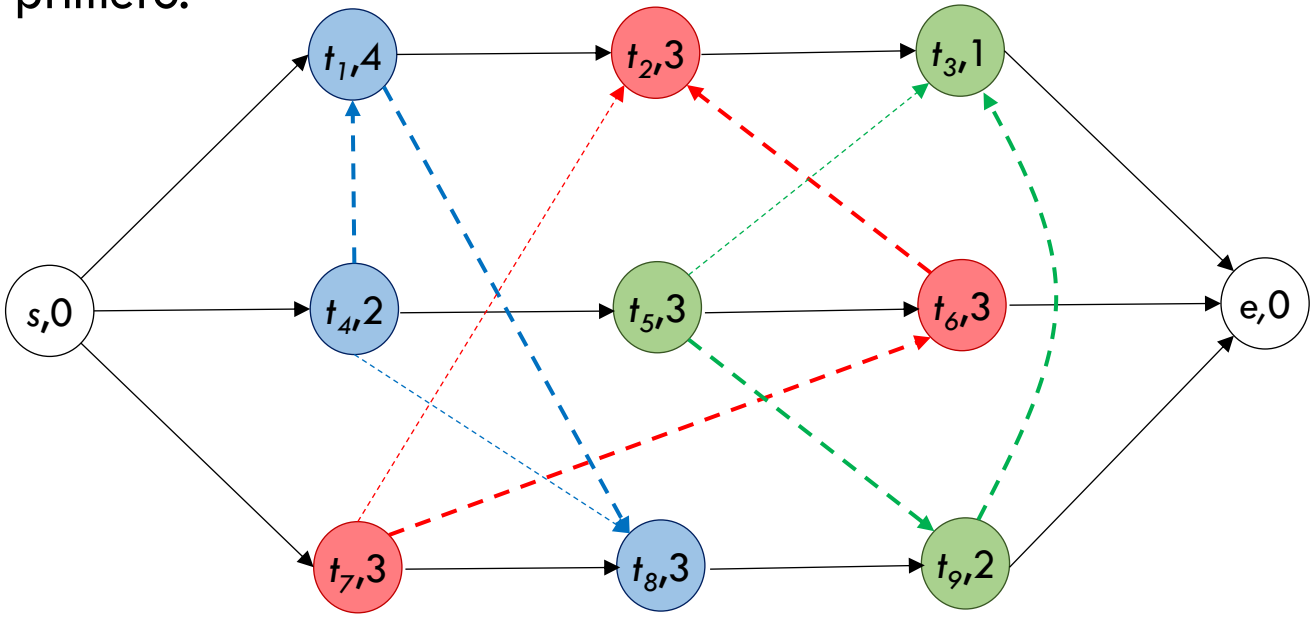
- Resolver el problema = elegir un sentido para los arcos disyuntivos
  - Solución factible = Grafo Dirigido **Acíclico**



Orden de ejecución de las tareas = ordenación topológica del grafo

Ejemplo calculo orden topológico

- Calcular un orden topológico para el siguiente grafo solución
  - Nodos con tareas de menor índice primero.



Orden:∅
Nodos sin arcos de entrada: s

Ejemplo calculo orden topológico

- Calcular un orden topológico para el siguiente grafo solución
  - Nodos con tareas de menor índice primero.

¡No único!

$(s, t_4, t_1, t_5, t_7, t_6, t_2, t_8, t_9, t_3, e)$

$(s, t_4, t_7, t_1, t_8, t_5, t_6, t_2, t_9, t_3, e)$

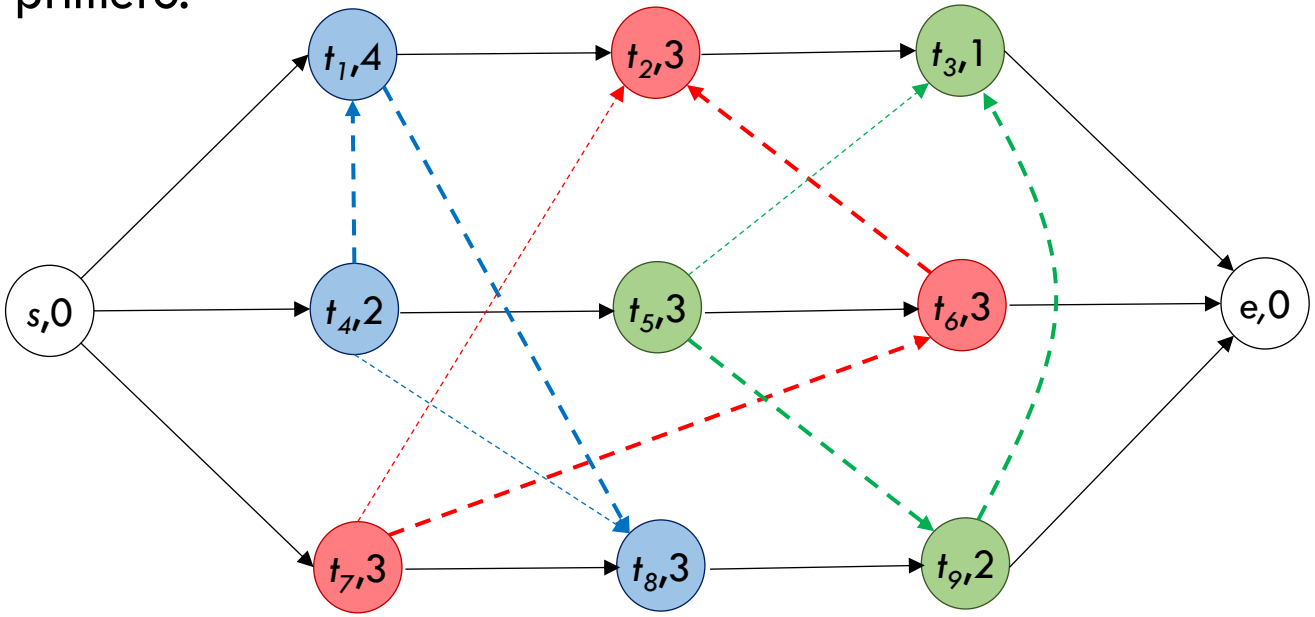
$(s, t_4, t_7, t_1, t_5, t_8, t_6, t_2, t_9, t_3, e)$

$(s, t_7, t_4, t_1, t_8, t_5, t_6, t_2, t_9, t_3, e)$

$(s, t_7, t_4, t_1, t_5, t_8, t_6, t_2, t_9, t_3, e)$

…

Orden: $s, t_4, t_1, t_5, t_7, t_6, t_2, t_8, t_9, t_3, e$
Nodos sin arcos de entrada: ∅

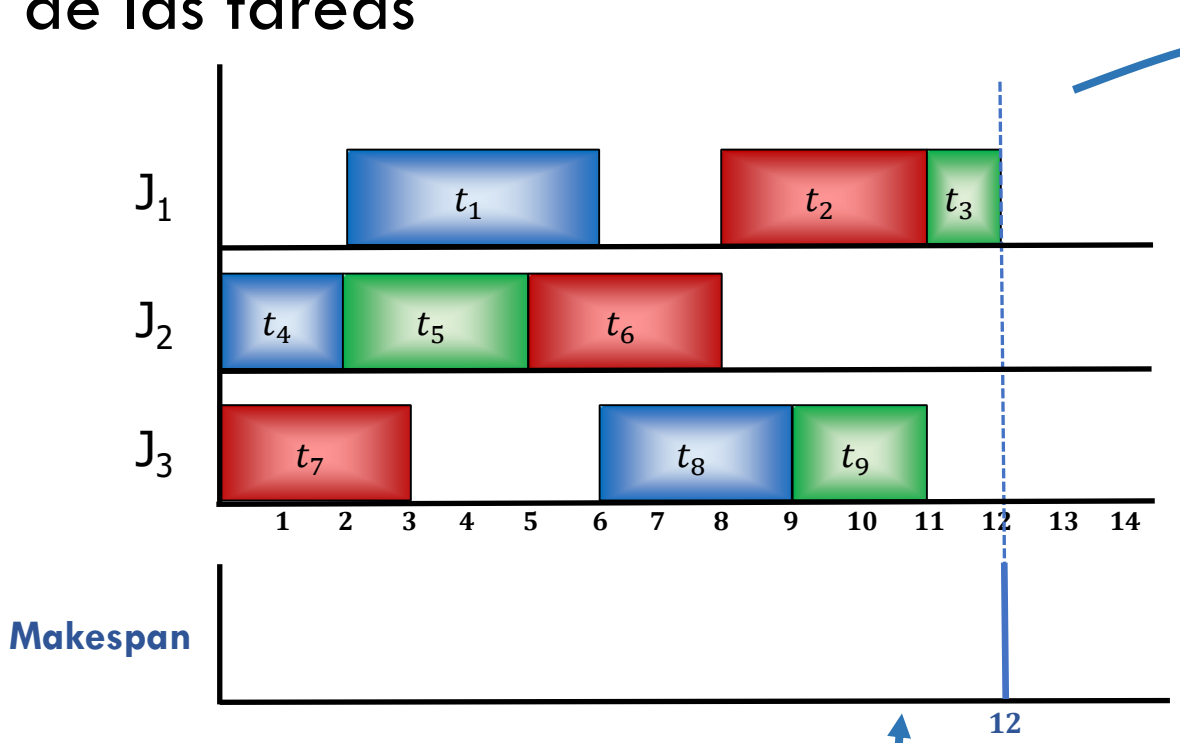


Orden de ejecución de las tareas = ordenación topológica del grafo

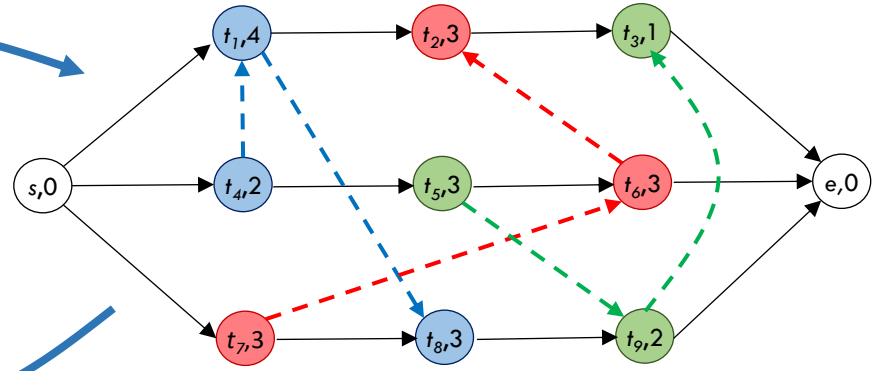
$(s, t_4, t_1, t_5, t_7, t_6, t_2, t_8, t_9, t_3, e)$

Gráfico Gantt - Grafo Solución - Orden tareas

- Representa un schedule mediante los tiempos de inicio y las duraciones de las tareas



¡Único!



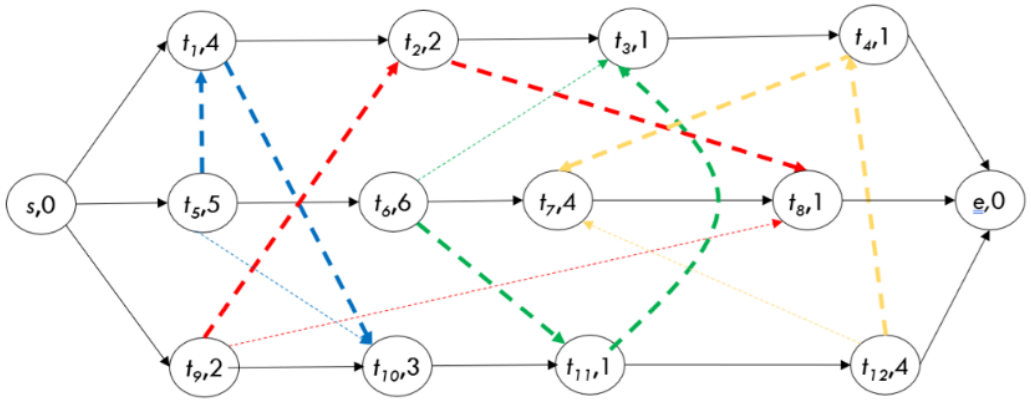
¡Único!

Orden de ejecución de las tareas

$(4,7,1,5,6,8,2,9,3)$

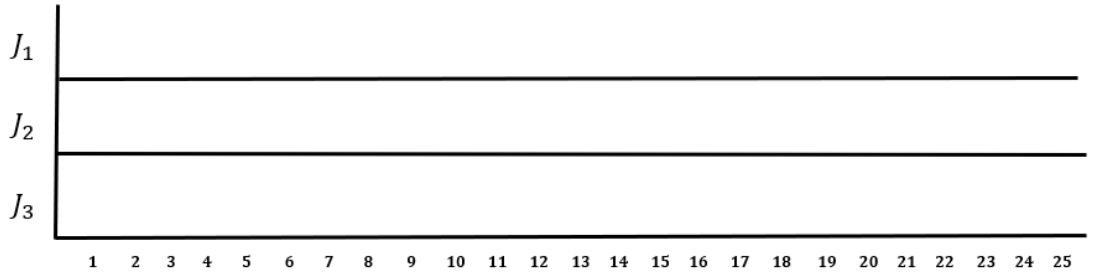
Practiquemos un poco!!!

- Dado el grafo disyuntivo que representa una solución para el problema Job Shop Scheduling cuyos datos se muestran en la tabla de la derecha.



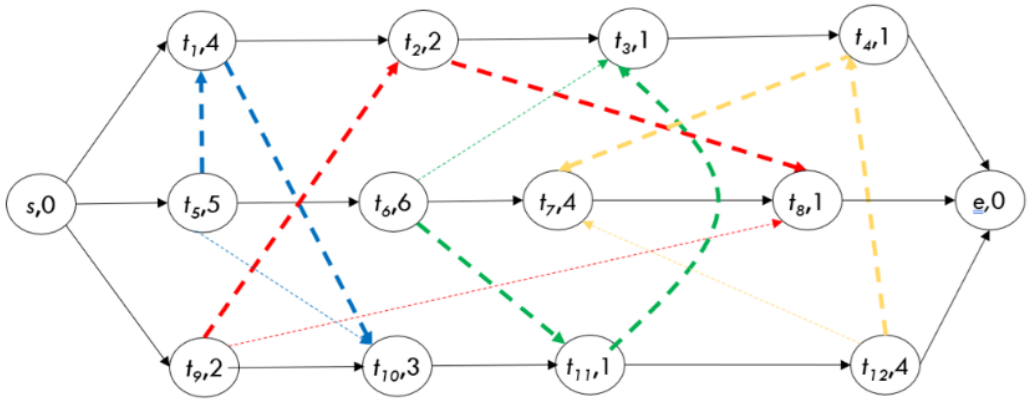
	$J_1, d_1 = 17$				$J_2, d_2 = 20$				$J_3, d_3 = 15$			
Tarea i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	4	2	1	1	5	6	4	1	2	3	1	4
m_i	0	1	2	3	0	2	3	1	1	0	2	3

a) Dibuja un diagrama de Gannt compatible con la solución representada por el grafo disyuntivo.



Practiquemos un poco!!!

- Dado el grafo disyuntivo que representa una solución para el problema Job Shop Scheduling cuyos datos se muestran en la tabla de la derecha.



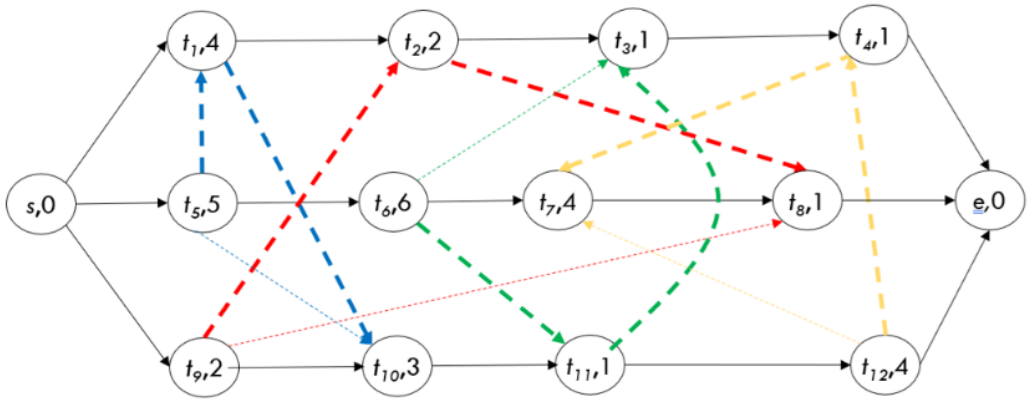
	$J_1, d_1 = 17$				$J_2, d_2 = 20$				$J_3, d_3 = 15$			
Tarea i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	4	2	1	1	5	6	4	1	2	3	1	4
m_i	0	1	2	3	0	2	3	1	1	0	2	3

b) Para la solución representada en el grafo disyuntivo, indica los costes de las siguientes funciones objetivo:

- Makespan:
- Total Flow Time:
- Total Tardiness:

Practiquemos un poco!!!

- Dado el grafo disyuntivo que representa una solución para el problema Job Shop Scheduling cuyos datos se muestran en la tabla de la derecha.



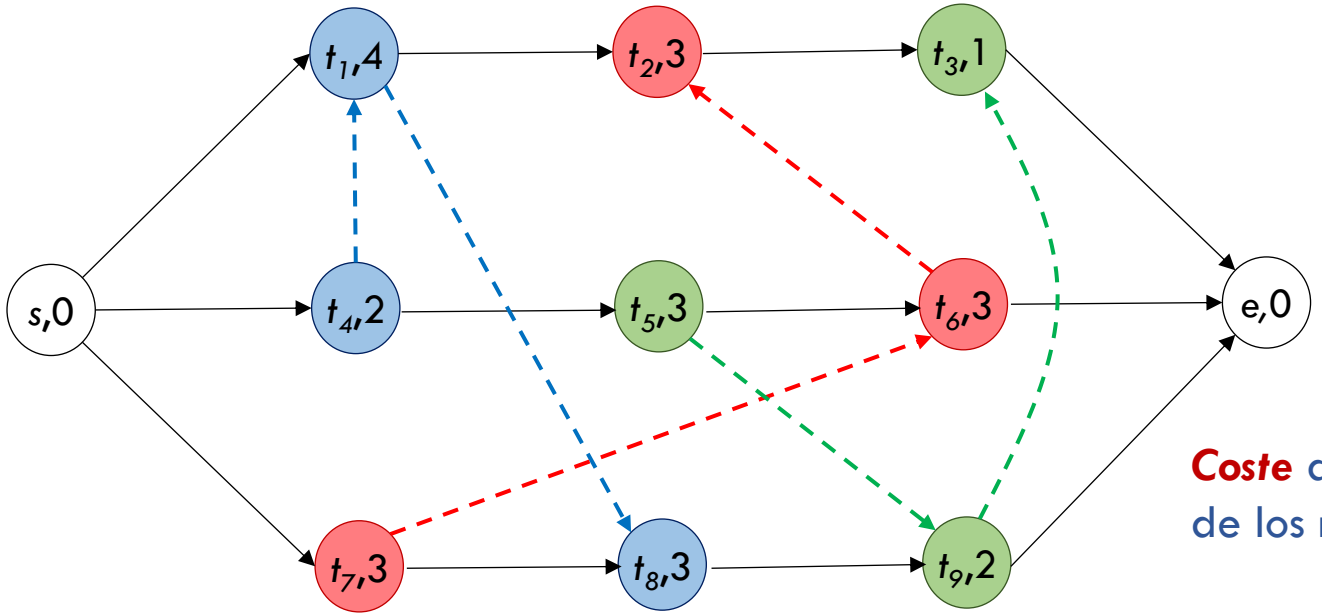
	$J_1, d_1 = 17$				$J_2, d_2 = 20$				$J_3, d_3 = 15$			
Tarea i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	4	2	1	1	5	6	4	1	2	3	1	4
m_i	0	1	2	3	0	2	3	1	1	0	2	3

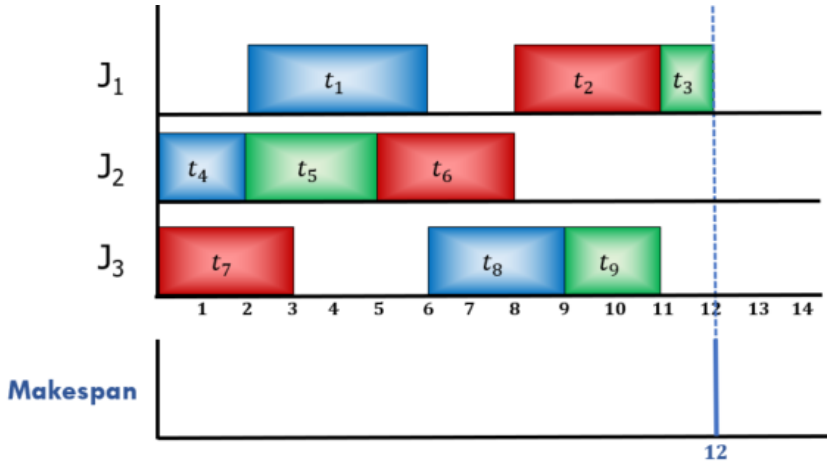
c) Indica si el siguiente orden topológico es un orden topológico acorde con la solución del grafo. En el caso de que no lo sea, justifica por qué no lo es. Indica otro orden topológico para el grafo solución.

$(s, t_5, t_1, t_6, t_{11}, t_9, t_2, t_3, t_{10}, t_{12}, t_4, t_7, t_8, e)$

Grafo Solución. Cabezas y Colas

- **Cabeza de t_i**: coste máximo de s a t_i. Es el tiempo de inicio más temprano posible para t_i
- **Cola de t_i** : coste máximo de t_i a e. Es el menor tiempo de procesamiento restante del proyecto tras la ejecución de t_i

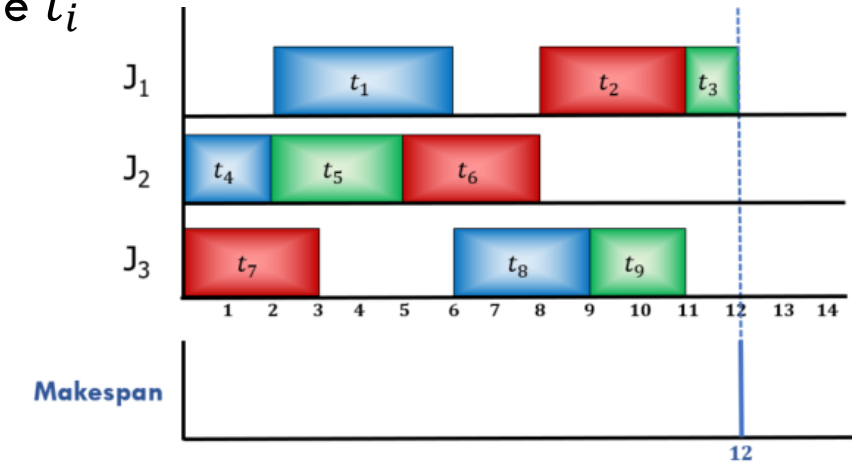


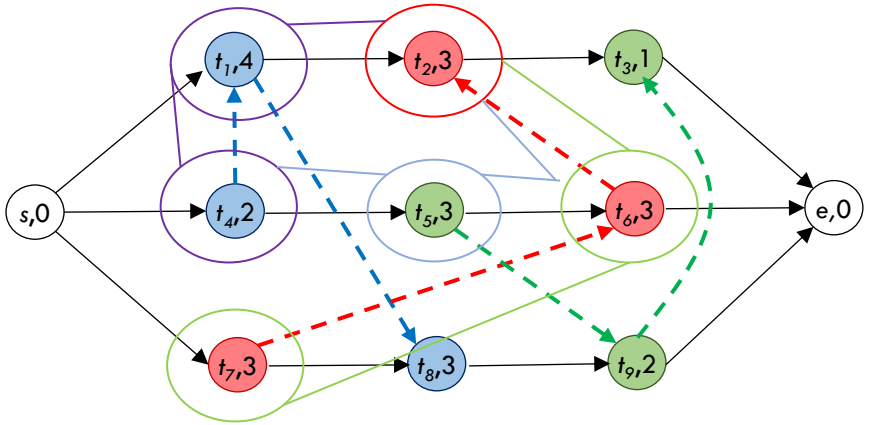


Coste de un camino de s a t_i = suma de las duraciones de los nodos del camino

Grafo Solución. Cabezas y Colas

- **Cabeza de t_i:** coste máximo de s a t_i. Es el tiempo de inicio más temprano posible para t_i
- **Cola de t_i :** coste máximo de t_i a e. Es el menor tiempo de procesamiento restante del proyecto tras la ejecución de t_i





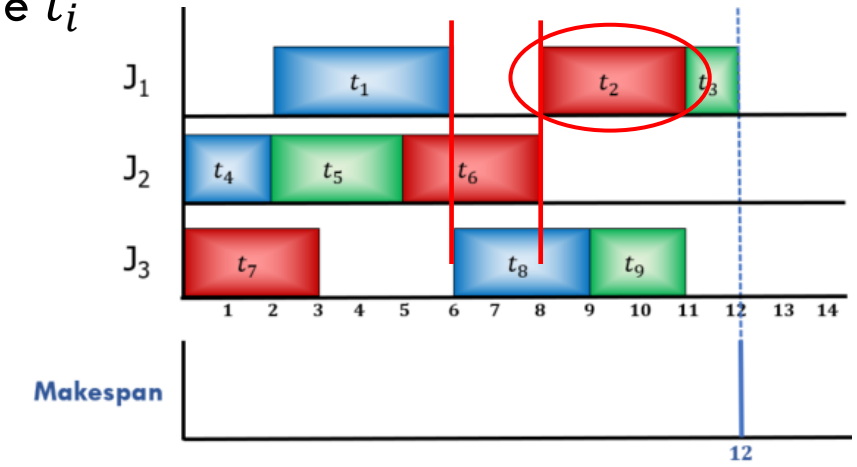
Coste de un camino de s a t_i. = suma de las duraciones de los nodos del camino

