Proyecto Fin de Carrera
Grado de Ingeniería de Organización Industrial

Análisis del problema de la programación de trabajos en máquinas no relacionadas con tiempos de setup

Autor: Manuel Alejandro Farinango Sierra
Tutor: José Manuel García Sánchez

Dep. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 20108

Trabajo Fin de Grado
Grado de Ingeniería de Organización Industrial

Análisis del problema de la programación de trabajos en máquinas no relacionadas con tiempos de setup

Autor:
Manuel Alejandro Farinango Sierra

Tutor:
José Manuel García Sánchez

Dep. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Carrera: Análisis del problema de la programación de trabajos en máquinas no relacionadas con tiempos de setup

Autor: Manuel Alejandro Farinango Sierra

Tutor: José Manuel García Sánchez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

Agradecimientos

A D. José Manuel García Sánchez por su valiosa ayuda, por su dedicación, por haberme brindado las herramientas necesarias para finalizar mi trabajo fin de grado satisfactoriamente y por haberme guiado tanto como profesor como tutor de este trabajo.

También me gustaría agradecer a mi familia y amigos, por todo su apoyo incondicional durante todos estos años.

Muchas gracias a todos!

Manuel Alejandro Farinango Sierra

Sevilla, 2018

Resumen

Este trabajo fin de grado presenta el análisis, mediante modelos matemáticos y heurística, de un problema de maquinas paralelas no relacionadas con tiempos de setup dependientes de la secuencia, donde el objetivo principal es minimizar el tiempo de terminación del último trabajo o makespan. Para logara nuestro objetivo desarrollaremos modelos matemáticos para obtener una secuencia, posteriormente construiremos un algoritmo el cual asignará los trabajos de tal forma que cada trabajo será asignado a la maquina menos cargada, de esta forma lograremos cumplir nuestro objetivo principal.

Dicho análisis se realizara utilizando los conocimientos obtenidos en las asignaturas de métodos cuantitativos de gestión, para construir los modelos matemáticos, y Programación y control de la producción, para desarrollar el algoritmo. El problema se clasifica como NP-hard. Así mismo el problema tiene la facultad de que, a pesar de su carácter teórico, tienen una amplia aplicación práctica, como pueden ser el caso de secuenciar las tareas de los hornos de cocción de cerámica. El estudio de Máquinas Paralelas no Relacionadas es de interés debido a su amplia aplicación en la industria manufacturera. La solución eficiente y eficaz de los modelos teóricos hace un planteamiento más sencillo para el estudio de modelos reales que permitan a la industria aumentar su grado de competitividad.

Índice

Índice de Tablas

Índice de Figuras

Notación

x	Vector de variables
≈	Signo de las restricciones [≤; =; ≥]
b	Vector de términos independientes
$G_i(x) \approx b$	Conjunto de m restricciones, i=1...m
f(x)	Función objetivo
1	Una sola máquina.
P	Máquinas paralelas idénticas.
Q	Máquinas paralelas uniformes.
R	Máquinas paralelas no relacionadas.
O	Taller de flujo regular (flow shop).
F	Taller de trabajo (job shop).
J	taller abierto (open shop).
αijr	Variable binaria
$Cmáx$	Makespan
Ci	Completion time de la maquina i
βijk	Variable binaria
P_{ij}	Tiempos de proceso del trabajo "j" en la máquina "i"
S_{ijk}	Tiempos de setup para procesar el trabajo "k" después de procesar el trabajo "j" en la maquina "i"
Xij	Variable medible continua (tiempo de inicio de proceso del trabajo "j" en la máquina"
Fij	Tiempo de finalización del proceso del trabajo "j" en la máquina "i"
$CS_{X_{ij}}$	Cota superior de X_{ij}
ω_{ijk}	Variable binaria
$\omega 1_{ijk}$	Variable binaria