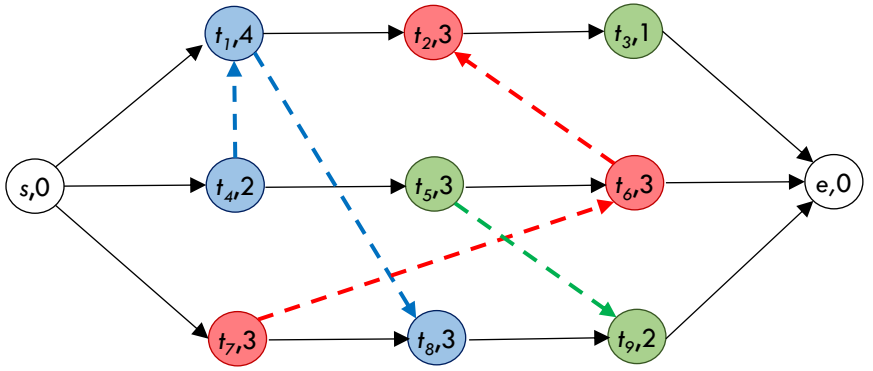
Cabeza de $t_2 = \max\{2+4, 3+3, 2+3+3\} = 8$

Calculadas en "orden" (topológico) $r_{t_2} = \max\{r_{t_1} + 4, r_{t_6} + 3\} = \max\{2+4, 5+3\} = 8$

Grafo Solución. Cabezas y Colas

- **Cabeza de t_i**: coste máximo de s a t_i. Es el tiempo de inicio más temprano posible para t_i
- **Cola de t_i** : coste máximo de t_i a e. Es el menor tiempo de procesamiento restante del proyecto tras la ejecución de t_i



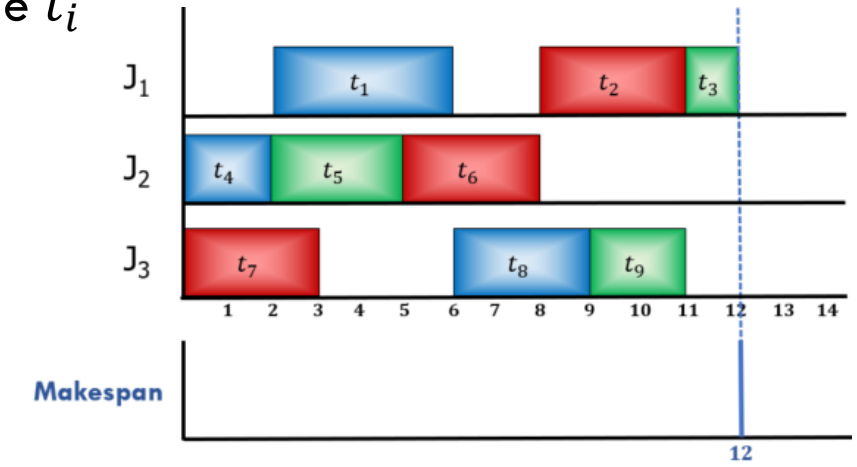


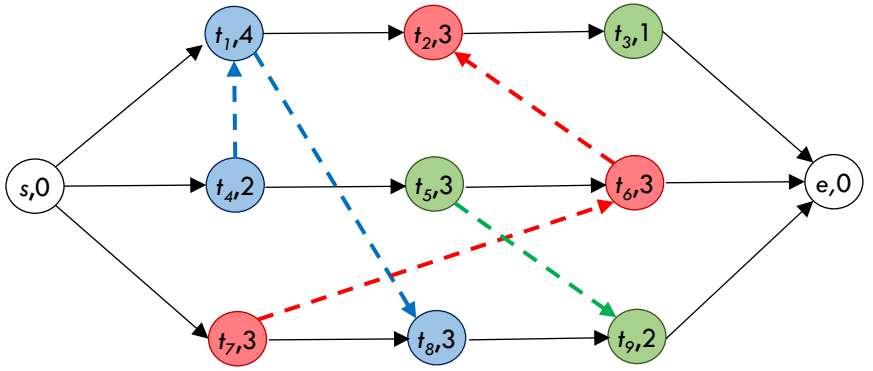
En el gráfico de Gantt = Máximo entre el tiempo de fin de la anterior en su trabajo y la anterior en su máquina

Cabeza de t_2, $r_{t_8} = \max\{6, 8\} = 8$

Grafo Solución. Cabezas y Colas

- **Cabeza de t_i**: coste máximo de s a t_i. Es el tiempo de inicio más temprano posible para t_i
- **Cola de t_i** : coste máximo de t_i a e. Es el menor tiempo de procesamiento restante del proyecto tras la ejecución de t_i

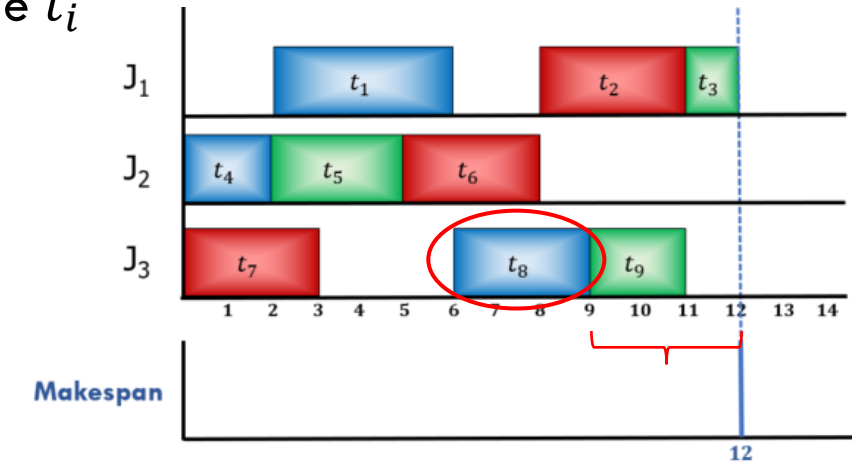


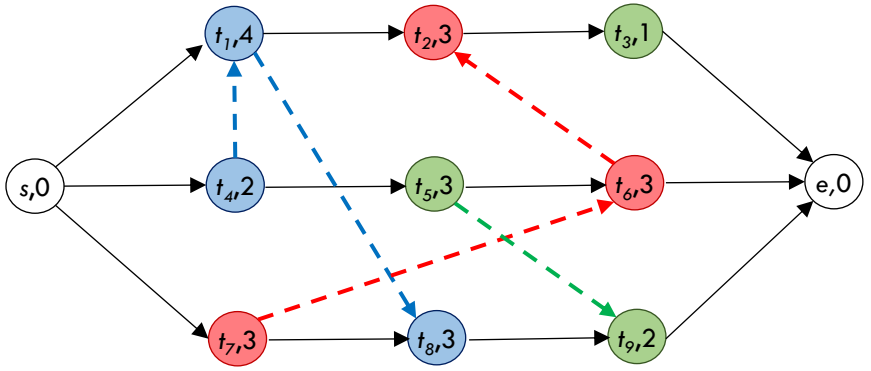


	s	t_4	t_7	t_5	t_1	t_6	t_8	t_2	t_9	t_3	e
Cabezas	**0**	0	0	2	2	5	6	8	9	11	12

Grafo Solución. Cabezas y Colas

- **Cabeza de t_i:** coste máximo de s a t_i. Es el tiempo de inicio más temprano posible para t_i
- **Cola de t_i :** coste máximo de t_i a e. Es el menor tiempo de procesamiento restante del proyecto tras la ejecución de t_i

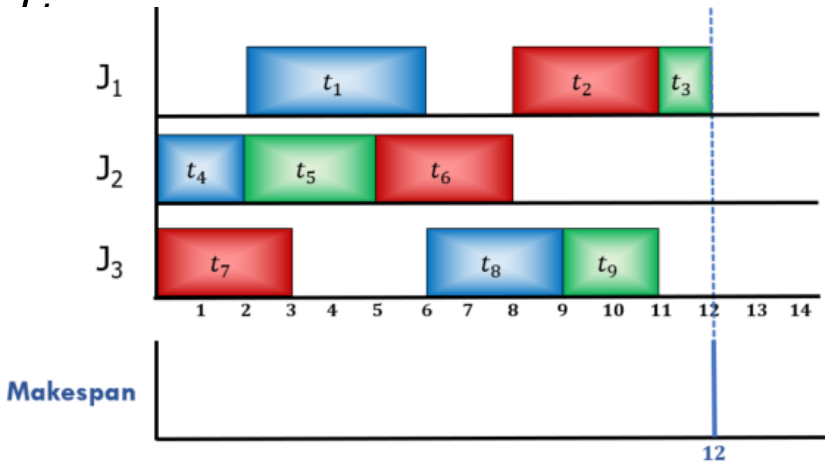


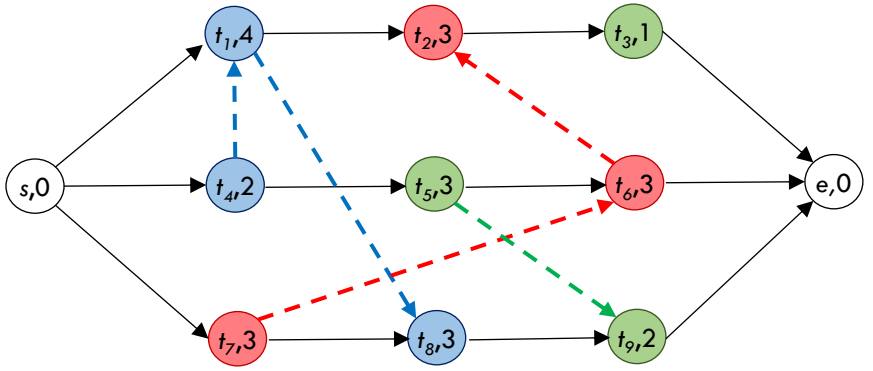


Cola de t_8 , $\mathbf{q}_{t_8} = 12 - 9 = 3$

Grafo Solución. Cabezas y Colas

- **Cabeza de t_i:** coste máximo de s a t_i. Es el tiempo de inicio más temprano posible para t_i
- **Cola de t_i :** coste máximo de t_i a e. Es el menor tiempo de procesamiento restante del proyecto tras la ejecución de t_i.

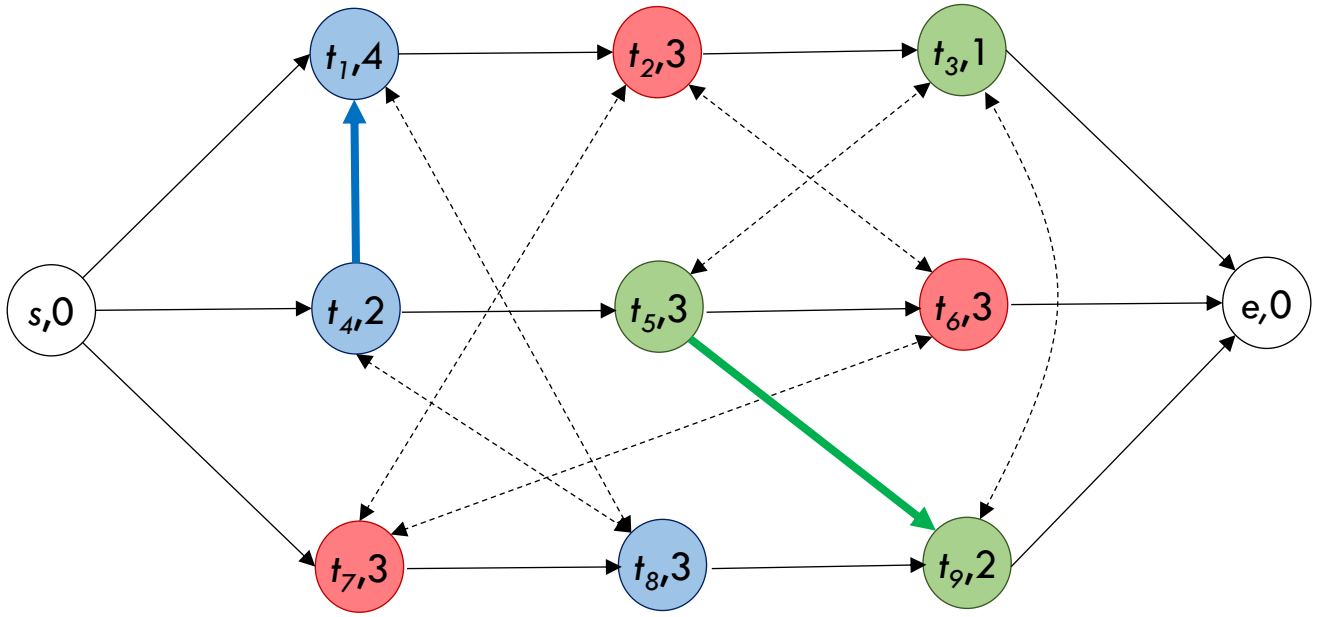




	s	t_4	t_7	t_5	t_1	t_6	t_8	t_2	t_9	t_3	e	
Cabezas	**0**	0	0	2	2	5	6	8	9	11	12	
	12	10	9	7	6	4	3	1	1	0	0	**Colas**

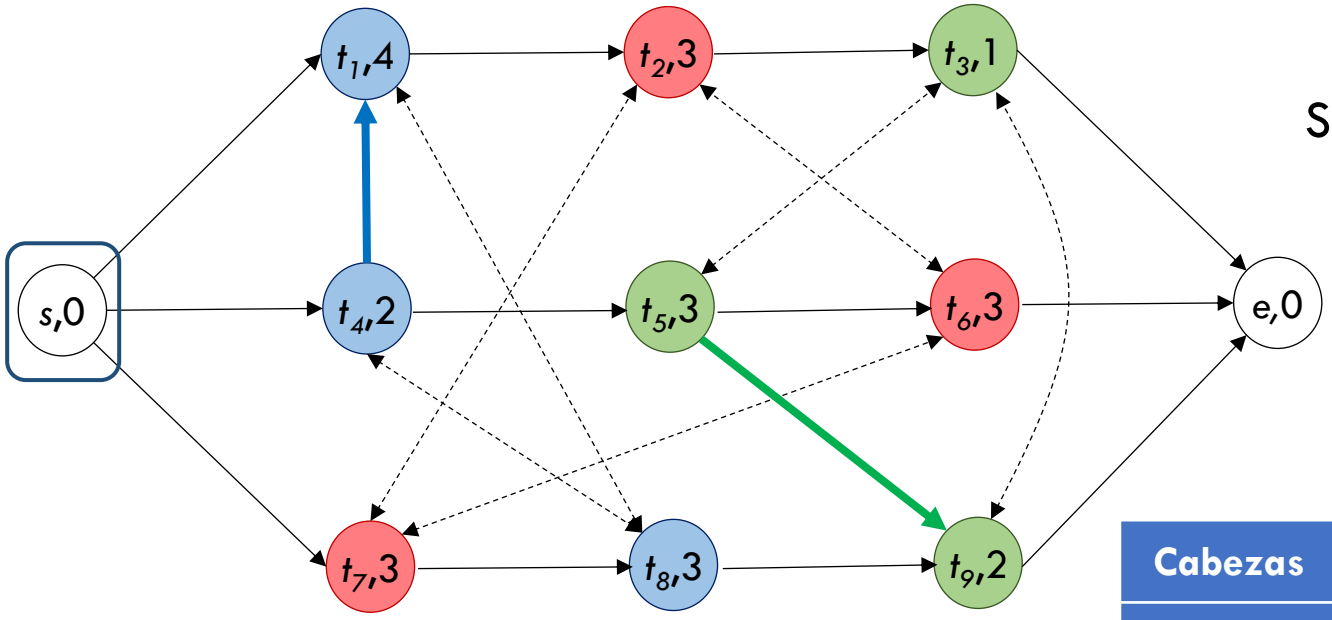
Grafo Solución Parcial

- Representa un Schedule parcial, es decir, algunos arcos disyuntivos están fijados y otros no.



Grafo Solución Parcial

- Representa un Schedule parcial, es decir, algunos arcos disyuntivos están fijados y otros no.



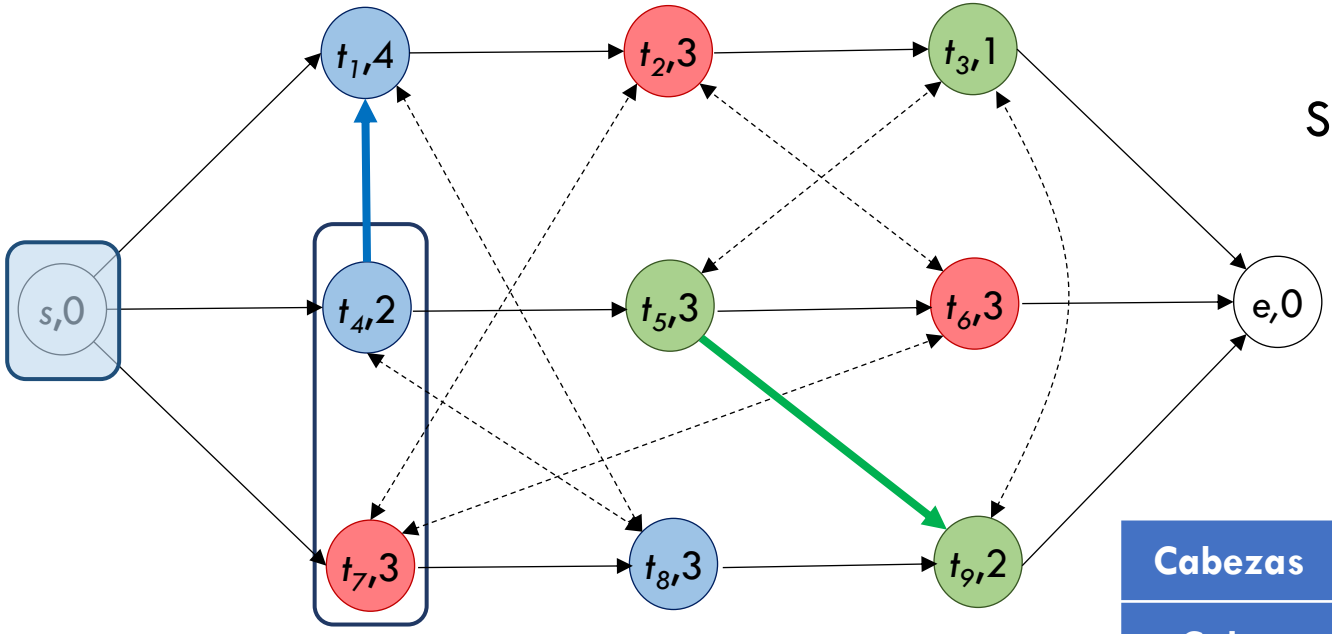
Seguir orden topológico:

$$\max\{(r+p)PJ, (r+p)PM(si\ fijada)\}$$

	s	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	e
Cabezas	0										
Colas											

Grafo Solución Parcial. Cabezas y Colas

- Cabeza y cola de t_i : coste máximo de s a t_i. y de t_i a e *respectivamente*. Son cotas inferiores del tiempo de inicio más temprano para t_i y del tiempo de procesamiento requerido tras t_i respectivamente



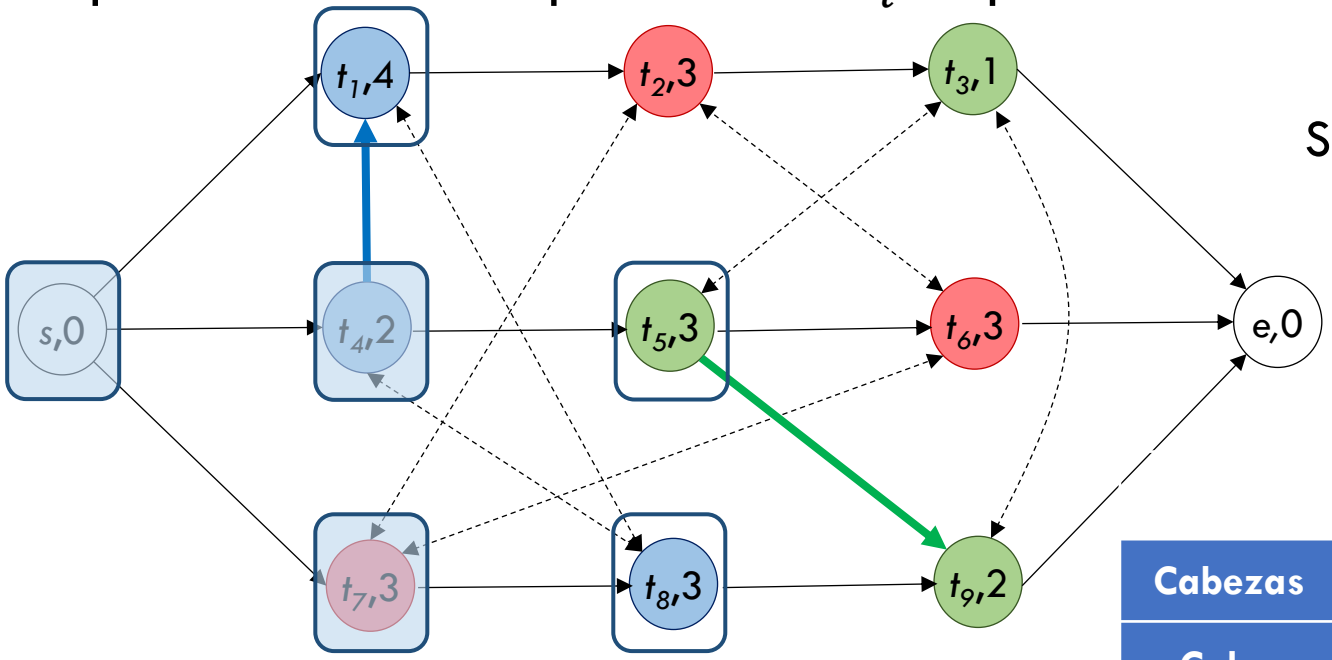
Seguir orden topológico:

$$\max\{(r+p)PJ, (r+p)PM(si\ fijada)\}$$

	s	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	e
Cabezas	0				0			0			
Colas											

Grafo Solución Parcial. Cabezas y Colas

- Cabeza y cola de t_i : coste máximo de s a t_i. y de t_i a e *respectivamente*. Son cotas inferiores del tiempo de inicio más temprano para t_i y del tiempo de procesamiento requerido tras t_i respectivamente



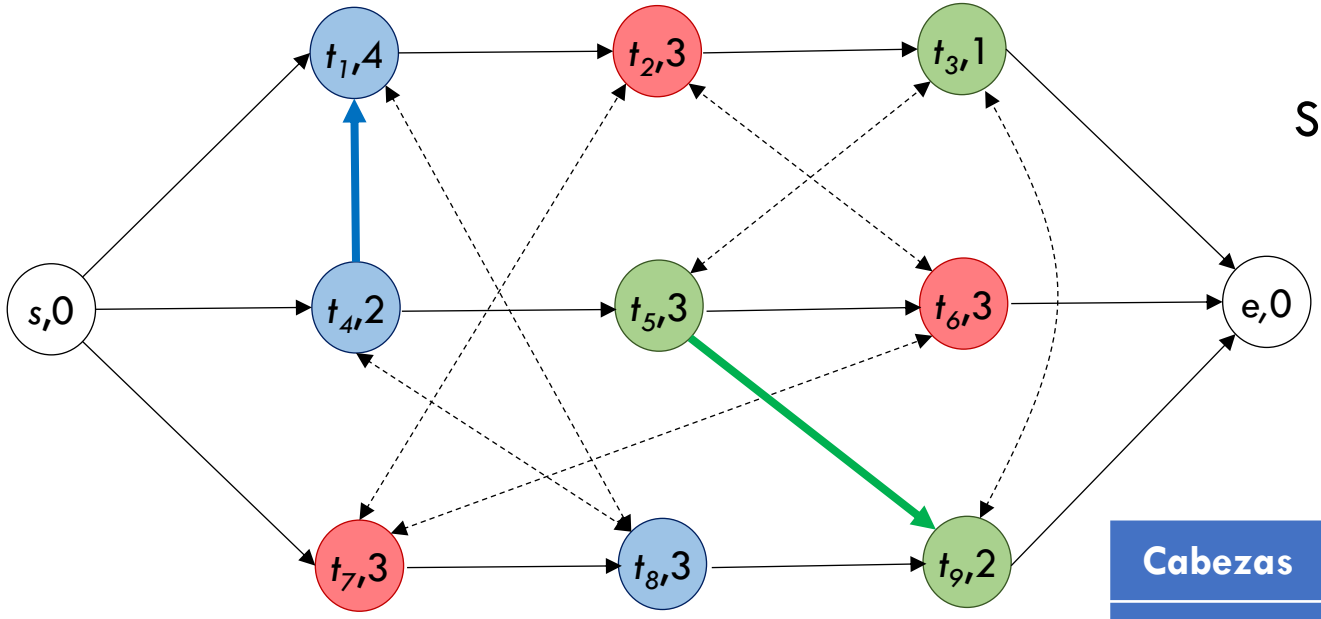
Seguir orden topológico:

$$\max\{(r+p)PJ, (r+p)PM(si\ fijada)\}$$

	s	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	e
Cabezas	0	2			0	2		0	3		
Colas											

Grafo Solución Parcial. Cabezas y Colas

- Cabeza y cola de t_i : coste máximo de s a t_i. y de t_i a e *respectivamente*. Son cotas inferiores del tiempo de inicio más temprano para t_i y del tiempo de procesamiento requerido tras t_i respectivamente



Seguir orden topológico:

$$\max\{(r+p)PJ, (r+p)PM(si\ fijada)\}$$

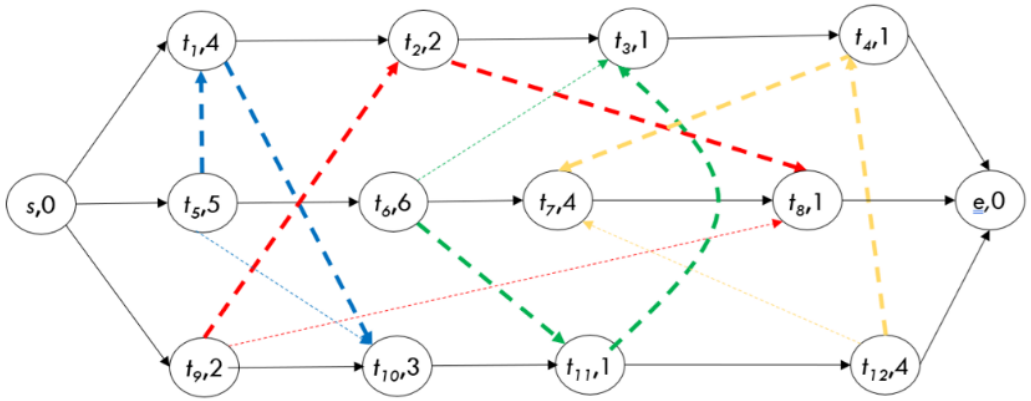
$$r_{t_3} = r_{t_2} + p_{t_2} = 6 + 3$$

$$r_{t_9} = max\{r_{t_5} + 3, r_{t_8} + 3\}$$

	s	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	e
Cabezas	0	2	6	9	0	2	5	0	3	6	10
Colas	10	4	1	0	8	3	0	5	2	0	0

Practiquemos un poco!!!