$\omega 2_{ijk}$	Variable binaria
$\omega 3_{ijk}$	Variable binaria
$CS_{F_{ij}}$	Cota superior de F_{ij}
$CI_{\alpha_{ij}+\alpha_{ik}}$	Cota inferior de $\alpha_{ij} + \alpha_{ik}$
$CS_{X_{ij}-X_{ik}}$	Cota superior de $X_{ij} + X_{ik}$

1 INTRODUCCIÓN

En este Trabajo fin de grado analizaremos, mediante modelos matemáticos y heurística en lenguaje C++, un sistema de producción de maquinas paralelas no relacionadas con tiempos de setup dependientes de la secuencia. La base principal en la que se fundamenta la heurística es la asignación de los trabajos, ya que cada trabajo será asignado a la maquina menos cargada, obteniendo con ello un menor makespan o C_{max}. En primer lugar analizaremos el caso de maquinas paralelas no relacionadas sin tiempos de setup R_m/ /C_{max} y a continuación estudiaremos la evolución de problema añadiendo tiempos de setup dependientes de la secuencia $R_m/S_{ijk}/C_{max}$. Este análisis se realizara en dos partes...

En primer lugar desarrollaremos distintos modelos matemáticos, mediante la ***investigación operativa (Modelado)***, donde estudiaremos las distintas variables y ecuaciones que se desarrollan en los dos casos. Una vez estructurados los distintos modelos, estudiaremos el comportamiento de cada uno de ellos mediante la herramienta de resolución de modelos matemáticos"LINGO", la cual nos proporcionará una secuencia de asignación de los trabajos de cada modelo.

En segundo lugar desarrollaremos una heurística resolutiva, en lenguaje C++ (***Programación y control de la producción)***, el cual se encargará de la resolución del problema.

1.1. Programación Matemática

El desarrollo de los métodos cuantitativos o la investigación operativa, según la opinión de varios autores, ha representado uno de los grandes avances científicos más significativos desde mediados del siglo XX. En la actualidad es una herramienta muy utilizada en diversos campos, como administración, la economía y la ingeniería. Así mismo existen varios documentos, libros, archivos y miles de artículos científicos en revistas especializadas sobre el tema.

La investigación operativa tiene como base el método científico para investigar y ayudar a tomar decisiones sobre los problemas complejos de las organizaciones de hoy en día. Básicamente sigue los siguientes pasos:

I. Observación de un problema

II. Construcción de un modelo matemático que contenga un conjunto de variables, conjunto de restricciones y una o varias funciones objetivo del problema

III. La obtención, con la utilización de un ordenador, de las mejores soluciones posibles o soluciones optimas con la implantación de algoritmos exactos o heurísticos y para finalizar

IV. Calibración e interpretación de la solución y su comparación con otros métodos de toma de decisiones.

Para ilustrar la dificultad a la hora de poner en práctica la investigación operativa veremos un ejemplo, el cual se trata de un problema simple de asignación: Una empresa industrial tiene 70

trabajadores cada uno con características diferentes (electricistas, ingenieros, informáticos, contables, personal de administración, etc.) a los cuales vamos a asignar 70 actividades también diferentes. Si de alguna forma se pudiera determinar un valor que mostrase la asignación de un trabajador a una tarea determinada, tendríamos que escoger una entre 70! formas posibles de permutación de las asignaciones a dichas tarea que maximice el valor total. Este tipo de problemas de decisión son muy comunes hoy en día y se tienen que desarrollar modelos de programación matemática o métodos matemáticos para obtener así soluciones de los modelos, y algoritmos de ordenador (procedimientos pasos a paso) muy eficientes. Se dice que la investigación operativa constituye el 25% del tiempo total utilizado por los ordenadores para resolver problemas científicos.