- Dado el grafo disyuntivo que representa una solución para el problema Job Shop Scheduling cuyos datos se muestran en la tabla de la derecha.



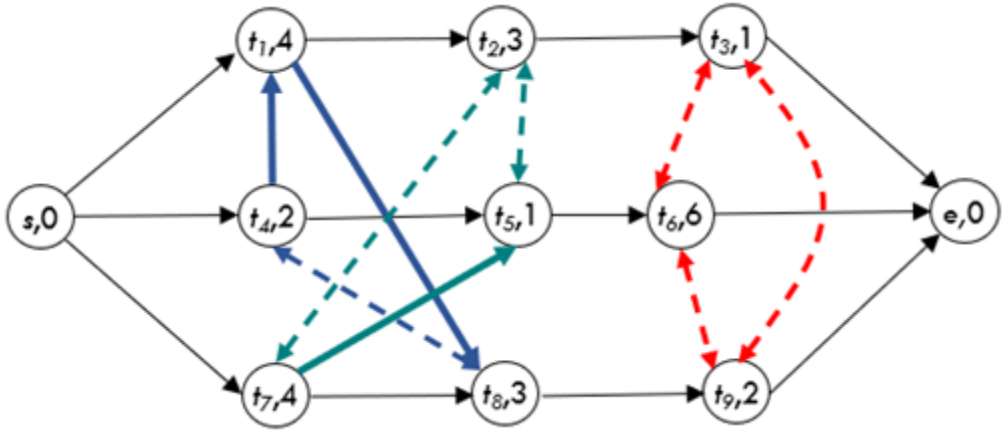
	$J_1, d_1 = 17$				$J_2, d_2 = 20$				$J_3, d_3 = 15$			
Tarea i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	4	2	1	1	5	6	4	1	2	3	1	4
m_i	0	1	2	3	0	2	3	1	1	0	2	3

a) Indica para las siguientes tareas sus cabezas y colas.

- $t_1: cab =$ $cola =$
- $t_2: cab =$ $cola =$
- $t_7: cab =$ $cola =$

Practiquemos un poco!!!

- Dado el siguiente grafo disyuntivo que representa una solución parcial, para un problema JSS, en la que sólo tenemos fijados los arcos (t_4, t_1), (t_1, t_8) y (t_7, t_5). Indique las cabezas y colas de las siguientes tareas.



a) Indica para las siguientes tareas sus cabezas y colas.

- $t_1: cab =$ $cola =$
- $t_2: cab =$ $cola =$
- $t_6: cab =$ $cola =$
- $t_9: cab =$ $cola =$

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - **Espacios de soluciones**
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

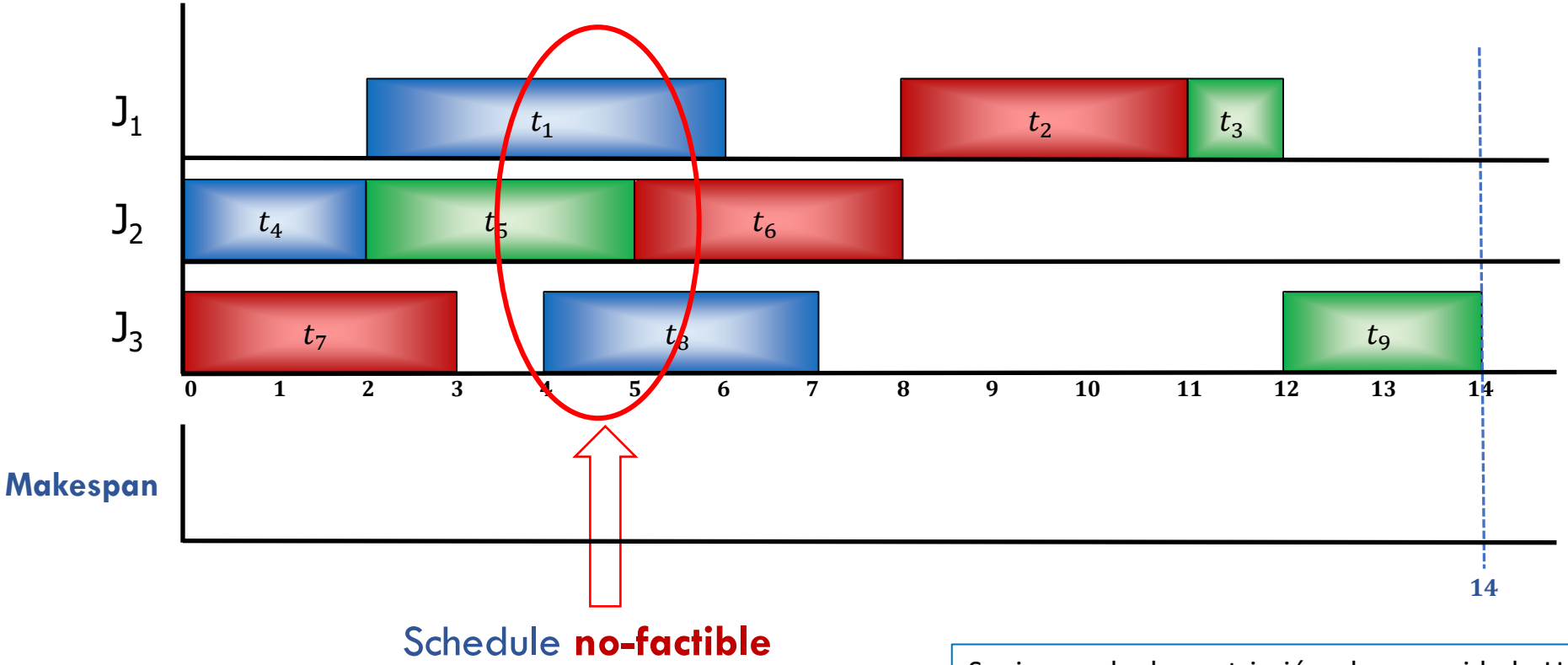
Espacios de Soluciones

- Schedules no-factibles (*Infeasible*)
- Schedules factibles (*Feasible*)
  - Schedules semi-activos (*Semi-active*)
  - Schedules activos (*Active*)
  - Schedules óptimos (*Optimal*)
  - Schedules densos (*Non-delay*)

Espacios de Soluciones

- **Schedules semi-activos**
  - Para que una tarea pueda comenzar antes, al menos hay que cambiar el orden de dos tareas.
  - Es decir, cada tarea empieza "lo antes posible" después de las planificadas antes que ella
- **Schedules activos**
  - Para que una tarea pueda comenzar antes, al menos otra debe retrasarse
  - No hay "huecos en las máquinas" en la planificación donde puedan ejecutarse tareas que empiezan más tarde
- **Schedules densos**
  - Ninguna máquina permanece ociosa si hay una tarea disponible para ejecutarse en ella

Espacios de Soluciones

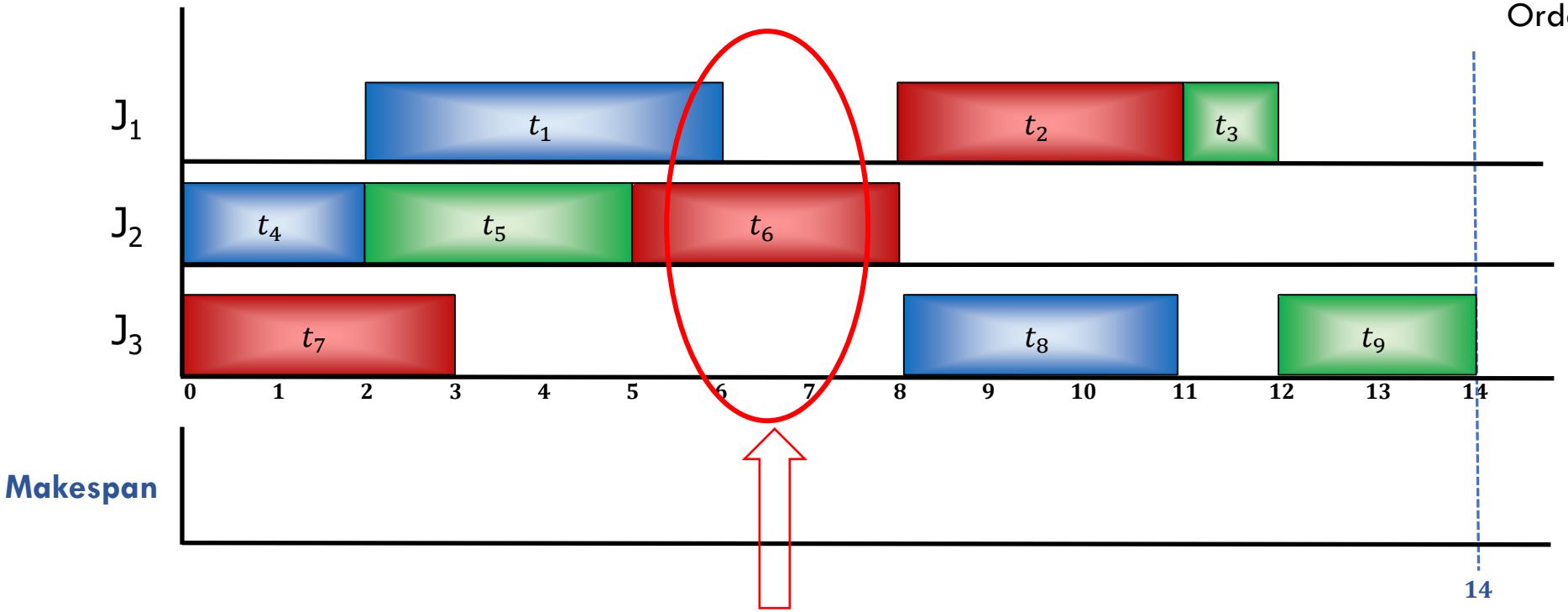


Schedule **no-factible**

Se incumple la restricción de capacidad. Una máquina sólo puede procesar una tarea de cada vez, por tanto, dos tareas que emplean la misma máquina no se pueden solapar en el tiempo.

Espacios de Soluciones

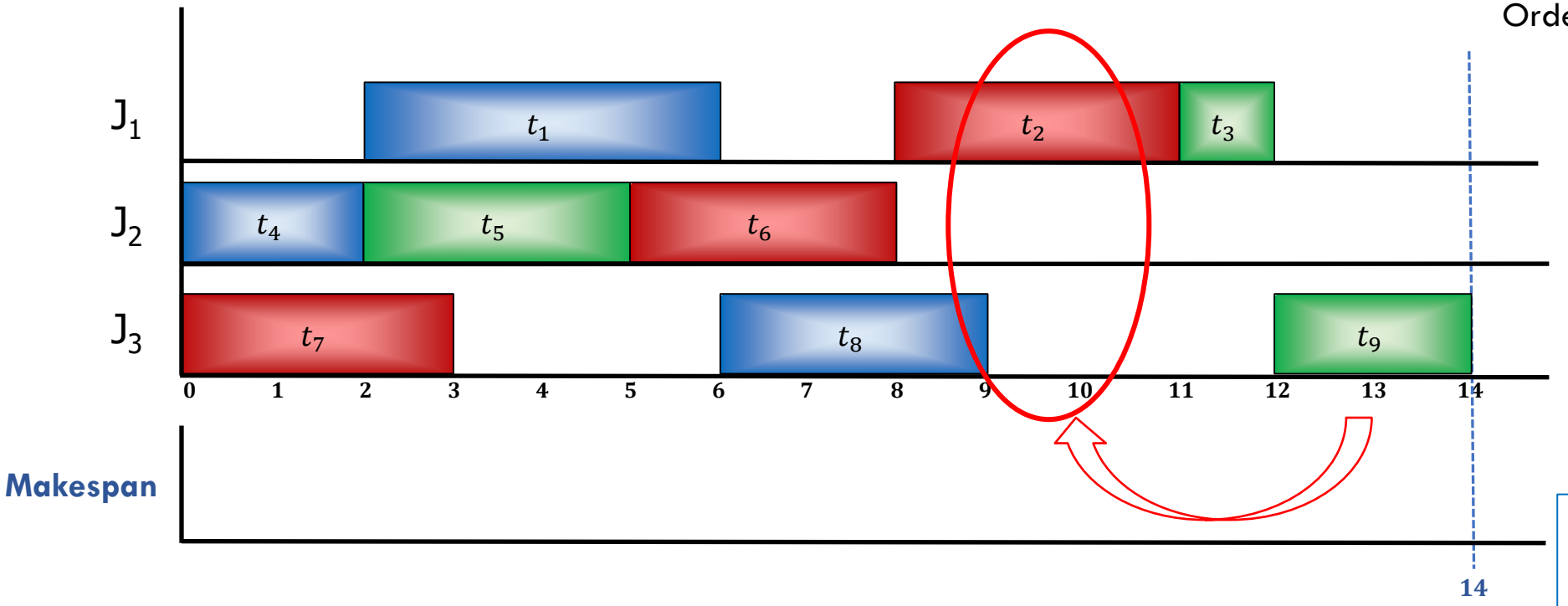
Orden de ejecución de las tareas
$(4,7,1,5,6,\mathbf{8},2,3,9)$



Schedule factible **no-semiactivo**

La tarea t_8 no comienza "lo antes posible" después de las planificadas antes que ella !!! Podría comenzar en el instante 6 sin incumplir ninguna restricción del problema.

Espacios de Soluciones



Orden de ejecución de las tareas

$(4,7,1,5,6,8,2,\mathbf{9},3)$

La tarea t_9 puede planificarse en el instante 9 sin incumplir ninguna restricción, ni retrasar ninguna tarea.

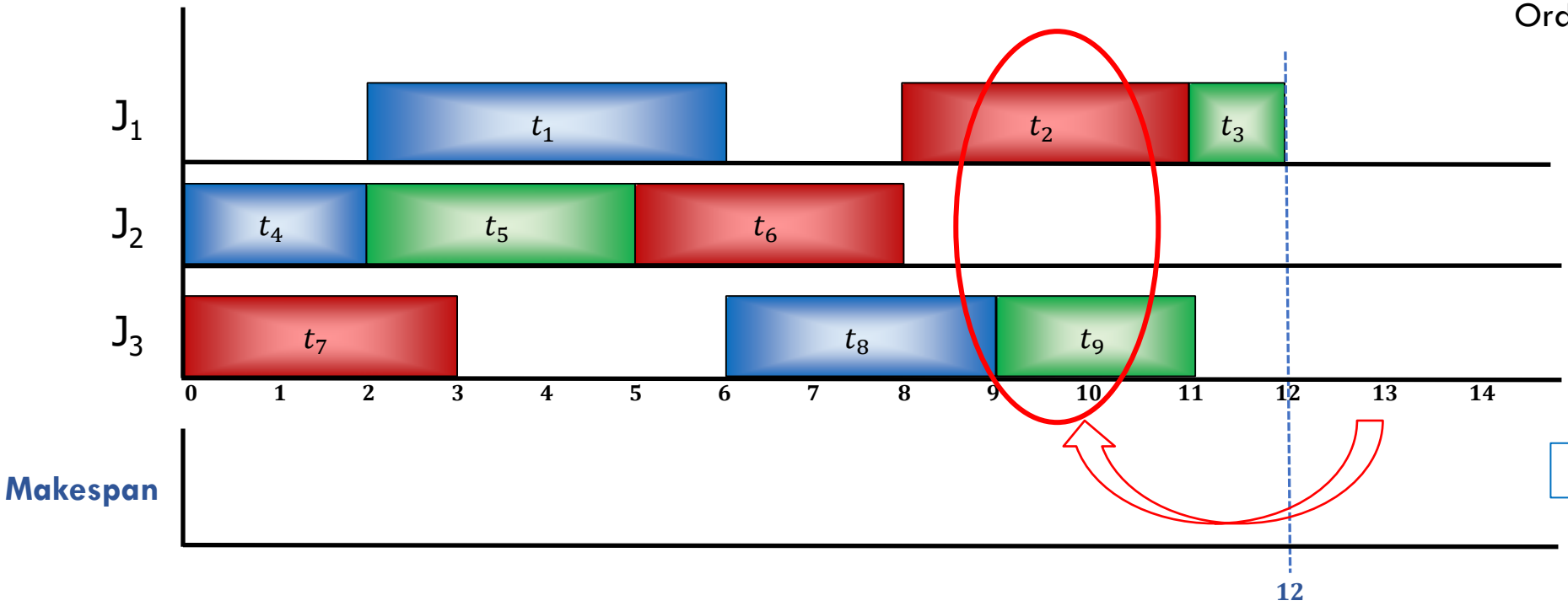
Schedule factible:

- **semiactivo**
- **no-activo:** si se adelanta la tarea t_9, la tarea t_3 no habría que retrasarla

Espacios de Soluciones

Orden de ejecución de las tareas

$(4,7,1,5,6,8,2,\mathbf{9},3)$



El makespan se reduce a 12.

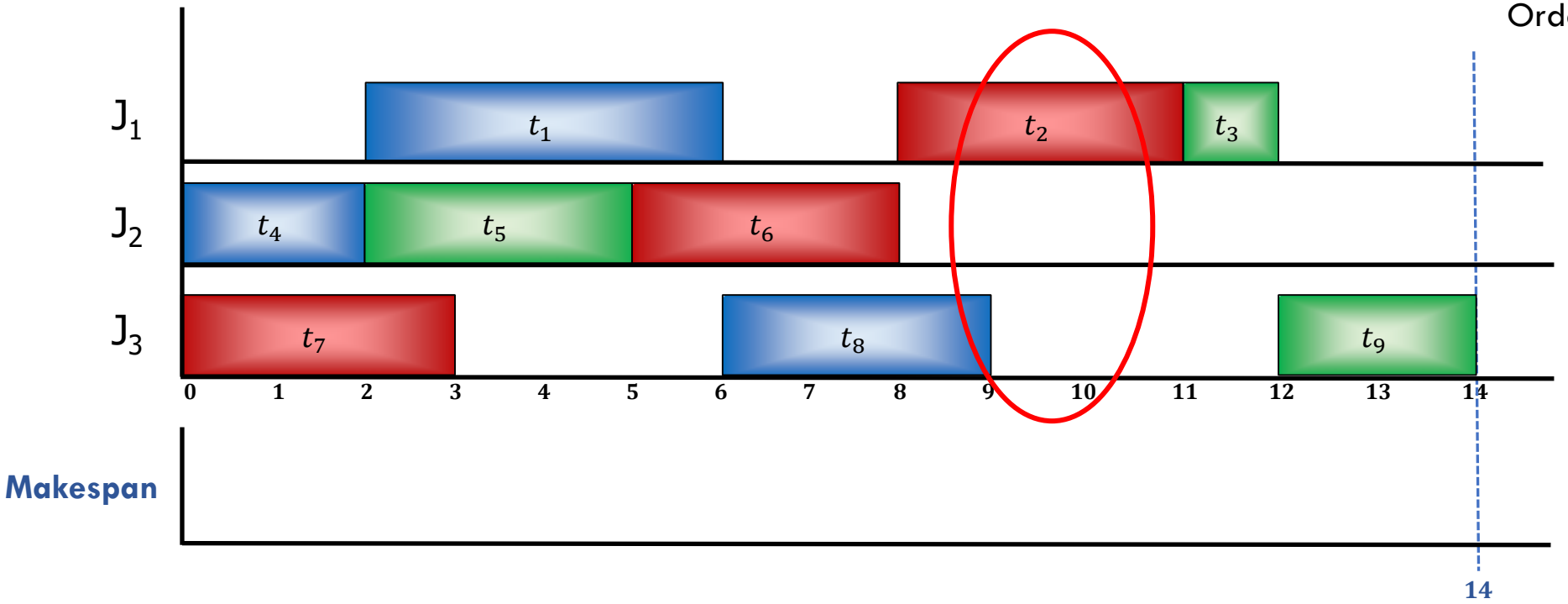
Schedule factible:

- **semiactivo**
- **activo,** ahora ya no se puede adelantar ninguna tarea sin retrasar otra

Espacios de Soluciones

Orden de ejecución de las tareas

$(4,7,1,5,6,8,2,\mathbf{9},3)$

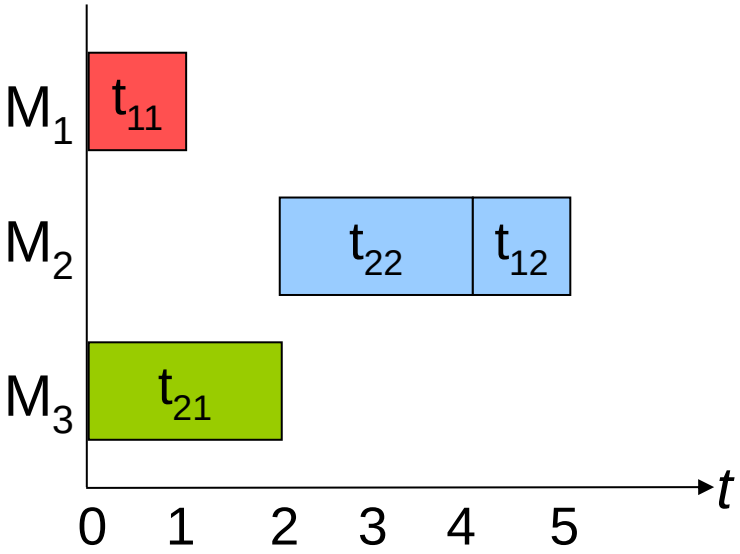


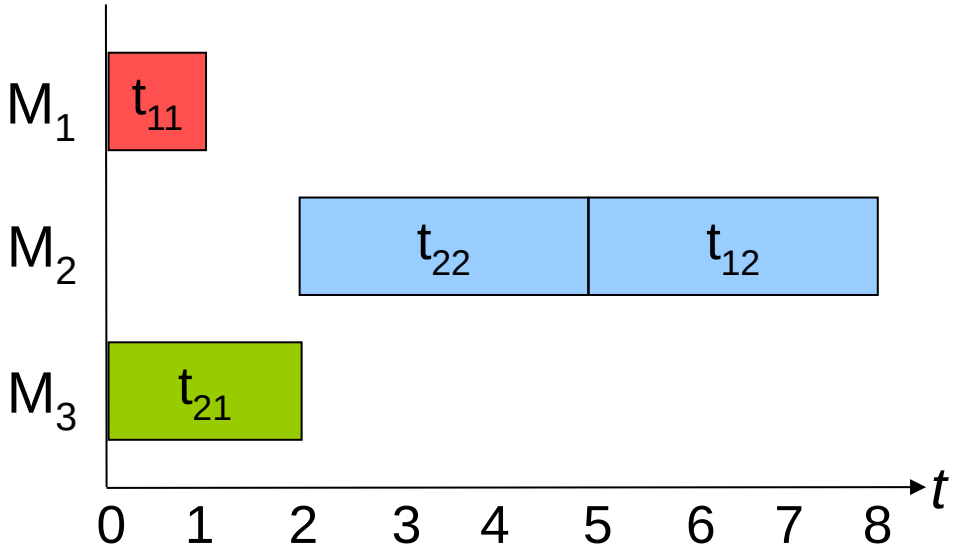
Schedule factible:

- **semiactivo, no-activo**
- **no-denso:** en el instante 9 la máquina 3 (verde) está inactiva y hay una tarea (la t_9) que está lista para ejecutarse

Espacios de Soluciones

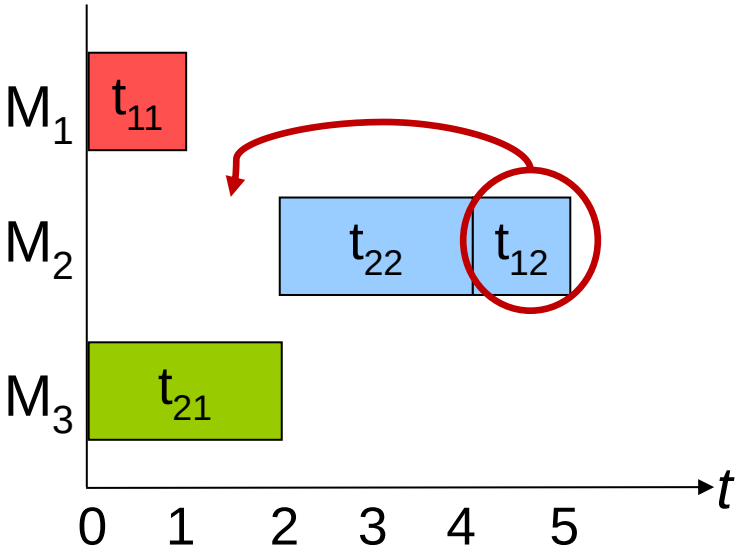
¿Cómo son las siguientes planificaciones?



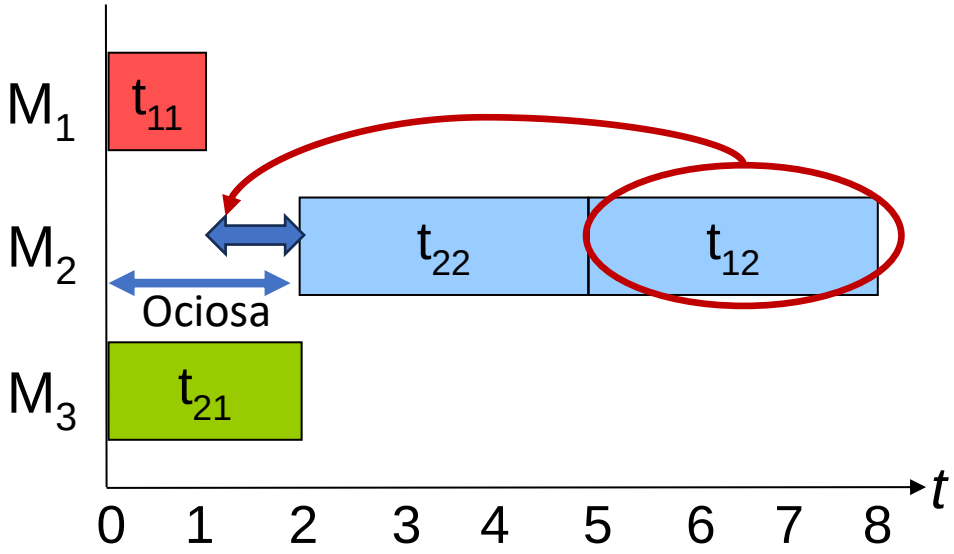


Espacios de Soluciones

¿Cómo son las siguientes planificaciones?

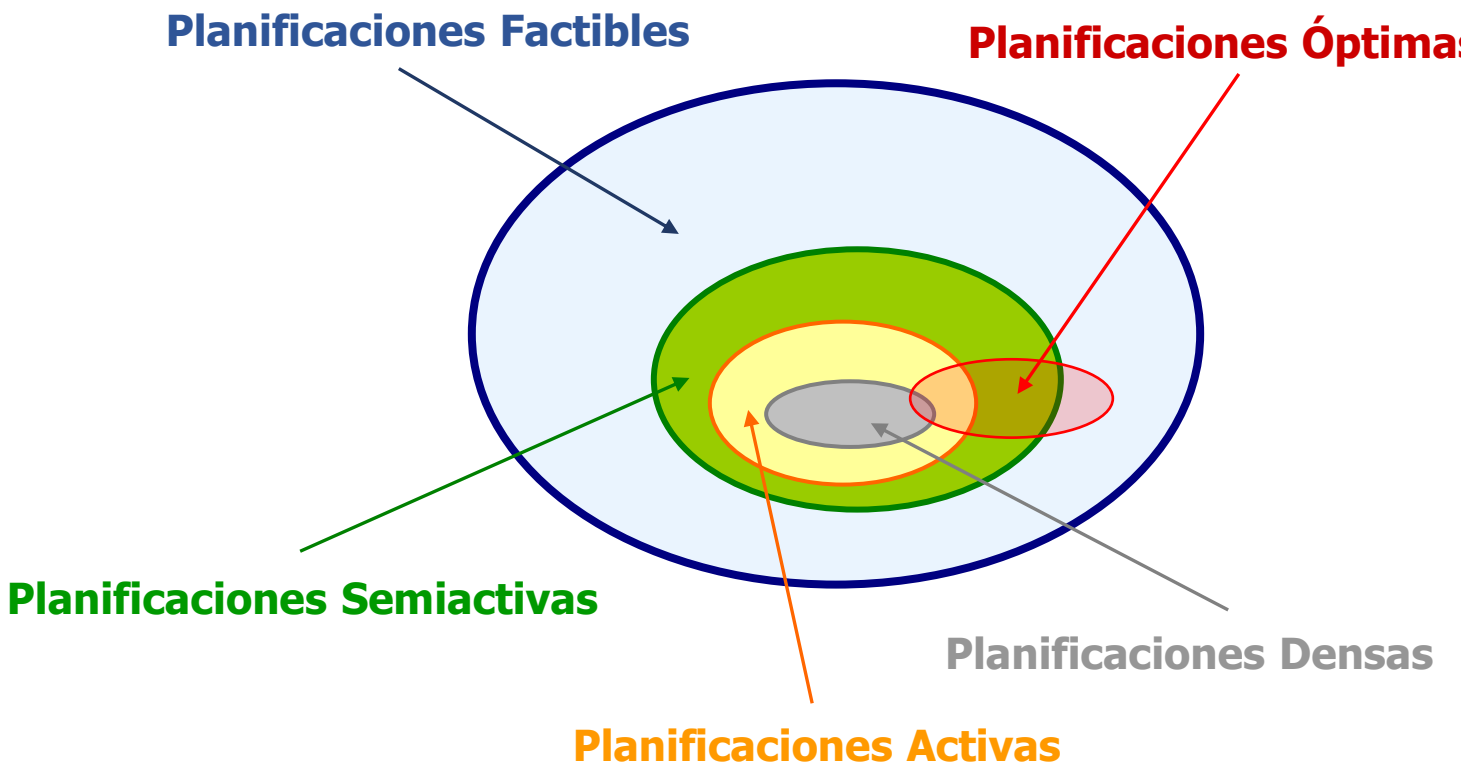


Planificación Semiactiva no Activa



Planificación Activa no Densa

Espacios de Soluciones



Planif. Densas ⊂ Planif. Activas ⊂ Planif. Semiactivas ⊂ Planif. Factibles

Practiquemos un poco!!!

Representa esta planificación con el FT03. ¿De qué tipo son las siguientes planificaciones con el mismo coste?

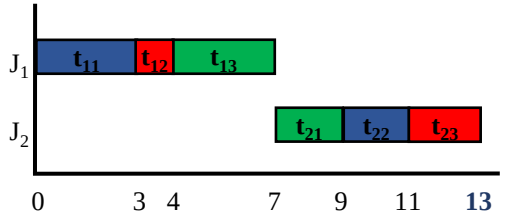
Si no es factible, ¿cómo la podrías convertir en factible?

Si no es semi-activa, ¿cómo la podrías convertir en semi-activa?

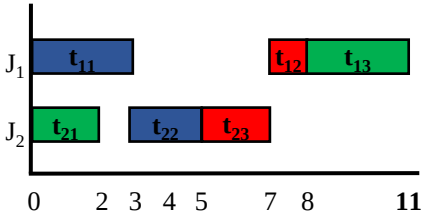
Si no es activa, ¿cómo la podrías convertir en activa?

Si no es densa, ¿cómo la podrías convertir en densa?

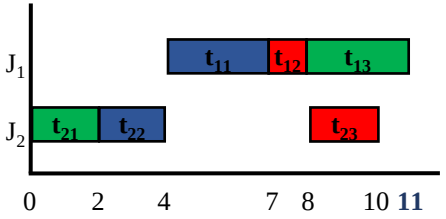
Todas
Planificaciones semi-activas



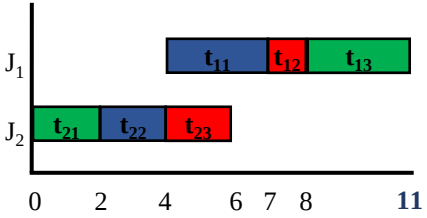
No activa



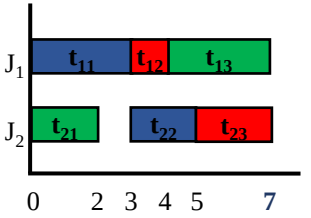
No activa



No activa



Activa no densa

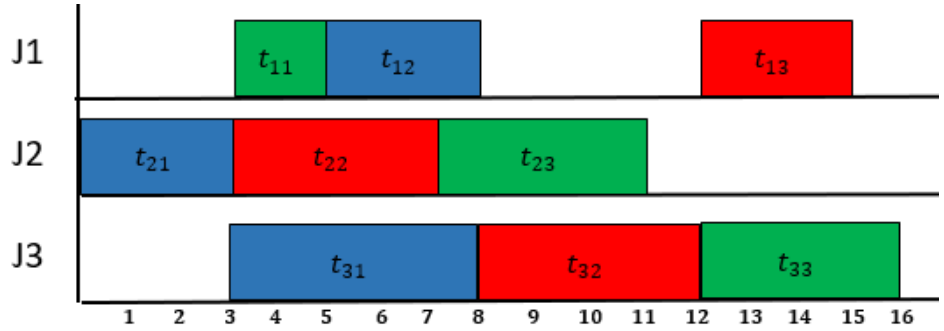


Activa y densa

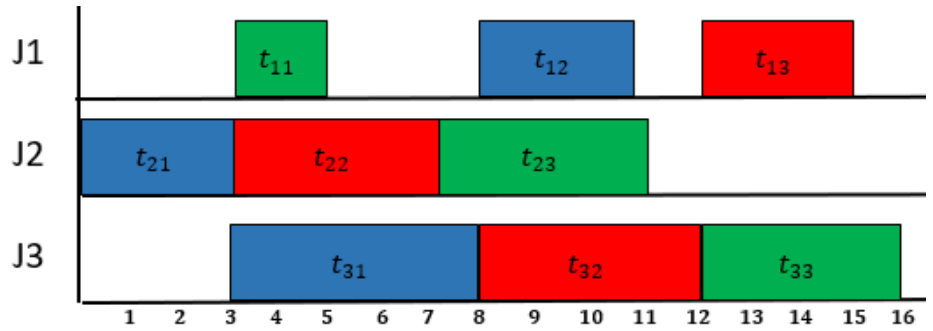
Practiquemos un poco!!!

¿De qué tipo son las siguientes planificaciones?

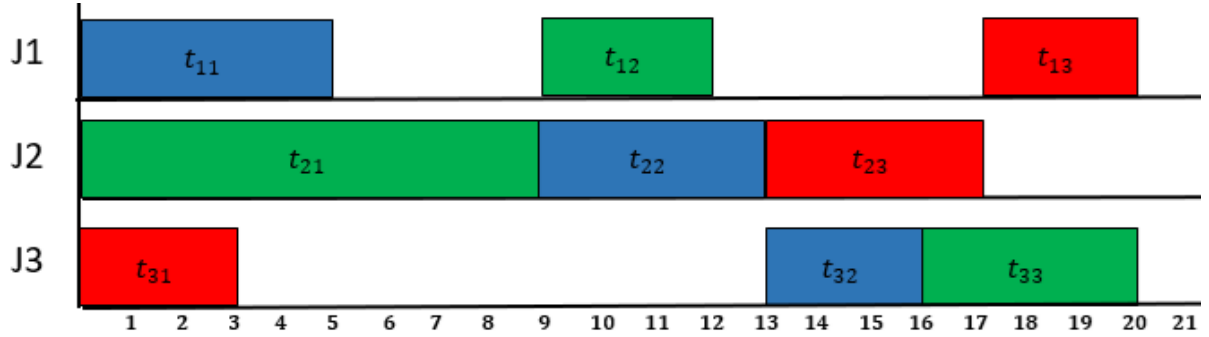
A



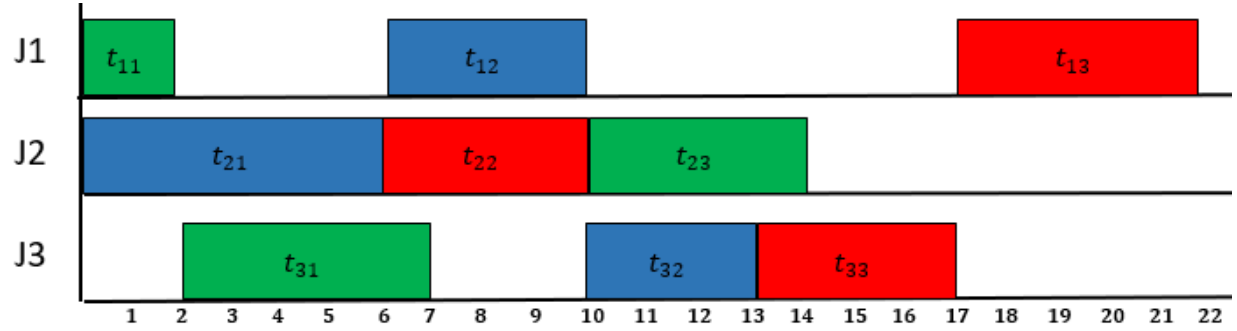
B



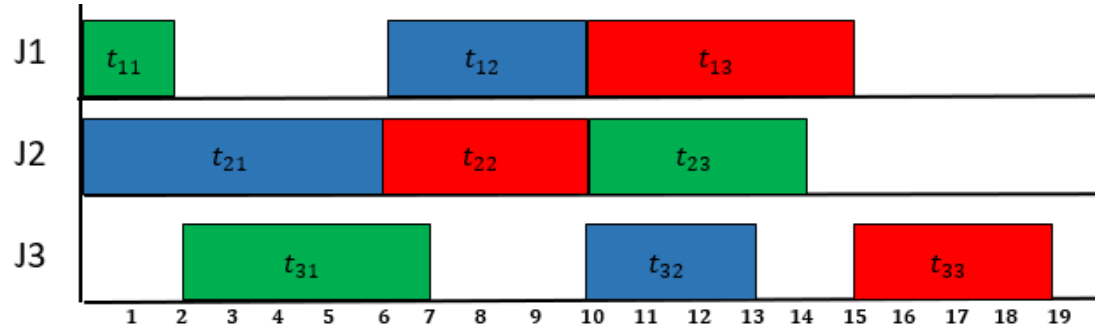
C



D

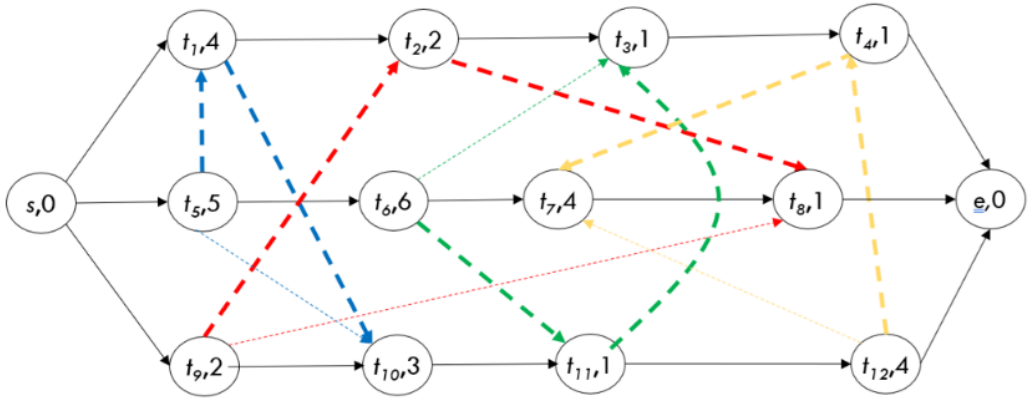


E



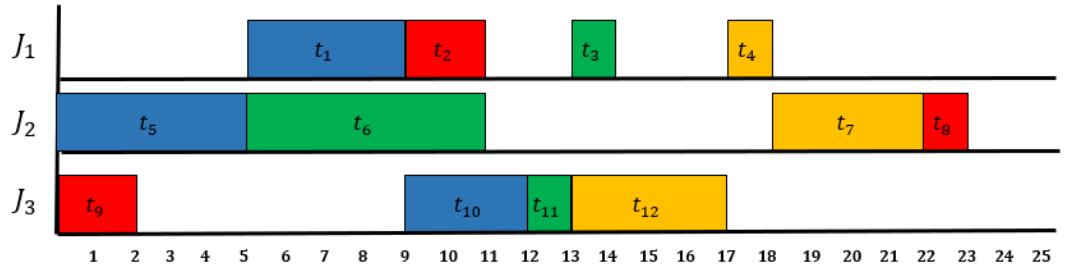
Practiquemos un poco!!!

- Dado el grafo disyuntivo que representa una solución para el problema Job Shop Scheduling cuyos datos se muestran en la tabla de la derecha.



	$J_1, d_1 = 17$				$J_2, d_2 = 20$				$J_3, d_3 = 15$			
Tarea i	1	2	3	4	5	6	7	8	9	10	11	12
p_i	4	2	1	1	5	6	4	1	2	3	1	4
m_i	0	1	2	3	0	2	3	1	1	0	2	3

- Siendo el siguiente un diagrama de Gannt una planificación compatible con la solución representada por el grafo solución, indica de que tipo es (semi-activa, activa, densa). En caso de que no sea semi-activa, activa y/o densa, explica el por qué.



Semi-activa(SI/NO): .No es semi-activa pues:

Activa(SI/NO): . No es activa pues:

Densa(SI/NO): . No es activa pues:

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - **Cómo resolver el JSS**
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensions del JSSP
5. Referencias Bibliográficas

Cómo resolver el JSSP

- Esquemas de generación de Schedules (SGS, *Schedule Generation Schema*) y Reglas de prioridad (*Dispatching rules*)
- Métodos exactos heurísticos: Búsqueda en espacio de estados y Ramificación y Poda ([*Brucker1994*])
- Métodos aproximados: Metaheurísticas
- Solvers comerciales: Minizinc, CP Optimizer (IBM ILOG CPLEX CP Optimizer [Laborie, 2009]), Comet (Dynadec [Michel and Hentenryck 2003])

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - **Esquemas Generadores de Schedules**
   - Reglas de Prioridad
   - Extensiones del JSSP
5. Referencias Bibliográficas

Esquemas de generación de schedules

Algoritmo SGS genérico

Recibe: una instancia de JSSP, $\boldsymbol{P}$, y un orden de procesamiento $\boldsymbol{\pi}$
Devuelve: un Schedule $\boldsymbol{st}$ para $\boldsymbol{P}$ de acuerdo con

1. $A = \{o_{j1} : 1 \leq j \leq n\}$ # Primeras tareas sin planificar de cada trabajo
2. Mientras $A \neq \emptyset$ hacer
   a. Calcular el conjunto elegible $B \subseteq A$
   b. Seleccionar $o_{j^*k^*} = argmin\{\pi_{jk} : o_{jk} \in B\}$ # Tarea antes en el orden de procesamiento
   c. $st_{j^*k^*} = ES_{j^*k^*}$ # Mínimo tiempo de iniciò para esa tarea
   d. $A = A - \{o_{j^*k^*}\} \cup \{o_{j^*(k+1)^*} \; si \; k^* < m_{j^*}\}$ # Añadimos la siguiente tarea en el trabajo
3. Devuelve st

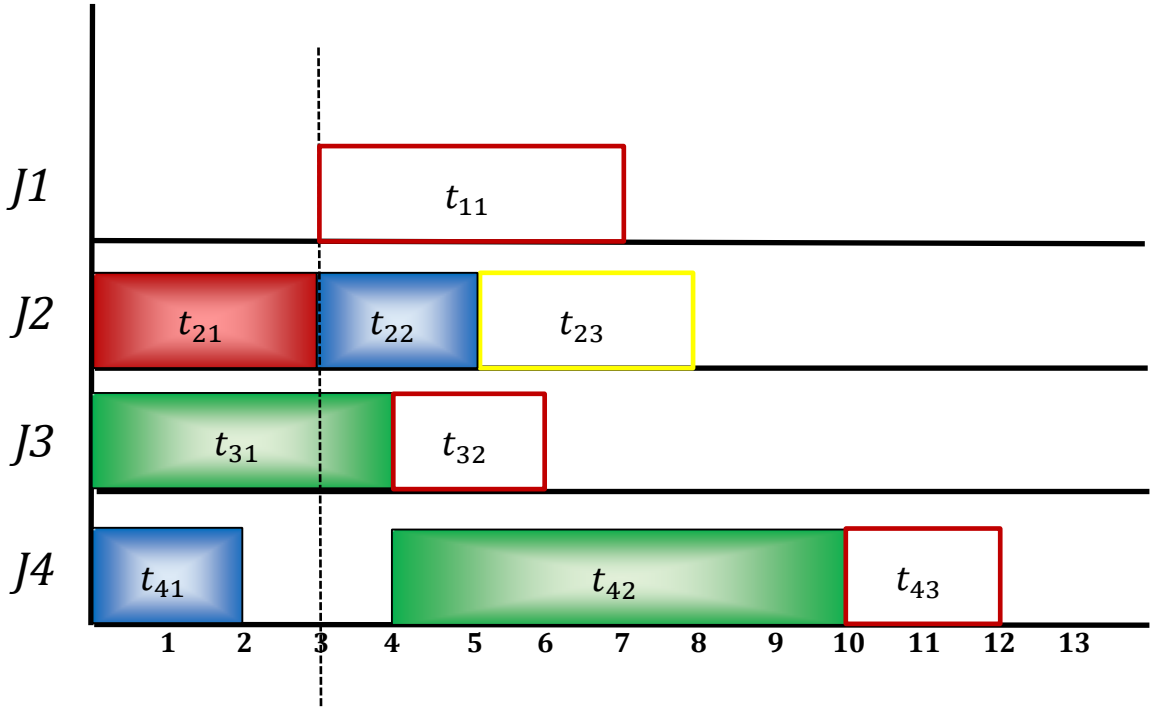
Schedules semi-activos: $\mathbf{B} = A$ y $ES_{j^*k^*} = \max\{C_{PJ_{jk}}, C_{PM_{jk}}\}$ (mínimo tiempo de inicio para la tarea, calculado como el máximo entre los tiempos de completud de la tareas predecesoras en el trabajo y en la máquina)

Schedules activos-inserción: $\mathbf{B} = A$ y $ES_{j^*k^*} = \min\{st \geq C_{PJ_{jk}} : st \; es \; inserción \; factible \; para \; o_{j^*k^*} \; en \; M_{j^*k^*}\}$ (mínimo tiempo de inicio para la tarea, posterior al de fin de la tarea anterior en el trabajo, que se le puede asignar a la tarea $o_{j^*k^*}$, si hay un hueco en la planificación, en el que la máquina $M_{j^*k^*}$ esté libre para ejecutar completamente la tarea $o_{j^*k^*}$.

SGS. Planificador Básico

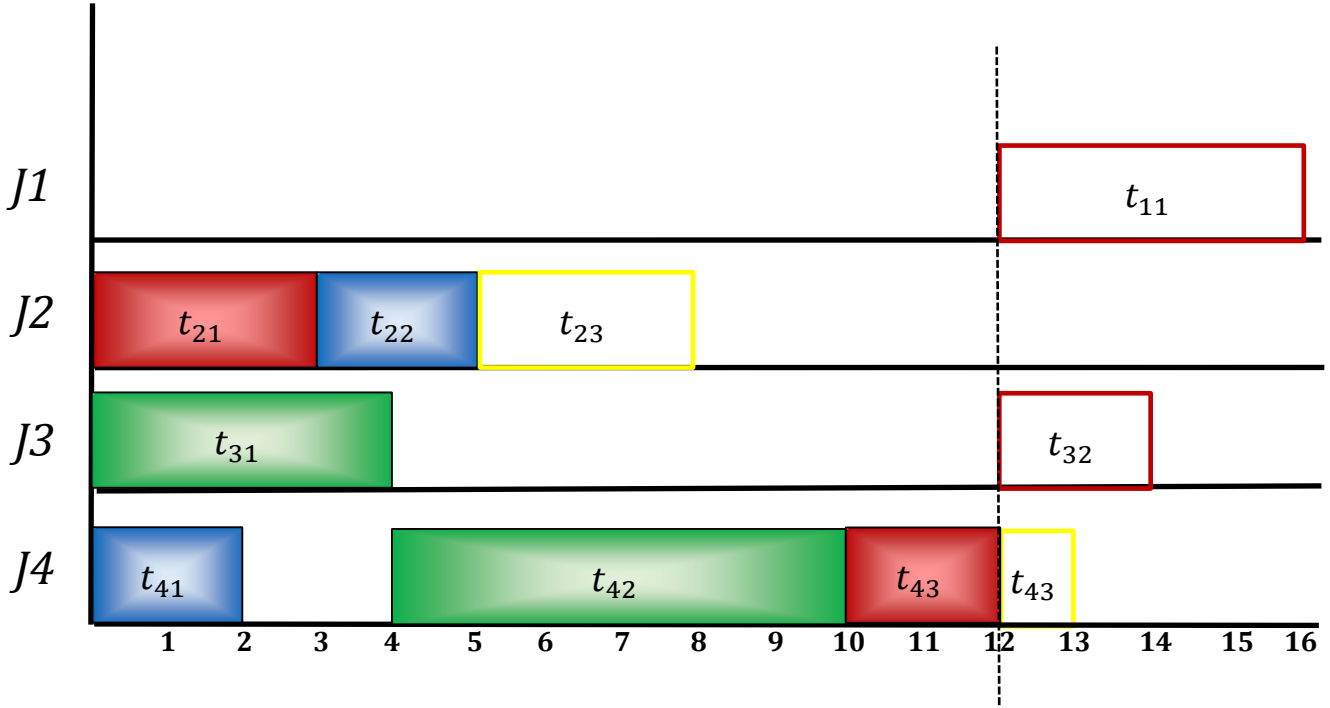
- Podemos modificar el SGS genérico para que no requiera que se le pase un orden de procesamiento de las tareas, sino que se le pase un criterio (función) que, en cada paso de la ejecución del algoritmo, ayude a tomar la decisión de la siguiente tarea a planificar de entre las que pueden comenzar, conjunto A.
  - Criterios posibles: Aleatoria, tarea perteneciente al trabajo de menor índice, tarea más corta, etc…
- En adelante a al SGS genérico modificado que construye planificaciones semi-activas lo denominaremos "**Planificador Básico**". El criterio (función) que ayuda a tomar la decisión de la siguiente tarea a planificar puede ser no determinista.
  - No Determinista: No obtendrá las mismas soluciones en todas las ejecuciones.

SGS semiactivo



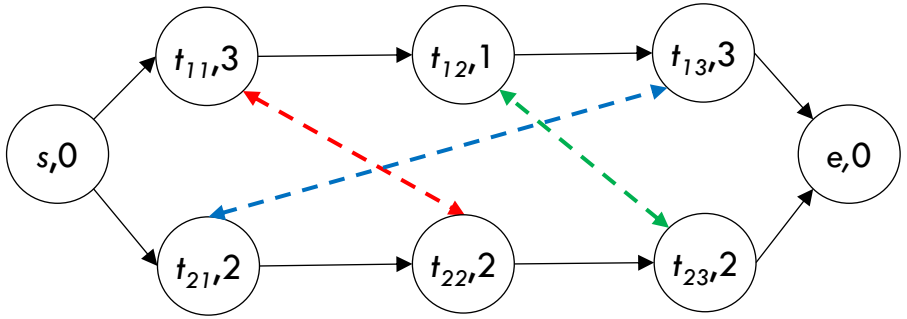
- $A = \{t_{11}, t_{23}, t_{32}, t_{43}\}$
  - Elegir la tarea que aparezca antes en el orden de procesamiento dado.
- Si fuera la tarea t_{43}

SGS semiactivo

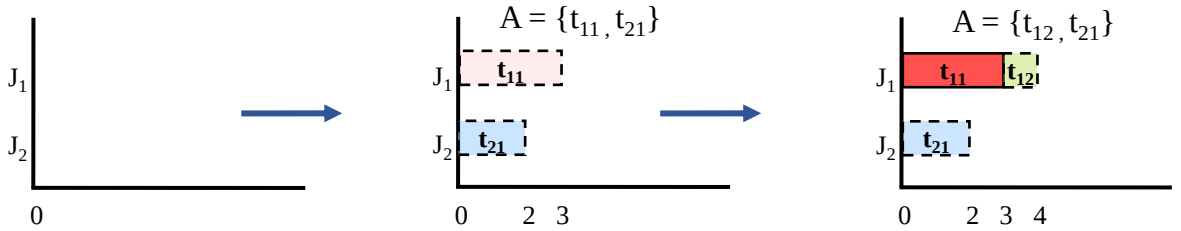


- $A = \{t_{11}, t_{23}, t_{32}, t_{43}\}$

Ejemplo. SGS Semiactivo



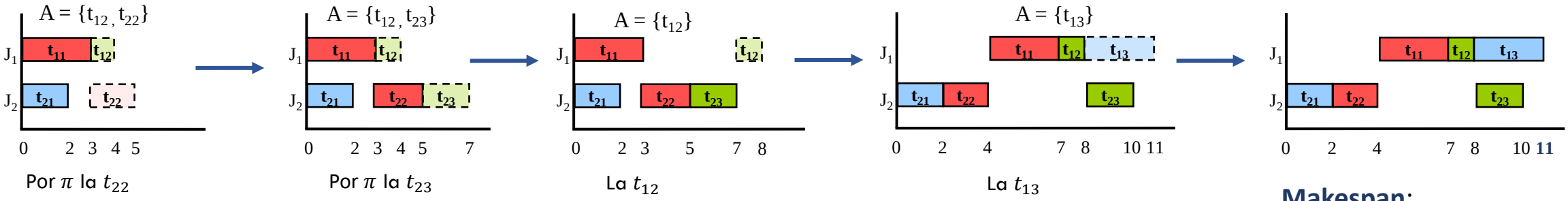
$$\pi = (s, t_{11}, t_{21}, t_{22}, t_{23}, t_{12}, t_{13}, e)$$



Se puede elegir la t_{11}ó $la\ t_{21}$
En π la t_{11} va antes que la t_{21}
Planificar la t_{11}

Por π la t_{21}

Planificación Determinista



Por π la t_{22}

Por π la t_{23}

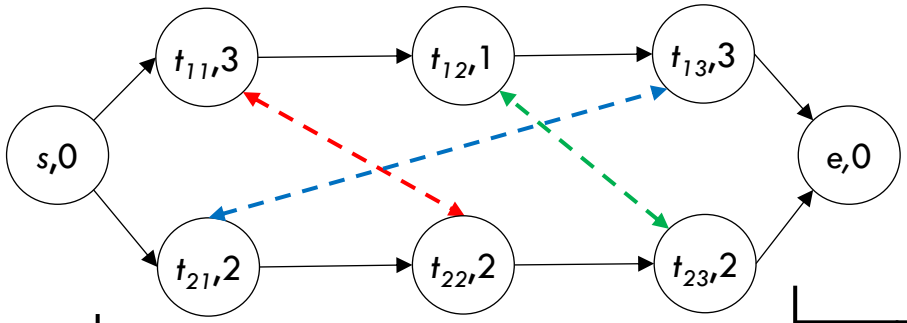
La t_{12}

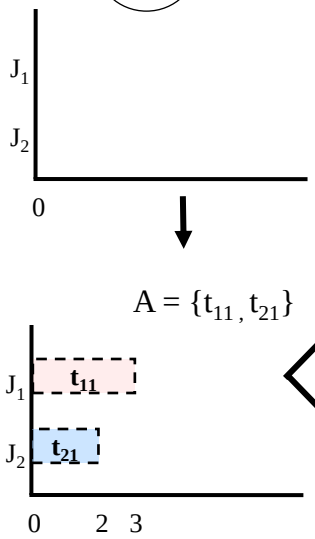
La t_{13}

Makespan:

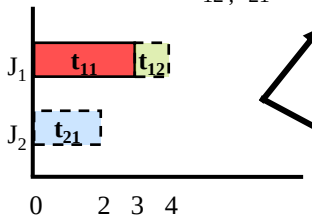
$$C_{max} = \max\{10,11\} = 11$$

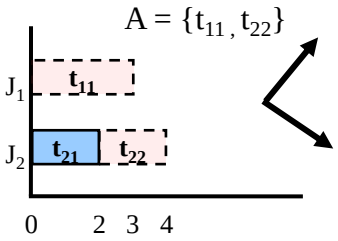
Ejemplo. SGS Semiactivo
Espacio de búsqueda completo

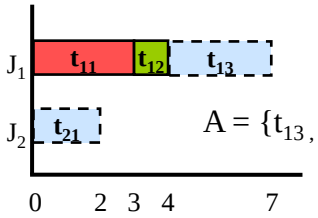


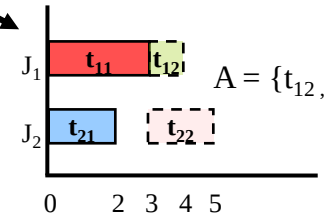


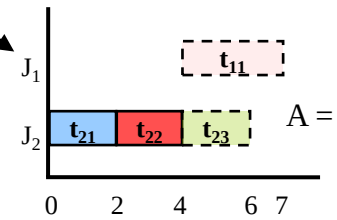
Se puede elegir la t_{11}ó *la* t_{21}

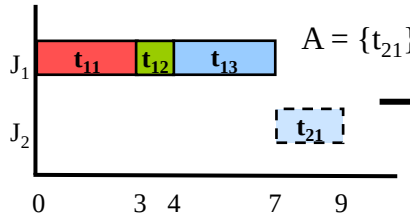


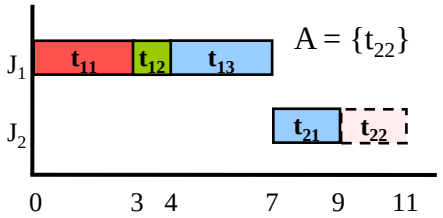


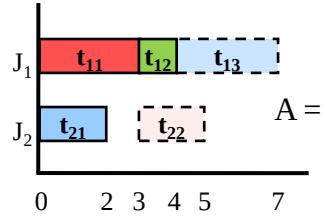


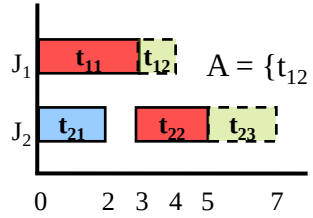


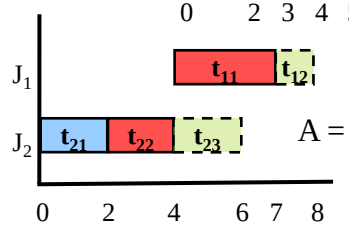


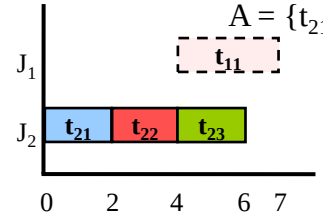




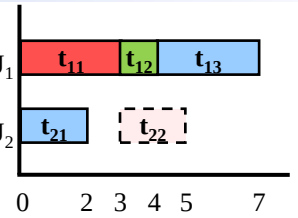




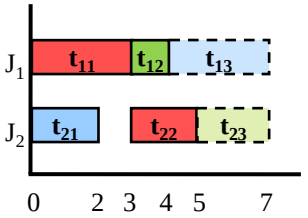




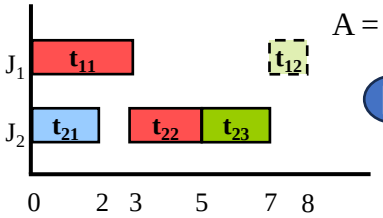
A



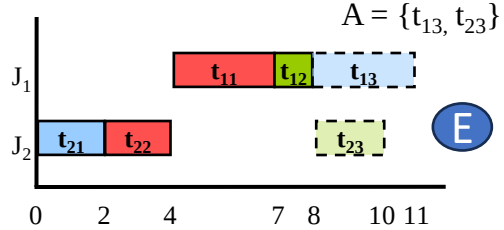
B



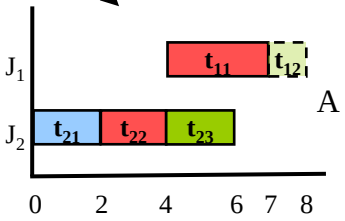
C



D

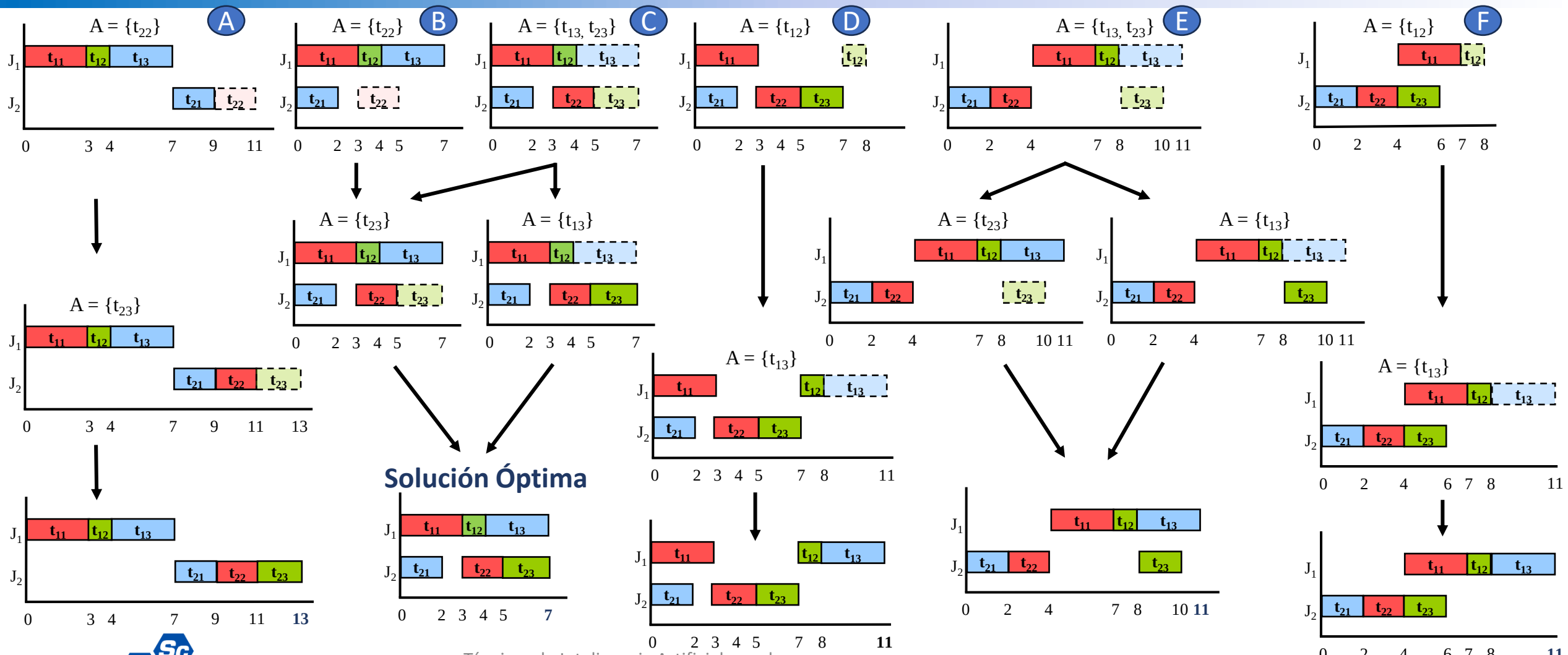


E

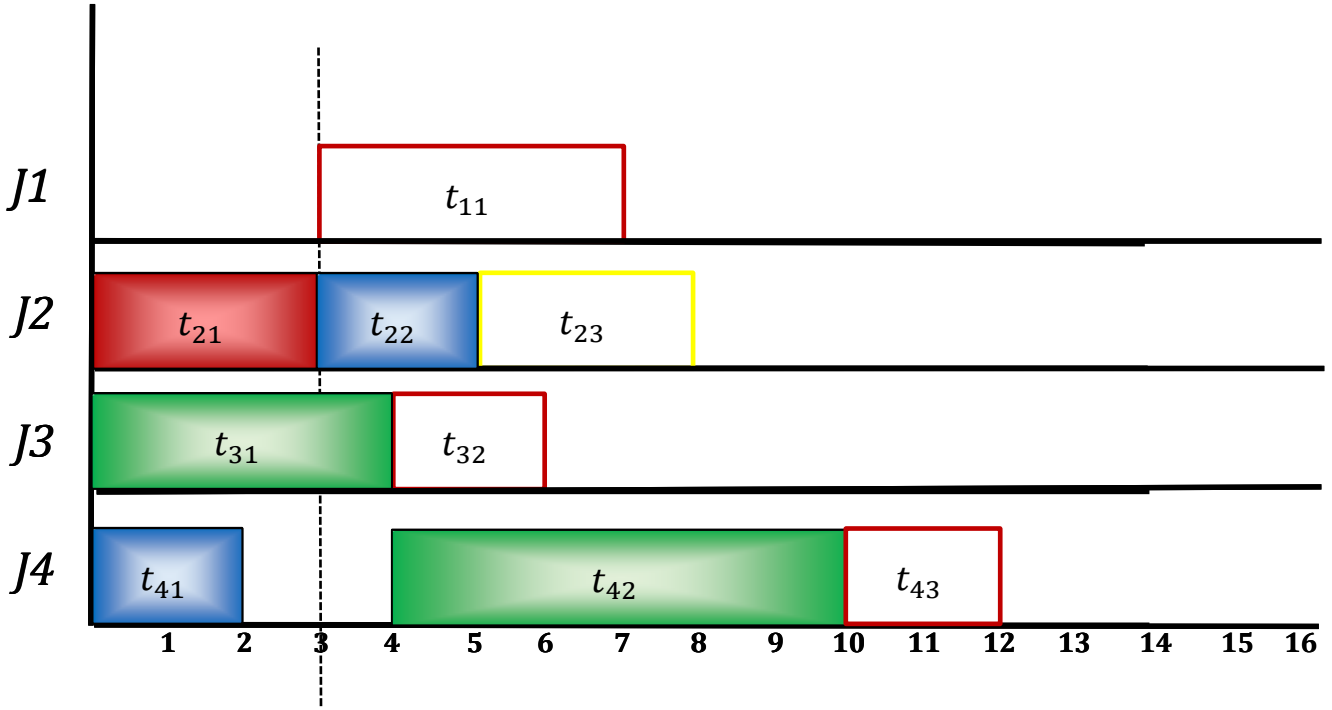


F

Ejemplo. SGS Semiactivo
Espacio de búsqueda completo

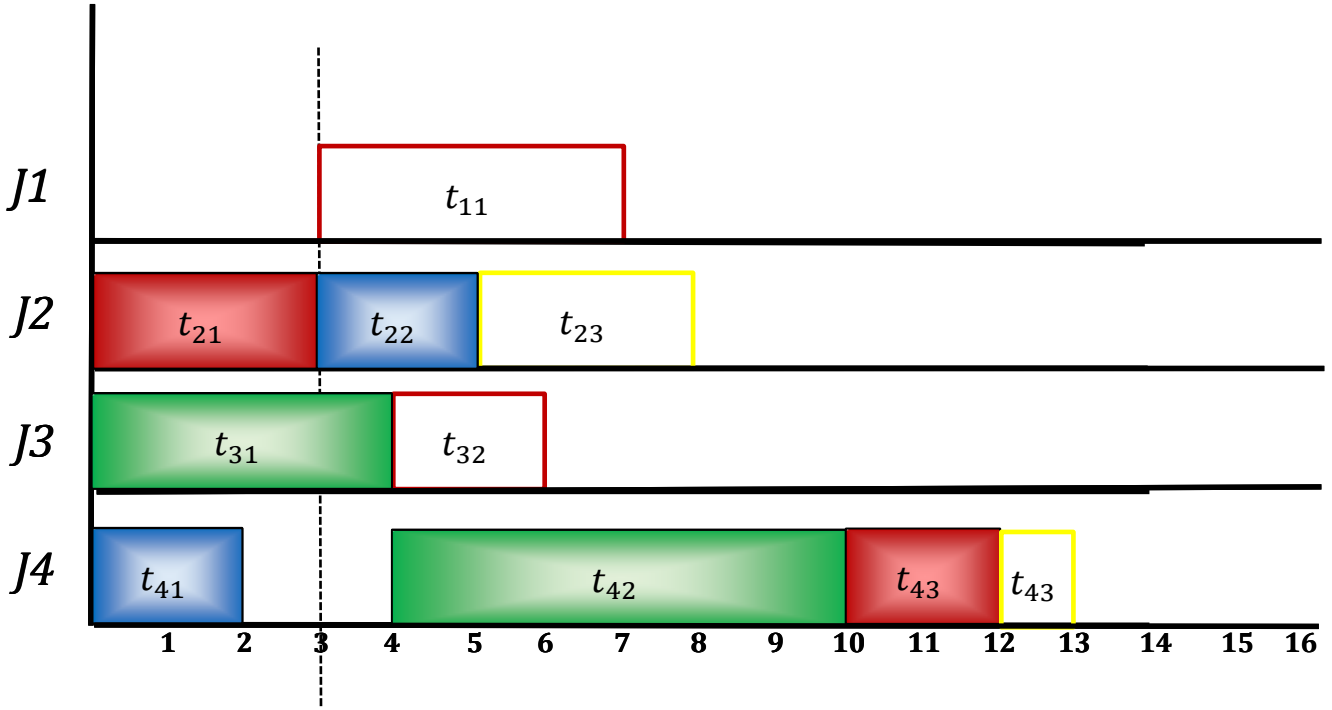


SGS activo por inserción



- $A = \{t_{11}, t_{23}, t_{32}, t_{43}\}$
  - Elegir la tarea que aparezca antes en el orden de procesamiento dado.
- Si fuera la tarea t_{43}

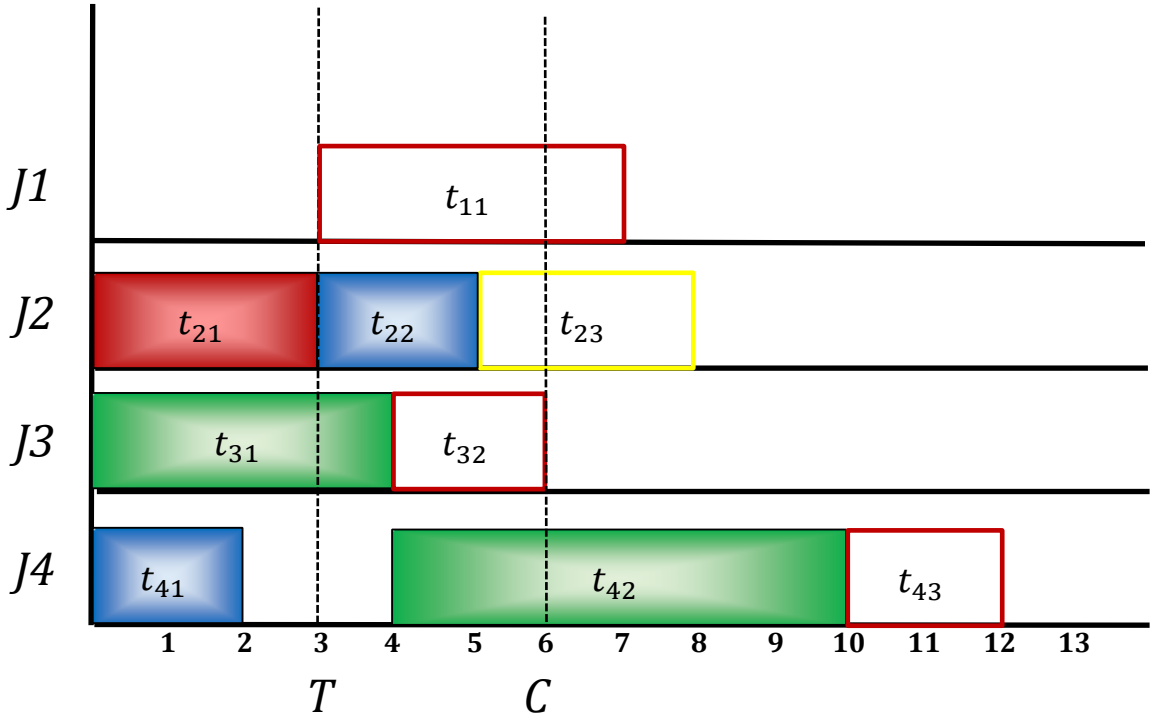
SGS activo por inserción



- $A = \{t_{11}, t_{23}, t_{32}, t_{43}\}$

SGS activo por agregación: Algoritmo G&T

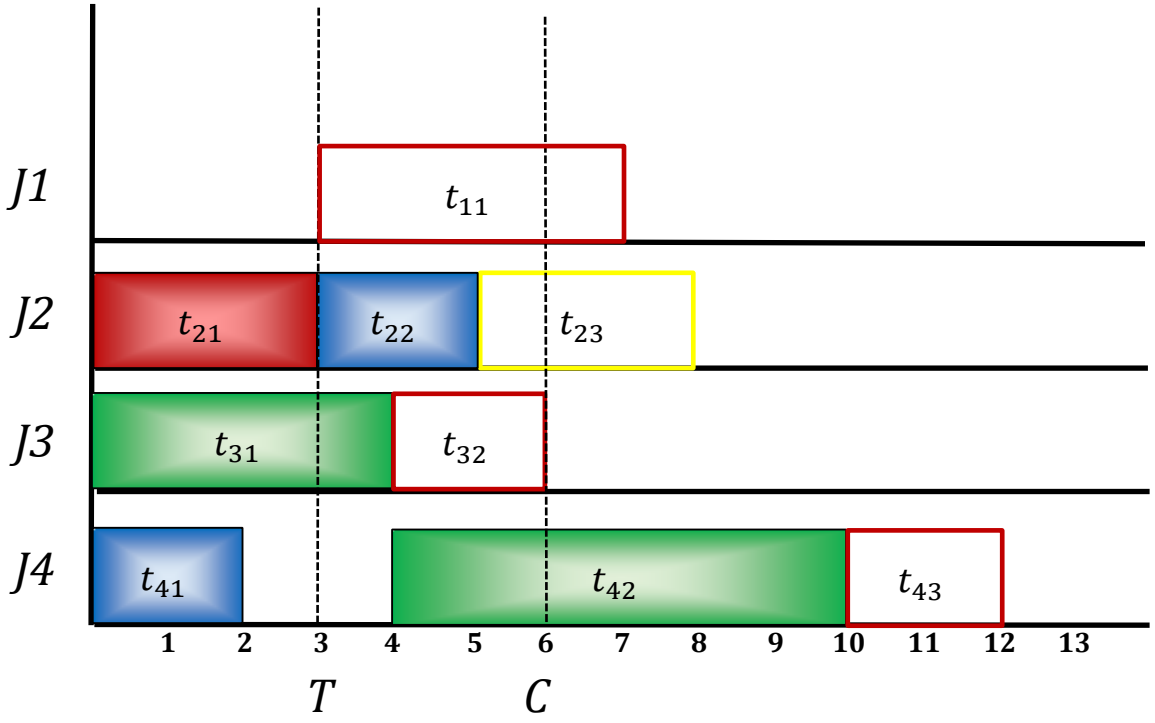
[Giffler&Thomson, 1960]



1. Determinar el conjunto A
   - Primeras tareas sin planificar de cada trabajo
2. Determinar C
   - Mínimo tiempo de fin de las tareas disponibles
3. Seleccionar la máquina M
   - Máquina requerida por la tarea que determinó C
4. Determinar el conjunto de tareas críticas B
   - Las que requieren M y pueden comenzar antes que C
5. Seleccionar de B la siguiente tarea a planificar
   - Elegir de B siguiente tarea a ser planificada

SGS activo por agregación: Algoritmo G&T

[Giffler&Thomson, 1960]



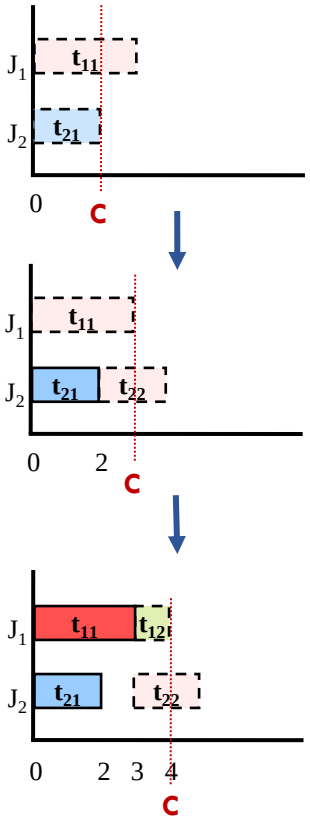
1. Determinar el conjunto A
   - $A = \{t_{11}, t_{23}, t_{32}, t_{43}\}$
2. Determinar C
   - $\mathrm{C} = 6$, el de t_{32}
3. Seleccionar la máquina M
   - $\mathrm{M} = roja$
4. Determinar el conjunto de tareas críticas B
   - $B = \{t_{11}, t_{32}\}$
5. Seleccionar de B la siguiente tarea a planificar
   - t_{11} ó t_{32}

Constructor de schedules Activos/Densos

t_{11}, t_{32}: activos

t_{11}: densos

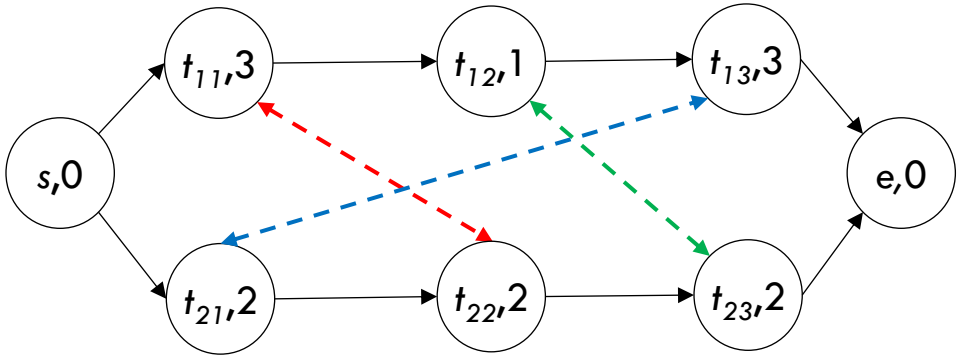
Ejemplo. Algoritmo G&T



A = {t_{11} , t_{21}}
M azul
B = {t_{21}}

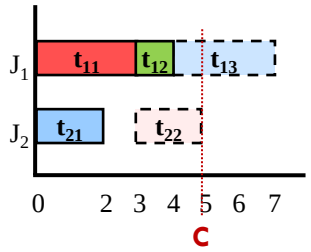
A = {t_{11} , t_{22}}
M roja
B = {t_{11} , t_{22}}
Se puede elegir la t_{11}ó $la\ t_{21}$
En π la t_{11} va antes que la t_{21}
Planificar la t_{11}

Planificación Determinista

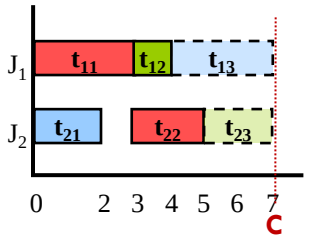


$$\pi = (s, t_{11}, t_{21}, t_{22}, t_{23}, t_{12}, t_{13}, e)$$

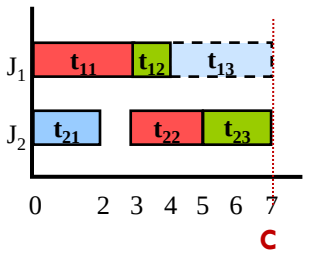
La t_{12}



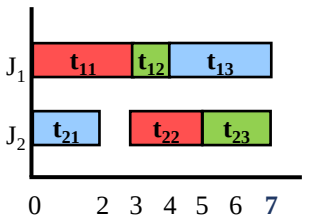
La t_{22}



La t_{23}



La t_{213}



Solución Óptima

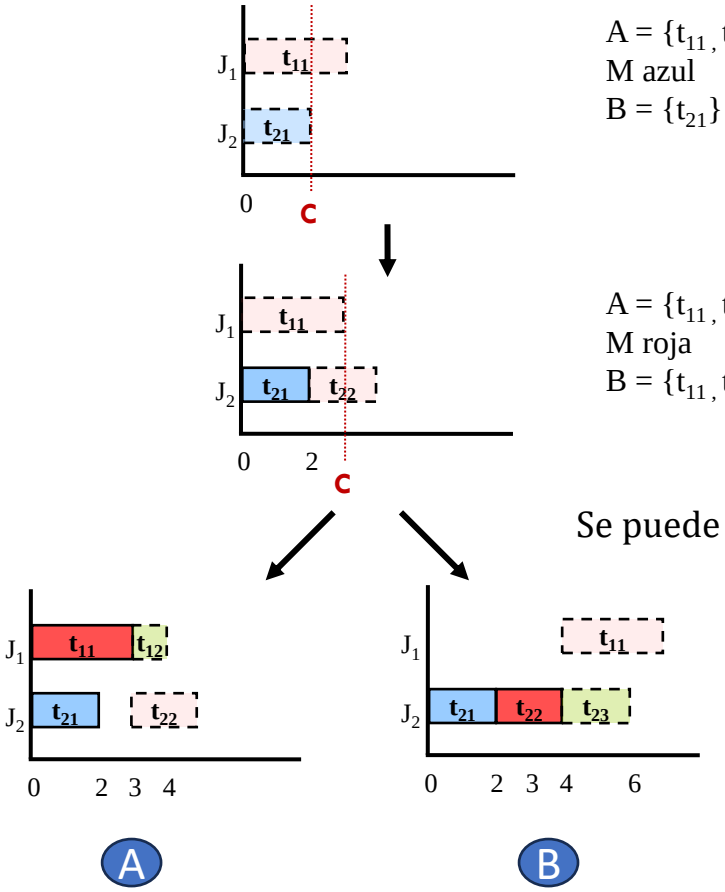
A = {t_{12} , t_{22}}
M verde
B = {t_{12}}

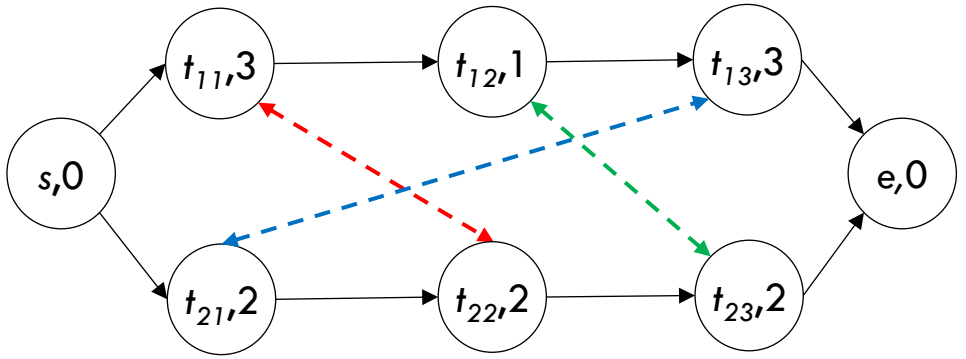
A = {t_{13} , t_{22}}
M roja
B = {t_{22}}

A = {t_{13} , t_{23}}
Ambas nos darían C, elegimos una. Por π la t_{23}
B = {t_{23}}

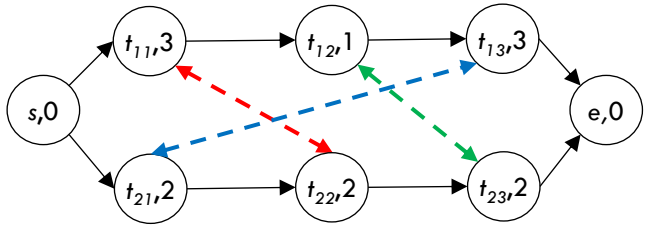
A = {t_{13}}
M azul
B = {t_{13}}

Ejemplo. Algoritmo G&T
Espacio de búsqueda completo

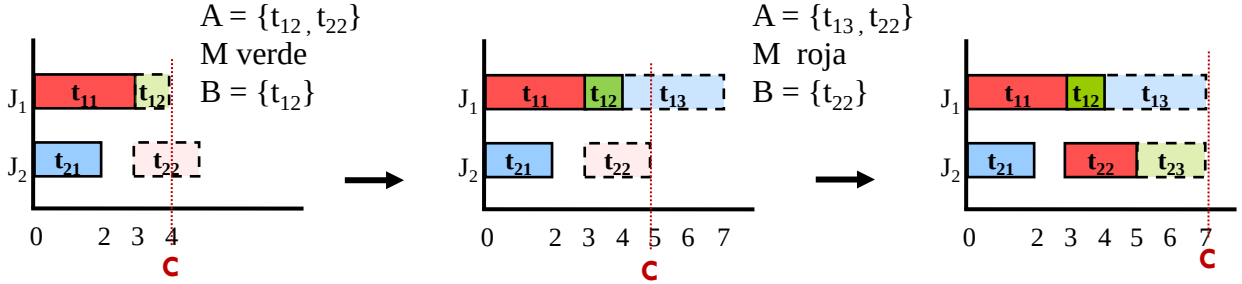




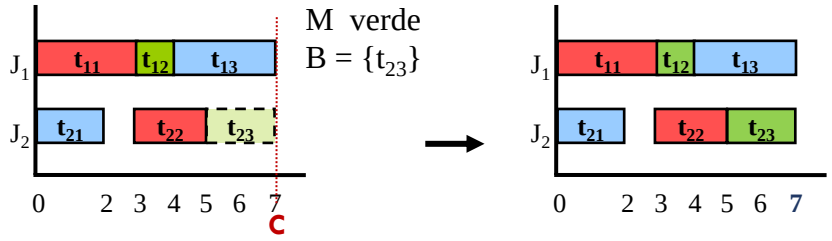
Ejemplo. Algoritmo G&T
Espacio de búsqueda completo



A **Si se elige la t_{11}**



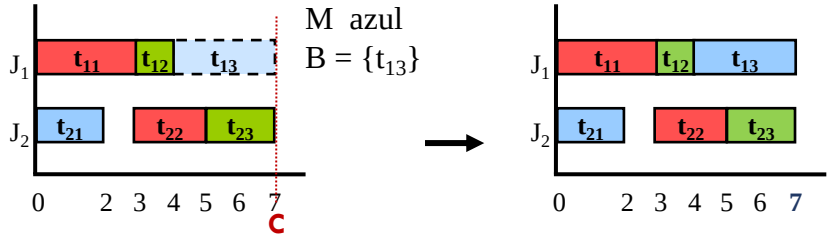
Elegimos la t_{13}
M azul
B = {t_{13}}



Solución Óptima

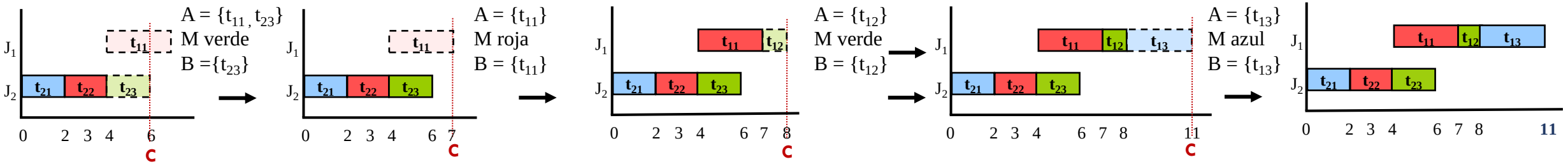
A = {t_{13} , t_{23}}
Ambas nos darían
C, elegimos una

Elegimos la t_{23}
M verde
B = {t_{23}}



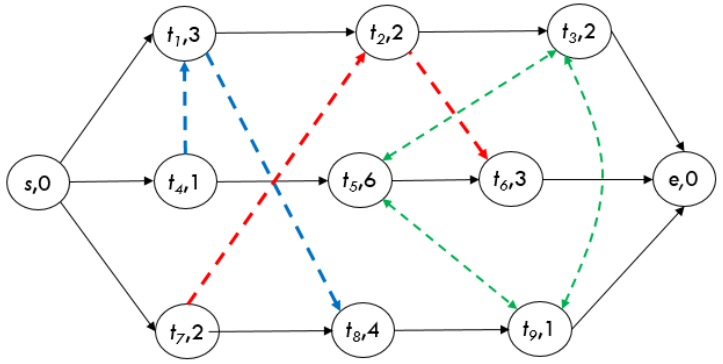
Solución Óptima

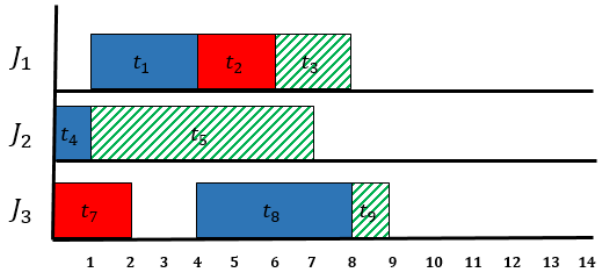
B **Si se elige la t_{22}**



Practiquemos un poco!!!

- Dada la siguiente planificación parcial (tareas rayadas primeras sin planificar de cada trabajo) del problema representado en el grafo de restricciones:



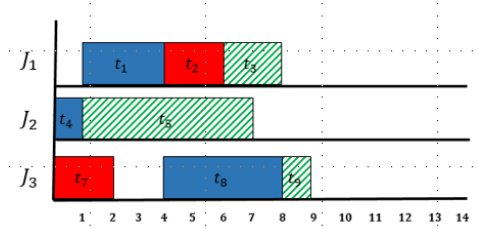


- Emplea tres planificadores, uno semi-activo, otro activo por inserción y otro activo con G&T, en el que se emplee la regla SPT para elegir la siguiente tarea a planificar.

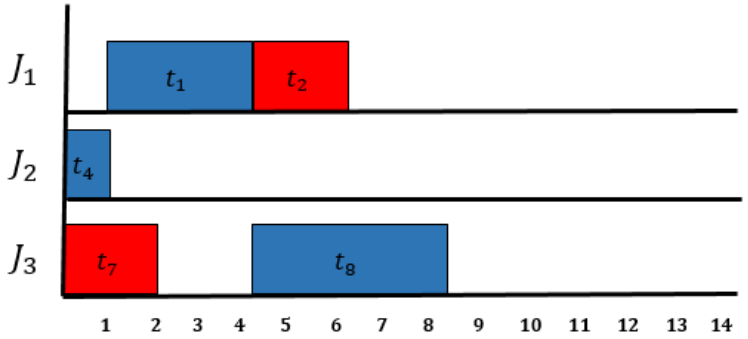
Semi-activo	Activo por Inserción	Activo G&T
Tareas candidatas: Tarea a planificar:	Tareas candidatas: Tarea a planificar:	Tareas candidatas: Tarea a planificar:

Practiquemos un poco!!!

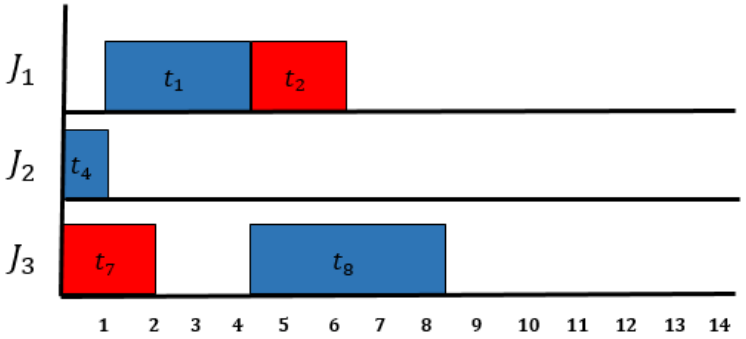
- Dibuja en todos los casos cuál sería la planificación siguiente, incluyendo en el dibujo las siguientes tareas sin planificar de cada trabajo.



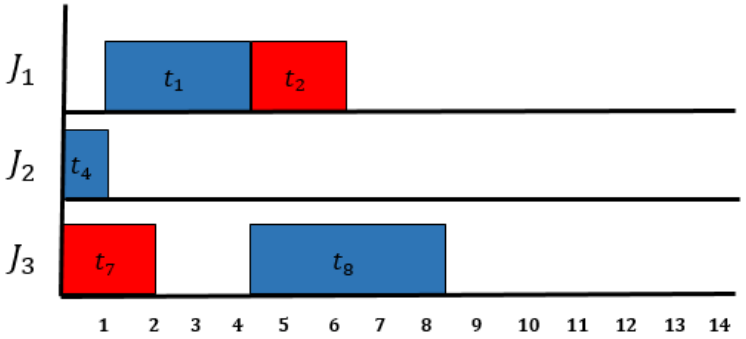
Semi-activo



Activo por Inserción



Activo G&T



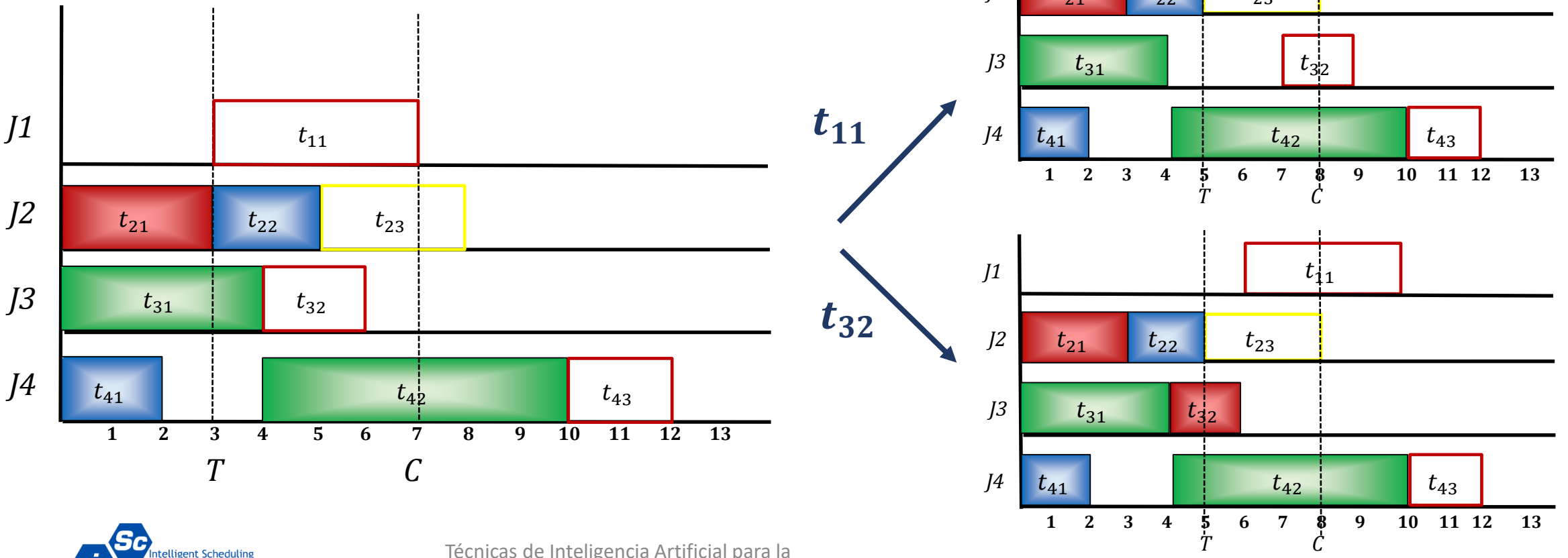
- En este caso, si la función objetivo es el makespan, ¿cuál es una planificación mejor y por qué?

Utilidad de los Algoritmos SGS

- **Búsqueda heurística:** permite generar el espacio de búsqueda de las planificaciones (activas, si se emplea el G&T).
  - Dado un estado intermedio, una planificación parcial, en la que aún no se han planificado todas las tareas, permite generar los sucesores (nuevas planificaciones parciales posibles) que garantizan poder alcanzar una solución óptima.
- **Cálculo de cotas superiores:** como algoritmo voraz, que en cada paso elija (con algún criterio), de entre las posibles, la siguiente tarea a planificar.
  - Criterio: Aleatorio, Heurístico (Reglas de Prioridad)
- **Metaheurísticas:** como algoritmo de decodificación de los cromosomas y de generación de poblaciones iniciales heurísticas.
  - En cada paso del algoritmo SGS, de entre el conjunto de tareas candidatas a ser la siguiente a planificar, elegiría aquella que aparece antes en el cromosoma.
- **Programación Genética:** como algoritmo para evaluar lo buenas que son las reglas de prioridad, calculadas automáticamente, a la hora de construir soluciones.

Utilidad de los Algoritmos SGS

- **Algoritmos exactos: Búsqueda heurística**
  - *Espacio de búsqueda de schedules activos*

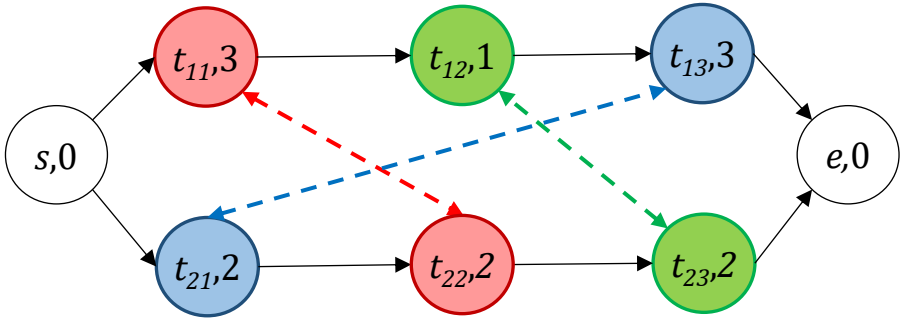


Utilidad de los Algoritmos SGS

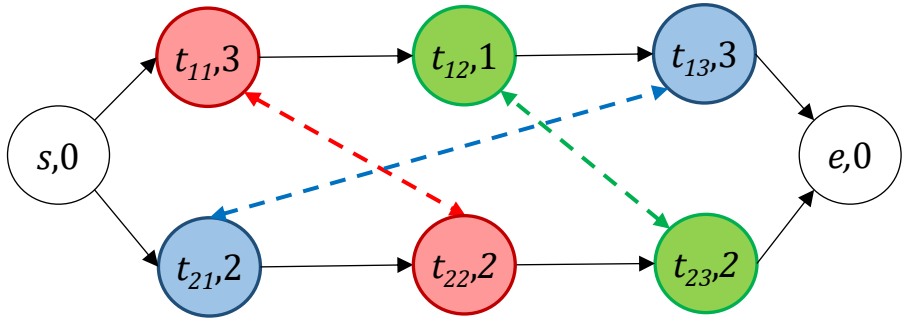
- **Metaheurísticas:**
  - *Decodificación Cromosomas*
    - *Directa: planificación semi-activa o activa por inserción*
    - *Indirecta con G&T: planificación activa*

Cromosoma

$$(t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, t_{23})$$



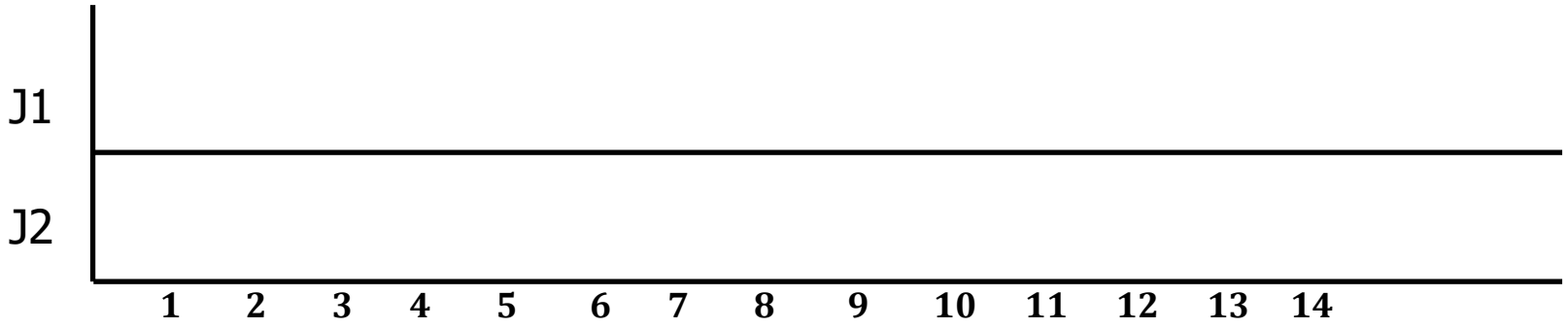
Ejemplo Decodificación cromosoma directa



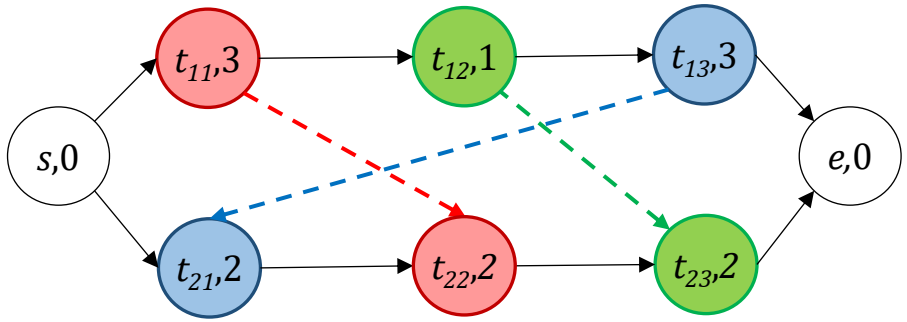
Vamos planificando las tareas en el orden expresado por el cromosoma

Cromosoma

$$(t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, t_{23})$$



Ejemplo Decodificación cromosoma directa



Cromosoma

$$(t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, t_{23})$$

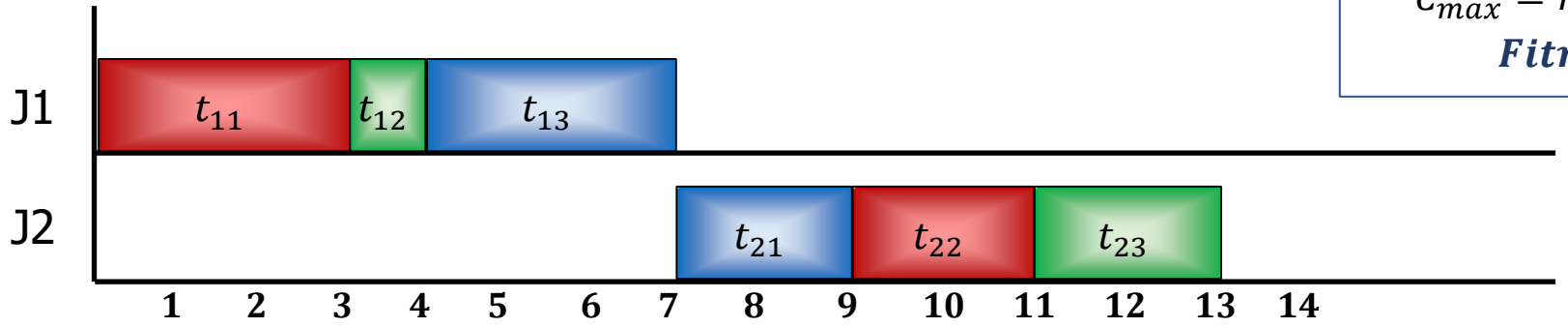
Fenotipo: Solución representada por el cromosoma

Makespan:

$$C_{max} = max\{7, 13\} = 13$$

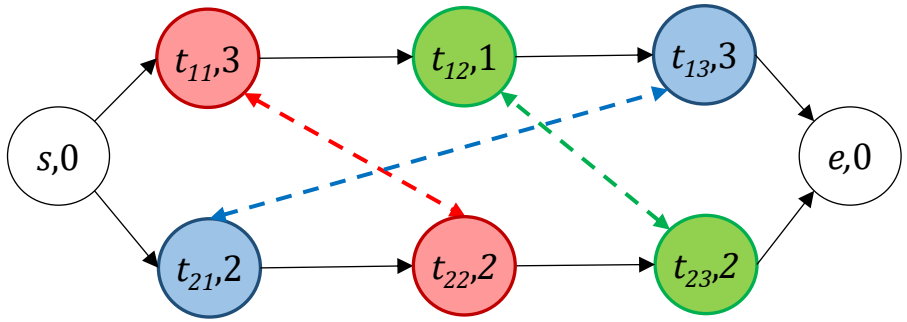
$$\boldsymbol{Fitness} = 13$$

Planificación semi-activa



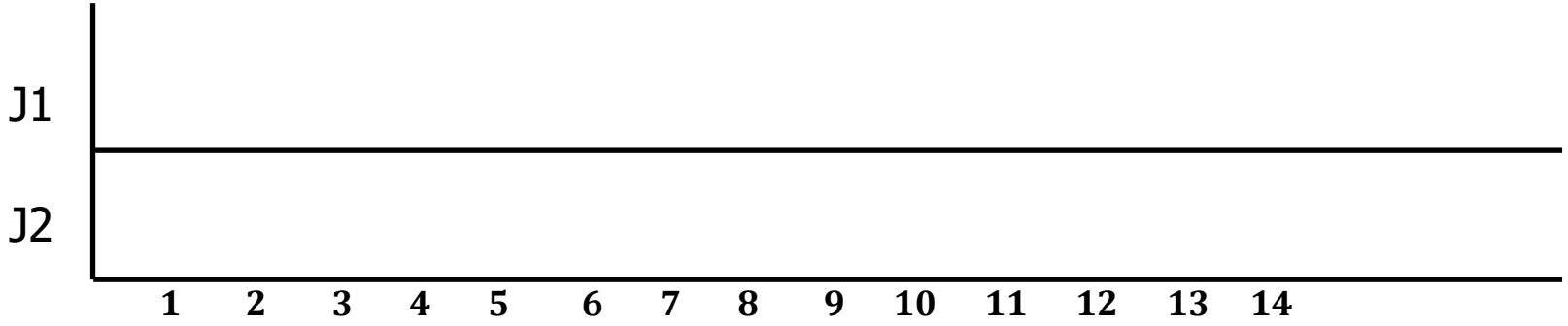
Ejemplo: Decodificación activa con G&T

En cada paso vamos a ir calculando el conjunto A (tareas candidatas a ser planificadas) y, a partir de este, el subconjunto de tareas B (que garantiza que la planificación sea activa).
Para ello calculamos C (menor tiempo de fin de las tareas en A), y construimos B con aquellas tareas que comparten máquina con la tarea que estableció C y pueden comenzar antes que C.

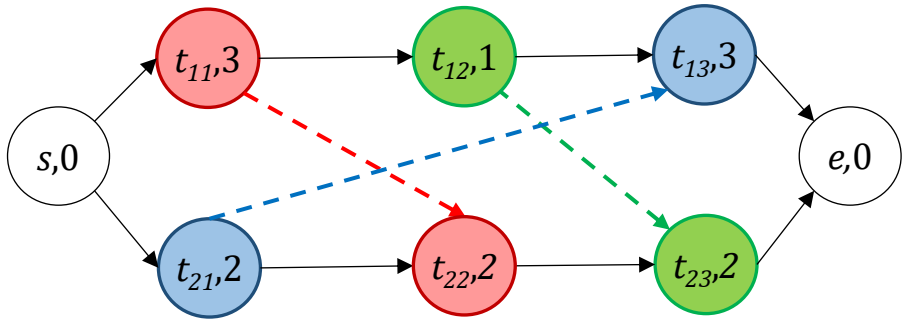


Cromosoma

$$(t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, t_{23})$$



Ejemplo: Decodificación activa con G&T



Cromosoma

$$(t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, t_{23})$$

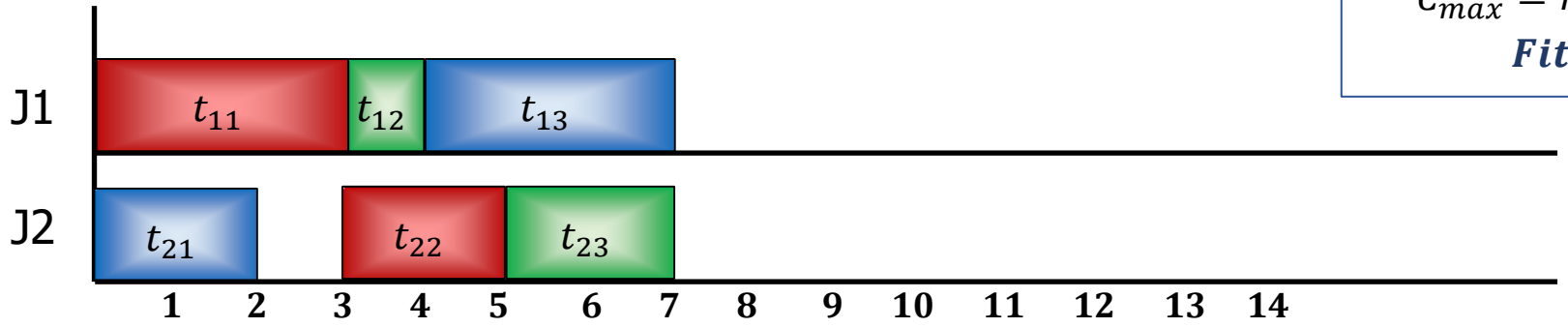
Fenotipo: Solución representada por el cromosoma

Makespan:

$$C_{max} = max\{7, 7\} = 7$$

$$\boldsymbol{Fitness} = 7$$

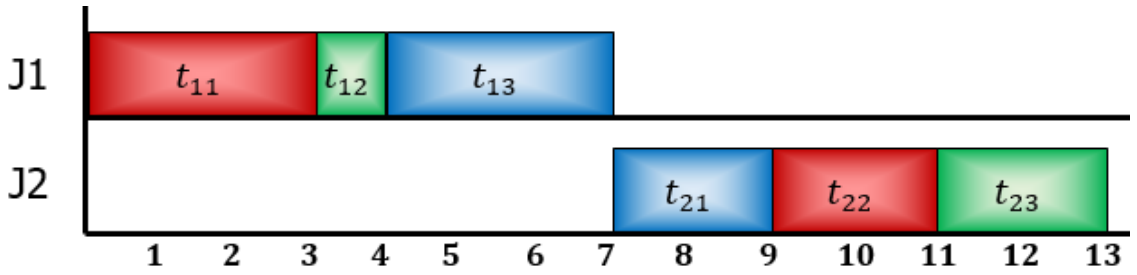
Planificación activa
Densa
Óptima



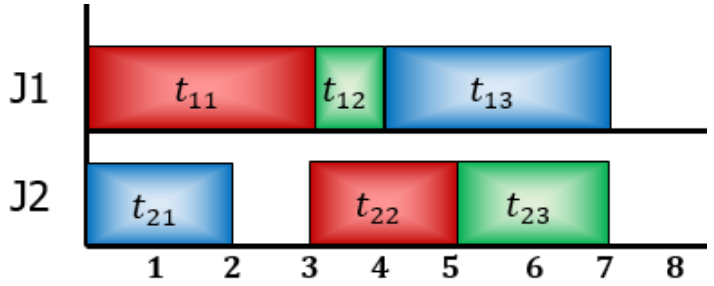
Decodificación directa/semi-áctiva vs G&T

Cromosoma $(t_{11}, t_{12}, t_{13}, t_{21}, t_{22}, t_{23})$

Fenotipo obtenido con las decodificaciones directa y semi-activa



Fenotipo obtenido con la decodificación activa con G&T



- Como se observa, el fitness calculado para el cromosoma es mejor con la decodificación activa con G&T ($\boldsymbol{fitness = 7}$) que con la codificación directa/semi-activa ($\boldsymbol{fitness = 13}$).

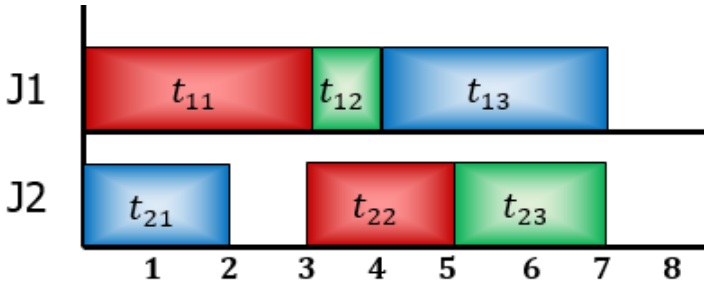
Decodificación directa/semi-áctiva vs G&T

- **¿Siempre es peor una decodificación directa/semi-activa que activa?**
  - No tiene por qué ser así. La decodificación semi-activa puede encontrar soluciones tan buenas como las activas, sólo depende del orden expresado en el cromosoma.
  - Por ejemplo, si el cromosoma expresa el orden de la planificación obtenida con el decodificador con G&T, los decodificadores directo y semi-activo obtiene la misma solución que el decodificador que emplea el G&T.

Cromosomas que obtienen la misma decodificación directa/semi-activa que con G&T

$(t_{11}, t_{21}, t_{12}, t_{22}, t_{13}, t_{23})$
$(t_{11}, t_{21}, t_{22}, t_{12}, t_{13}, t_{23})$
$(t_{11}, t_{21}, t_{12}, t_{22}, t_{23}, t_{13})$
$(t_{21}, t_{11}, t_{12}, t_{22}, t_{13}, t_{23})$

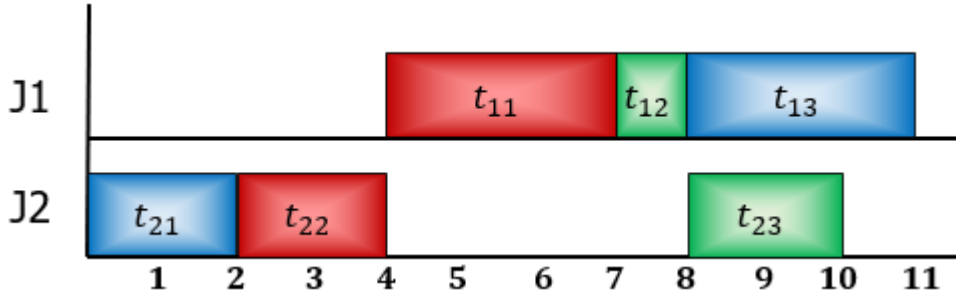
....



Decodificación directa/semi-áctiva vs G&T

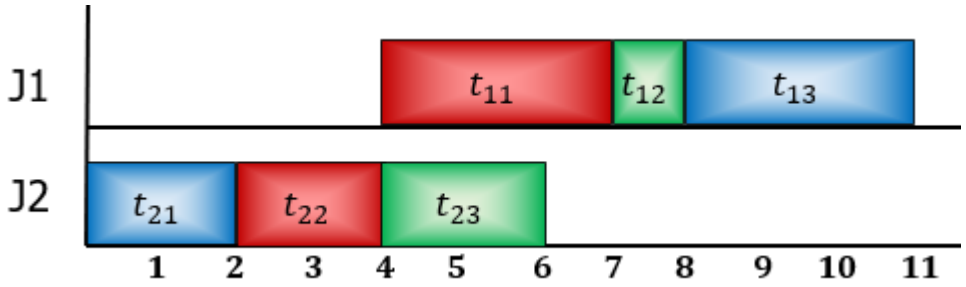
- **Otro ejemplo:**
  - Si tenemos el cromosoma $(t_{21}, t_{22}, t_{11}, t_{12}, t_{13}, t_{23})$
  - El fitness del cromosoma calculado con decodificación directa/semi-activa y con el decodificador G&T es el mismo, aunque la planificación construida en un caso sea semi-activa y en el otro activa.

Decodificación directa/semi-activa



Planificación semi-activa, $fitness = 11$

Decodificación activa con G&T

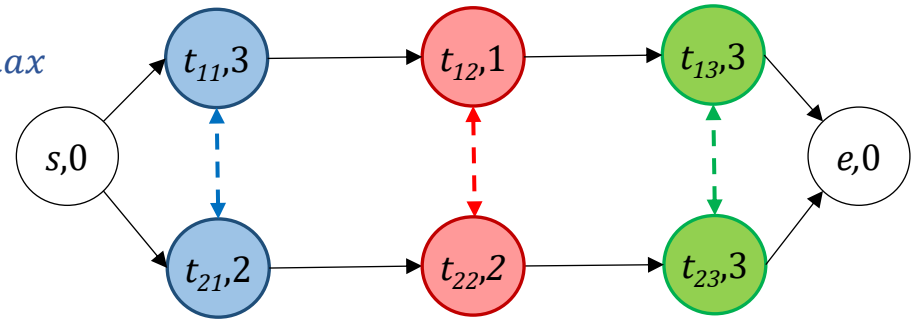


Planificación activa, no densa , $fitness = 11$

Influencia del cromosoma en el decodificador activo con G&T

- Si el **algoritmo G&T** tiene más de una tarea en el conjunto B (tareas candidatas a ser planificadas) la elección de la tarea a planificar se puede realizar mediante funciones de prioridad (Random, etc…) o empleando un orden (topológico) de procesamiento de las tareas.
  - Cuando el G&T se emplea como decodificador en un Algoritmo Genético, utilizará para tomar esta decisión el orden de las tareas expresado en el cromosoma.
  - Este orden incide de manera directa en la planificación (fenotipo) que se obtiene a partir de él y en el coste de la función objetivo para dicha planificación (fitness del cromosoma)
- Veamos con un ejemplo cómo la planificación construida por el decodificador G&T y su coste depende del cromosoma a decodificar.

Problema $J||C_{max}$

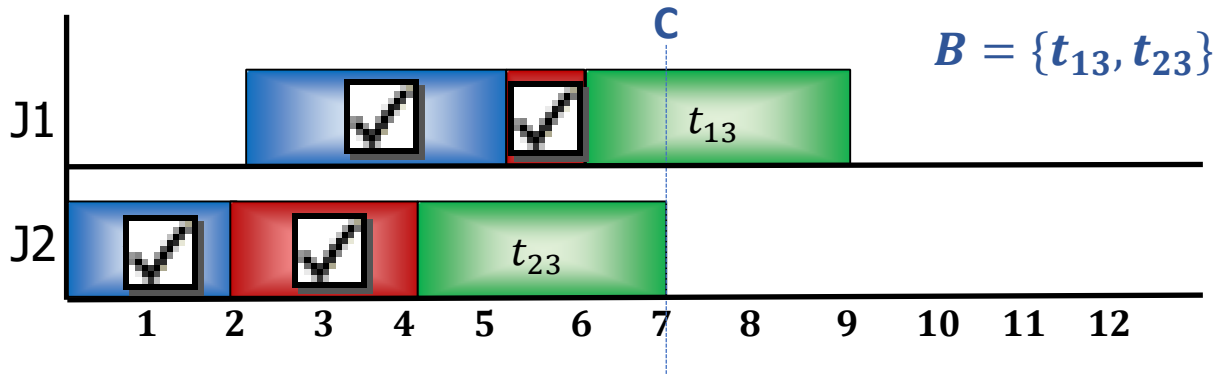


Cromosomas a decodificar

$(t_{21}, t_{11}, t_{22}, t_{12}, t_{23}, t_{13})$
$(t_{21}, t_{11}, t_{22}, t_{12}, t_{13}, t_{23})$

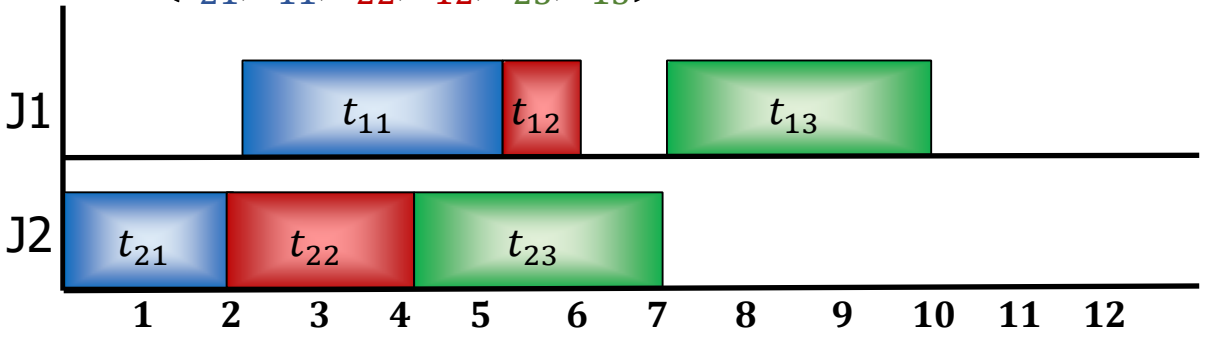
Influencia del cromosoma en el decodificador activo con G&T

Los dos cromosomas planifican en el mismo orden todas las tareas excepto las de la máquina verde. Por lo tanto, la decodificación de ambos llegará a esta misma solución parcial, en la que $B = \{t_{13}, t_{23}\}$. Sin embargo, al planificar en orden distinto las tareas de la máquina verde, la calidad de la planificación obtenida difiere.



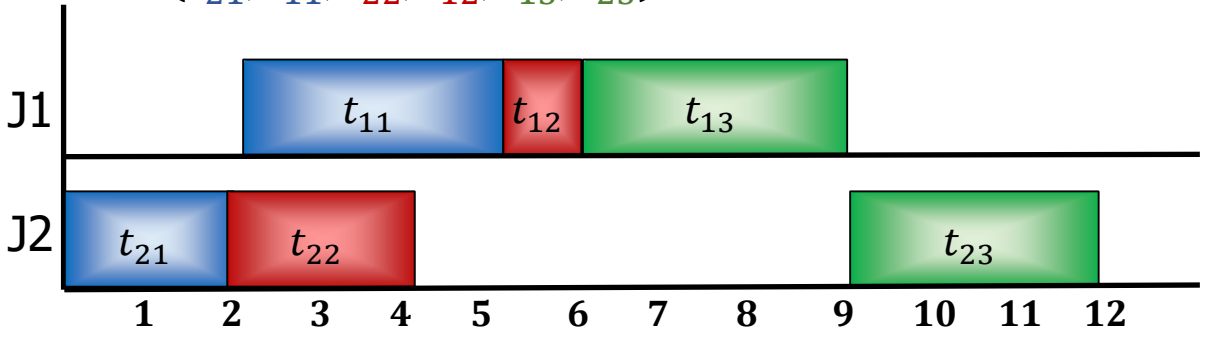
Ambas son planificaciones activas, pero una deja menos huecos lo que hace que sea una solución mejor (con menor coste) y por tanto menor fitness.

$(t_{21}, t_{11}, t_{22}, t_{12}, t_{23}, t_{13})$ → fitness = 10



Planificación activa, densa

$(t_{21}, t_{11}, t_{22}, t_{12}, t_{13}, t_{23})$ → fitness = 12



Planificación activa, no densa

Contenidos

Problema de Scheduling

1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - **Reglas de Prioridad**
   - Extensiones del JSSP
5. Referencias Bibliográficas

Reglas de Prioridad

- Dos componentes: un **SGS** y una **Función de prioridad**
- Algoritmo voraz en el que, el orden de ejecución de las tareas lo establece una función de prioridad que es la encargada de elegir en cada paso, de entre las candidatas, la siguiente tarea a planificar (la más prioritaria según la función de prioridad).
  - En caso de tareas con la misma prioridad se elige aleatoriamente.
- En lugar de pasar al SGS un orden de procesamiento se le pasaría una función de prioridad.
- Como SGS:
  - El que genera planificaciones semi-activas
  - El que genera planificaciones activas: por inserción o por agregación (G&T).
  - …

Reglas de Prioridad. SGS

Algoritmo para Regla de Prioridad (SGS modificado)

Recibe: una instancia de JSSP, ***P***, y **una función de prioridad f que devuelve la prioridad de la tarea que se le pasa como argumento**

Devuelve: un Schedule ***st*** para ***P*** de acuerdo con

1. $A = \{o_{j\mathbf{1}} : 1 \leq j \leq n\}$ # Primeras tareas sin planificar de cada trabajo
2. Mientras $A \neq \emptyset$ hacer
   a. Calcular el conjunto elegible $B \subseteq A$
   b. Seleccionar $o_{j^*k^*} = \boldsymbol{argmax}\{\boldsymbol{f}(o_{jk}) : o_{jk} \in \boldsymbol{B}\}$ **# Tarea más prioritaria**
   c. $st_{j^*k^*} = ES_{j^*k^*}$ # Mínimo tiempo de iniciò para esa tarea
   d. $A = A - \{o_{j^*k^*}\} \cup \{o_{j^*(k+1)^*} \; si \; k^* < m_{j^*}\}$ # Añadimos la siguiente tarea en el trabajo
3. Devuelve st

Reglas de Prioridad. JSSP

- **Random (RND):** selecciona una tarea de forma aleatoria.
- **Shortest Processing Time (SPT):** selecciona la tarea con menor tiempo de procesamiento.
- **Longest Processing Time (LPT):** selecciona la tarea con mayor tiempo de procesamiento.
- **Most Work remaining (MWR):** selecciona la tarea del trabajo con el mayor tiempo de procesamiento pendiente.
- **Least Work remaining (LWR):** selecciona la tarea del trabajo con el menor tiempo de procesamiento pendiente.
- **Most Operations Remaining (MOR):** selecciona la tarea del trabajo con el mayor número de operaciones pendientes.
- **Least Operations Remaining (LOR):** selecciona la tarea del trabajo con el menor número de operaciones pendientes.
- **Weighted Shortest Processing Time (WSPT):** selección la tarea con el tiempo de procesamiento ponderado más corto.
- **Earliest Due Date (EDD):** selecciona la tarea con la fecha de vencimiento (entrega) más temprana.
- **Minimum Slack (MS):** selecciona la tarea con mínima holgura.
- **First Come, First Served (FCFS):** selecciona la primera tarea en la cola de los trabajos para la misma máquina.
- …..

Reglas de Prioridad. JSSP

- También podemos tener reglas más complejas como resultado de combinar otras reglas, es el caso de la regla Apparent Tardiness Cost (ATC), una de las reglas más eficientes para resolver el JSSP para funciones objetivo que involucren a los tiempos de vencimiento (due dates, d_j). Determina la prioridad de una tarea de la forma:
  - $\pi_{ij} = \frac{w_j}{p_{ij}} \exp\left(-\frac{\max(d_j - p_{ij} - \gamma(\alpha), 0)}{g\bar{p}}\right)$

  donde:
    - $\gamma(\alpha)$ primer instante con capacidad disponible para planificar, al menos, una tarea
    - $\bar{p}$ tiempo de procesamiento medio de todas las tareas sin planificar hasta ese momento
    - g parámetro de escala introducido por el usuario.
- Combinación de dos reglas básicas:
  - Weighted Shortest Processing Time (WSPT). Elige la tarea con el tiempo de procesamiento ponderado más corto: $\pi_{ij} = \frac{w_j}{p_{ij}}$
  - Minimum Slack (MS). Elige la tarea con mínima holgura: $\pi_{ij} = (d_j - r_j) - p_{ij}$

Reglas de Prioridad. Ejemplo $1||C_{max}$

- Dado siguiente problema $1||C_{max}$, veamos cómo se aplicarían las reglas de prioridad SPT y EDD para construir una solución al problema.
  - Suponemos que todos los trabajos llegan en el intante 0
- Teniendo en cuenta el d_i calcularemos también:
  - **Lateness (retraso):** $L_i = C_i - d_i$. $L_i \leq 0$ ó $L_i \geq 0$.
  - **Tardiness (tardanza):** $T_i = max(C_i - d_i, 0) = max(L_i, 0)$. $T_i \geq 0$.
  - **Earliness (puntualidad):** $E_i = max(0,\ d_i - C_i)$. $E_i \geq 0$.
- Calcularemos los valores de las funciones objetivo:
  - **Makespan:** $C_{max} = max\{C_i |\ i = 1, \dots, n\}$
  - **Total Completion Time:** $\sum C_i = \sum_{i=1}^{n} C_i$
  - **Total Tardiness:** $\sum T_i = \sum_{i=1}^{n} T_i$
  - **Maximum Lateness:** $L_{max} = max\{L_i |\ i = 1, \dots, n\}$
  - **Total number of late jobs:** $\sum U_i = \sum_{i=1}^{n} U_i$

Job	p_j	d_j
J_1	2	7
J_2	8	16
J_3	4	4
J_4	10	17
J_5	5	15
J_6	12	18

Reglas de Prioridad. Ejemplo $1||C_{max}$

Job	p_j	d_j
J_1	2	7
J_2	8	16
J_3	4	4
J_4	10	17
J_5	5	15
J_6	12	18

SPT

Job	Schedule		Lateness	
	Inicio	Fin	Earliness $max(0, d_i - C_i)$	Tardiness $max(C_i - d_i, 0)$
J_1	0	2	5	0
J_3	2	6	0	2
J_5	6	11	4	0
J_2	11	19	0	3
J_4	19	29	0	12
J_6	29	41	0	23

Makespan: $C_{max} = \max\{2, 6, 11, 19, 29, 41\} = 41$

Total Completion Time: $\sum C_i = 2 + 6 + 11 + 19 + 29 + 41 = 108$

Total Tardiness: $\sum T_i = 0 + 2 + 0 + 3 + 12 + 23 = 40$

Maximum Lateness: $L_{max} = \max\{Ti\} = 23$

Total number of late jobs: $\sum U_i = 4$

Reglas de Prioridad. Ejemplo $1||C_{max}$

Job	p_j	d_j
J_1	2	7
J_2	8	16
J_3	4	4
J_4	10	17
J_5	5	15
J_6	12	18

EDD

Job	Schedule		Lateness	
	Inicio	Fin	Earliness $max(0, d_i - C_i)$	Tardiness $max(C_i - d_i, 0)$
J_3	0	4	0	0
J_1	4	6	1	0
J_5	6	11	4	0
J_2	11	19	0	3
J_4	19	29	0	12
J_6	29	41	0	23

Makespan: $C_{max} = \max\{4, 6, 11, 19, 29, 41\} = 41$

Total Completion Time: $\sum C_i = 4 + 6 + 11 + 19 + 29 + 41 = 110$

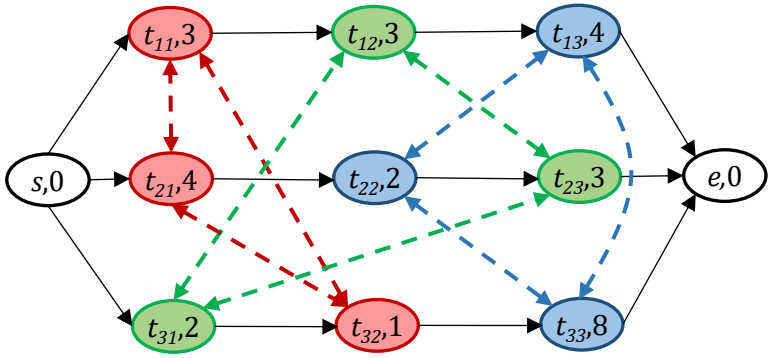
Total Tardiness: $\sum T_i = 0 + 0 + 0 + 3 + 12 + 23 = \mathbf{38}$

Maximum Lateness: $L_{max} = \max\{Ti\} = 23$

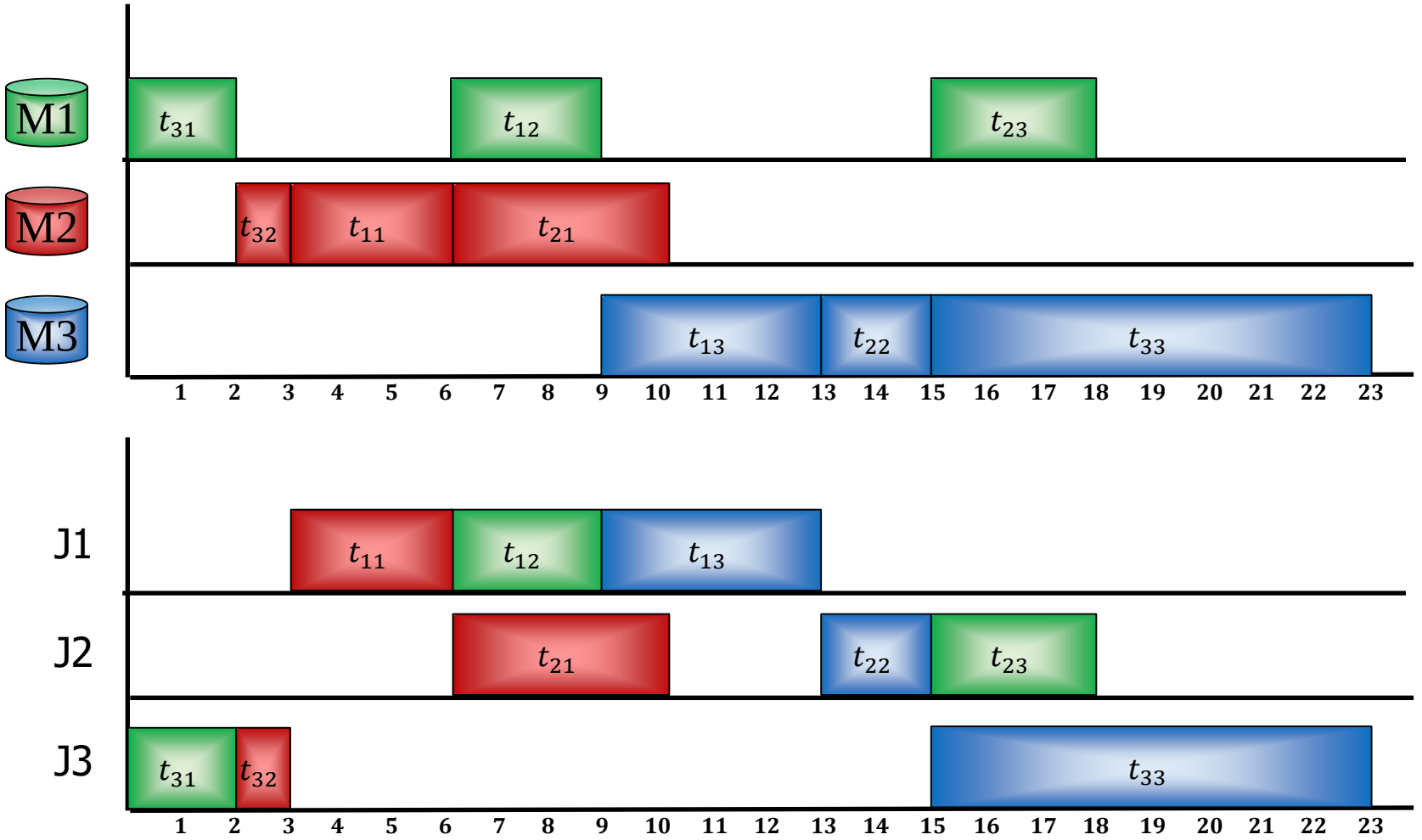
Total number of late jobs: $\sum U_i = 4$

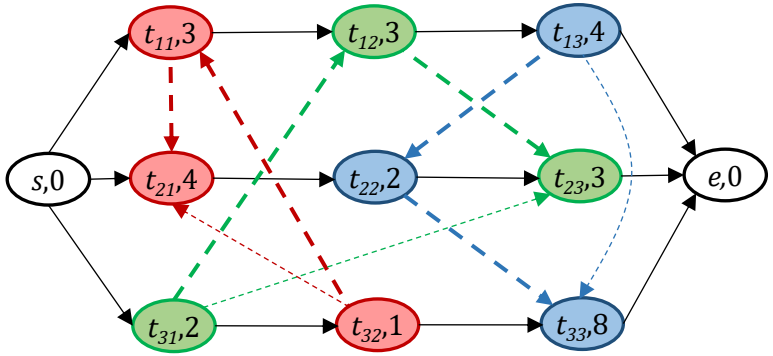
Reglas de Prioridad. Ejemplo $J||C_{max}$

- Dado siguiente problema $J||C_{max}$, veamos cómo se aplicaría la regla de prioridad **Shortest Processing Time (SPT)** para construir una solución al problema.
  - Suponemos que todos los trabajos llegan en el intante 0
- Emplear un **Planificador Básico: SGS semi-activo**
- Calcularemos los valores de las funciones objetivo:
  - **Makespan:** $C_{max} = max\{C_i | \, i = 1, \dots, n\}$
  - **Total Completion Time:** $\sum C_i = \sum_{i=1}^{n} C_i$



Reglas de Prioridad. Ejemplo $J||C_{max}$





Makespan:

$$C_{max} = max\{13, 18, 23\} = 23$$

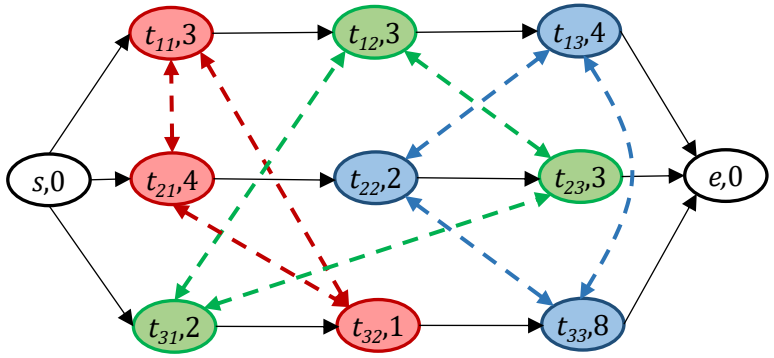
Total Completion Time:

$$\sum C_i = \sum_{i=1}^{n} C_i = 13 + 18 + 24 = 55$$

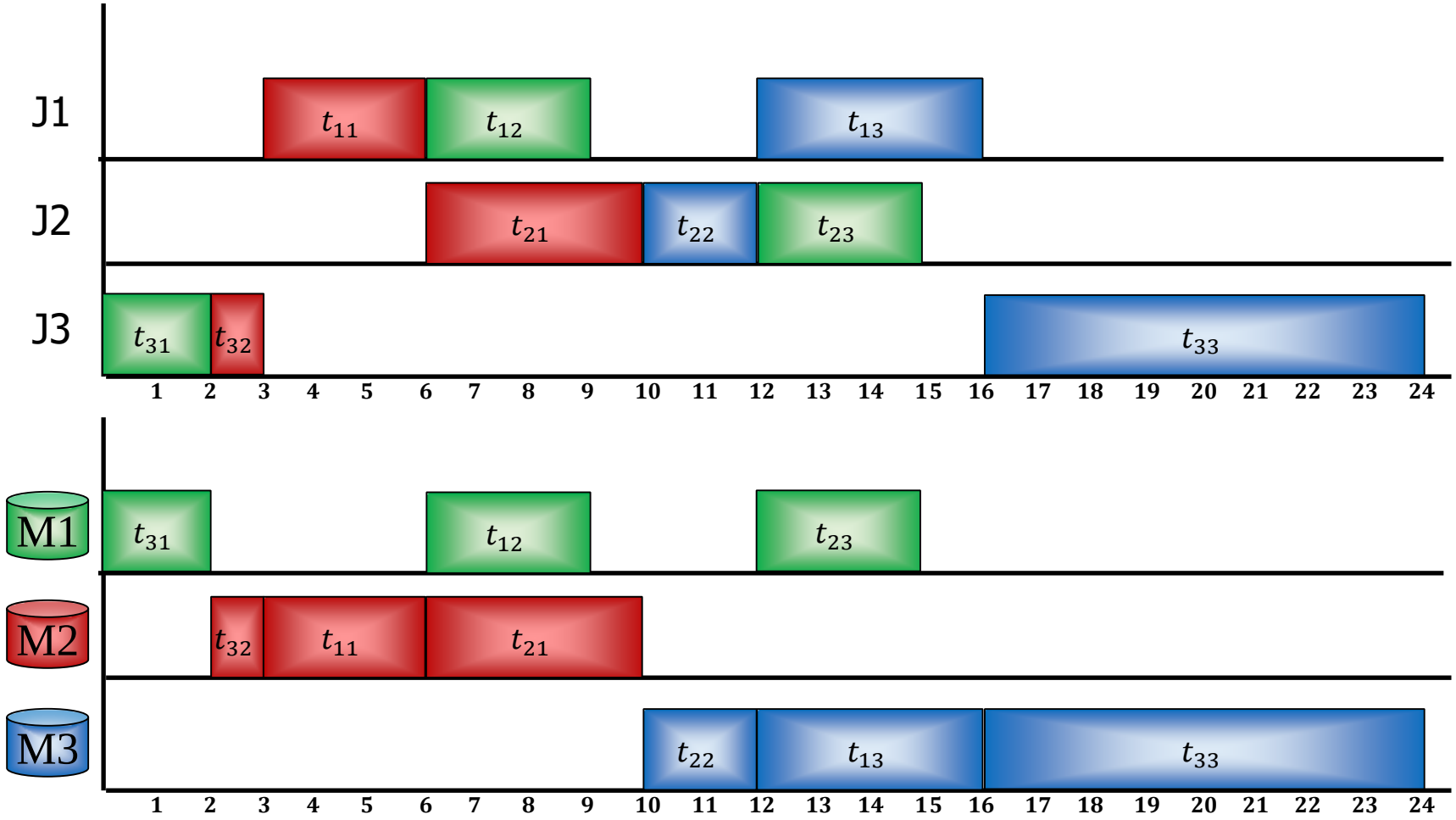
Semi-activa
Activa
No Densa

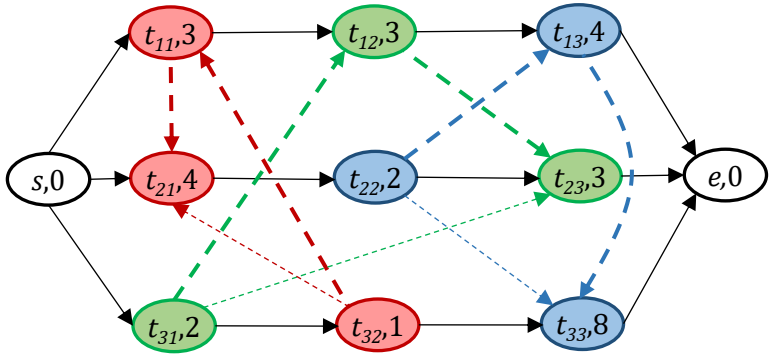
Reglas de Prioridad. Ejemplo $J||C_{max}$

- Dado siguiente problema $J||C_{max}$, veamos cómo se aplicaría la regla de prioridad **Shortest Processing Time (SPT)** para construir una solución al problema.
  - Suponemos que todos los trabajos llegan en el intante 0
- Emplear un **G&T: SGS activo**
- Calcularemos los valores de las funciones objetivo:
  - **Makespan**: $C_{max} = max\{C_i | \, i = 1, \dots, n\}$
  - **Total Completion Time**: $\sum C_i = \sum_{i=1}^{n} C_i$



Reglas de Prioridad. Ejemplo $J||C_{max}$





Makespan:

$$C_{max} = max\{16, 15, 24\} = 24$$

Total Completion Time:

$$\sum C_i = \sum_{i=1}^{n} C_i = 16 + 15 + 24 = 55$$

Semi-activa

Activa

No Densa

Contenidos

Problema de Scheduling

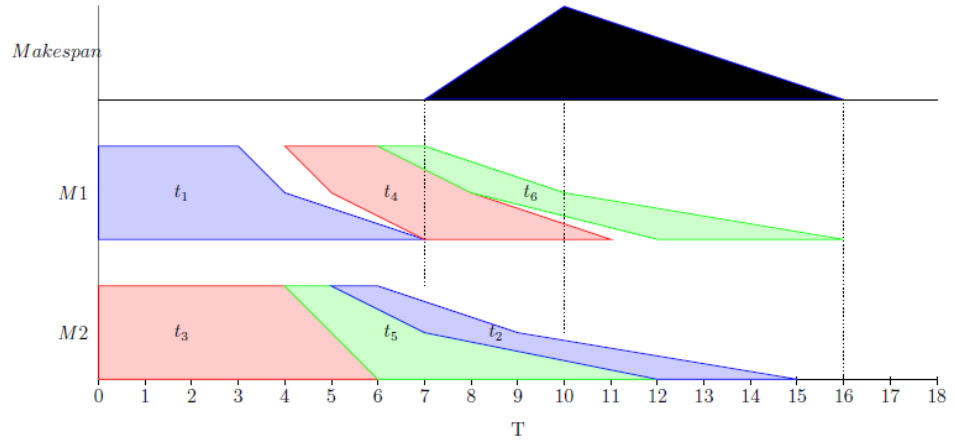
1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - **Extensions del JSSP**
5. Referencias Bibliográficas

Algunas extensiones y variantes del JSSP

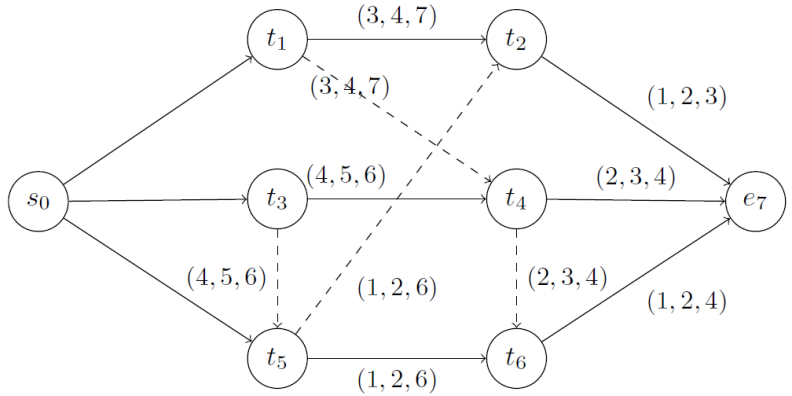
- JSSP con duraciones inciertas
- JSSP con tiempos de ajuste (*setup*)
- JSSP con operarios
- JSSP con restricciones de bloques
- JSSP con almacenes temporales (*buffers*)
- …

JSSP con incertidumbre

- El tiempo de procesamiento de las operaciones es fuzzy (a Triangular Fuzzy Number)

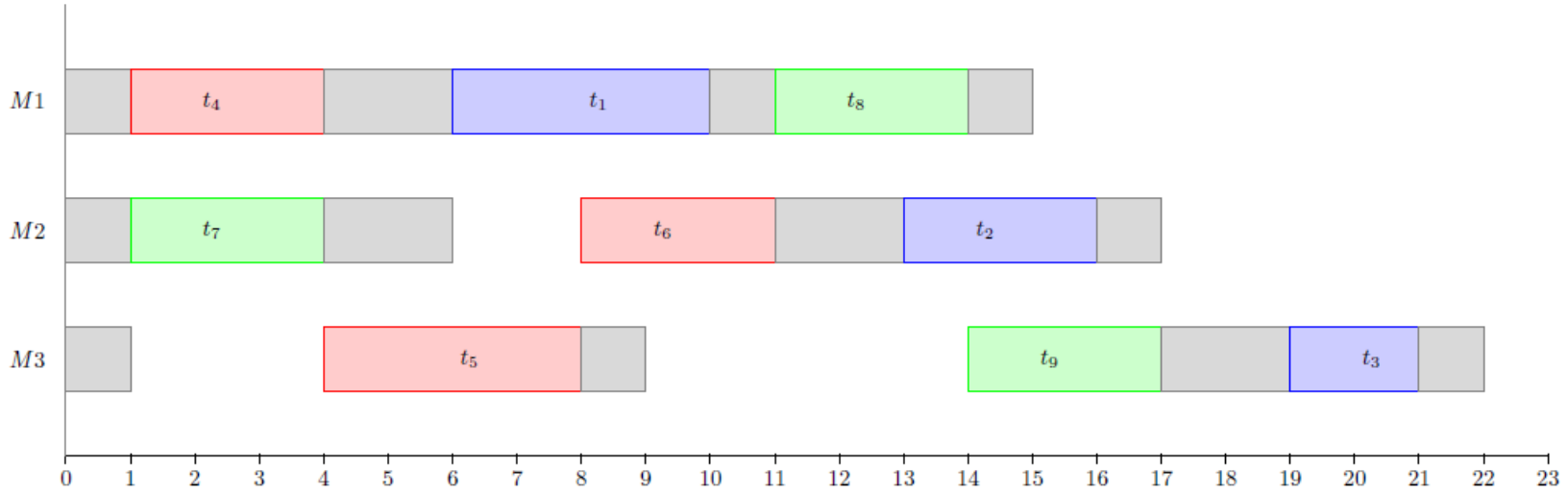


- El orden de procesamiento de las operaciones no tiene ninguna incertidumbre



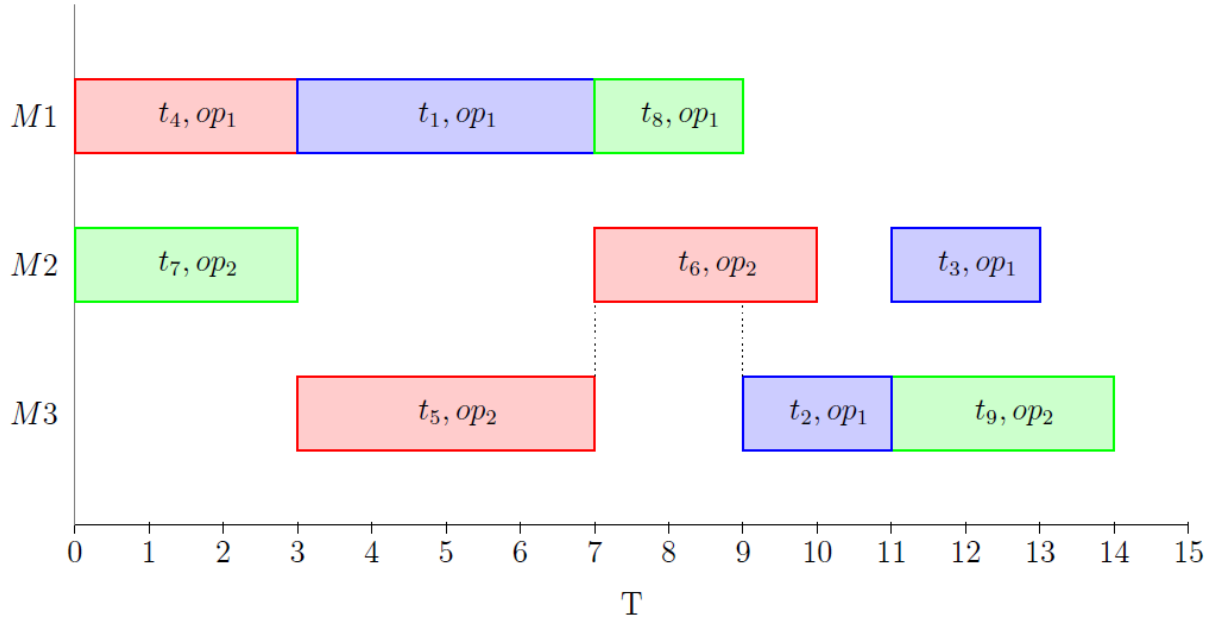
JSSP con tiempos de ajuste $J|s_{ijk}|C_{max}$

- Los tiempos de ajuste dependen tanto de la tarea que sale de la máquina como de la tarea que entra a continuación
- Los ajustes pueden hacerse en paralelo con otras tareas del mismo trabajo, pero no en la misma máquina



JSSP con Operarios JOp||C_{max} *[Agnetis, 2011]* (JSO(n,p))

- Ejemplo:
  - m=3 (máquinas) y p=2 (operarios)
  - No pueden procesarse más de 2 tareas al mismo tiempo



Contenidos

Problema de Scheduling

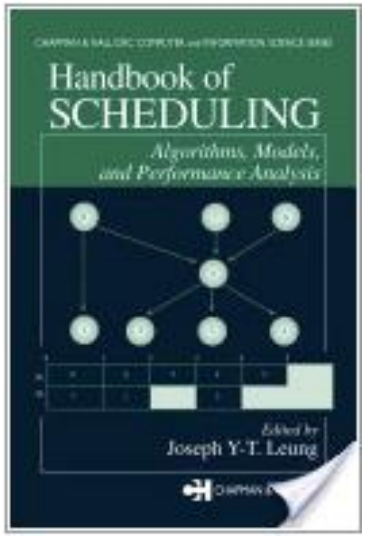
1. Introducción al Scheduling
2. Conceptos Básicos
3. Clasificación Problemas de Scheduling
4. **Problema Job Shop Scheduling Problem**
   - Definición del JSSP
   - Representación JSSP
   - Espacios de soluciones
   - Cómo resolver el JSS
   - Esquemas Generadores de Schedules
   - Reglas de Prioridad
   - Extensiones del JSSP
5. **Referencias Bibliográficas**

Referencias Bibliográficas

- Michael L. Pinedo. **Scheduling.Theory, Algorithms, and Systems.** Springer, 2021.
- Peter Brucker. **Scheduling Algorithms**. Springer, 2006.
- Stuart Russell & Peter Norvig. **Artificial Intelligence: A Modern Approach.** (4ª Edición), Pearson. 2022.
  - http://aima.cs.berkeley.edu/

Referencias Bibliográficas

- Joseph Y-T.Leung. **Handbook of Scheduling Algorithms, Models, and Performance.** Chapman & Hall/CRC, 2004.



- Pascal Van Hentenryck & Laurent Michel. **Constraint-Based Local Search.** The MIT Press, 2009.