Dentro de la investigación operativa la herramienta básica más utilizada es la programación lineal, esto es debido a que ofrece un inmenso ámbito de aplicaciones así como su simplicidad de implementación. Las técnicas de programación lineal se utilizan en un amplísimo espectro de problemas como, entre otros, de planificación y gestión de recursos humanos y materiales, de transporte, de planificación financiera, de organización de la producción, cómo organizar los horarios de vuelo de las azafatas en una compañía aérea, o la mezcla de ingredientes de un fertilizante para satisfacer las especificaciones agrícolas a un costo mínimo,… etc. En definitiva, una extensa gama de problemas que aparecen en las áreas de tipo industrial, económico, administrativo, militar...etc, en donde hay un objetivo a optimizar (maximización de beneficios, minimización de costes, maximización de la cobertura sanitaria, maximización de la producción, minimización de desperdicios,..Etc).

1.1.1 Orígenes de la programación lineal

En la actualidad la programación lineal se usa frecuentemente para resolver problemas de decisión, sin embargo antes de 1947 esta metodología era casi desconocida. Ninguna investigación significativa fue realizada antes de esta fecha, sin embargo hay que destacar que alrededor de 1823, Jean Baptiste Joseph Fourier (matemático de origen francés) parecía conocer el potencial del tema.

Un matemático ruso, Leonid Vitalievitx Kantorovitx, que publicó una extensa monografía en 1939, Matematitxeskie Metodi Organisatsi i Planirovaniia Proisvodstva (Métodos matemáticos para la organización y planificación de la producción) fue el primer investigador en reconocer que una amplia gama de problemas de producción y distribución tenían una estructura matemática y, que por lo tanto, se puedan formular con un modelo matemático. Pero desgraciadamente sus propuestas fueron desconocidas tanto en la Unión Soviética como en occidente durante dos décadas. Durante todo este periodo, la programación lineal experimentó un gran desarrollo tanto en Estados Unidos como en Europa. Después de la segunda guerra mundial, funcionarios del gobierno americano consideraron que la coordinación de las energías de toda una nación, debido al peligro de una guerra nuclear, requeriría la utilización de técnicas científicas de planificación. Esto se hizo posible gracias a la aparición del ordenador. Se crearon instituciones como la Corporación RAND en donde ingenieros y matemáticos trabajaron intensamente en la formulación y resolución de problemas matemáticos aplicados a la toma de decisiones. Entre otros, se propuso un modelo de programación lineal debido a su simplicidad y aplicabilidad, sin dejar de lado un marco lo suficientemente amplio para representar actividades interdependientes que han de compartir recursos limitados. El sistema (como, por ejemplo, la producción industrial) se compone de diversas actividades relacionadas entre ellas (formación, fabricación, almacenaje, transporte, distribución y venta). Este fue el primer modelo de programación lineal conocido.

1.1.2 Estructura del modelo matemático

Los problemas de optimización se distinguen porque siempre hay que realizar una actividad donde participan los elementos y recursos del problema, dichas actividades están sujetas a una o varias especificaciones (restricciones) y existen varias formas de realizar la actividad. Lo que se pretende es realizar la forma más eficiente (función objetivo). En general optimizar es usar eficientemente un conjunto de elementos para realizar una serie de actividades.

Así mismo cualquier problema de optimización posee un parámetro implícito que es el tamaño, el cual está determinado por el número de instancias que participan en el mismo, que no afecta en la definición del problema por el cambio del mismo, aunque si puede variar factores asociados a la resolución.

Atendiendo a la complejidad del problema distinguimos dos clases:

- Los problemas polinomiales o clase P: los cuales son problemas no complejos, se resuelven de manera exacta, no implica mucho tiempo de proceso y existen modelos para su resolución exacta.
- Los problemas no polinomiales o clase NP: son problemas complejos, la obtención de su óptimo es costoso en tiempo de proceso dependiendo del tamaño de la dimensión del problema.

En nuestro caso al realizar el análisis de maquinas paralelas no relacionadas con tiempos de setup dependientes de la secuencia se trata de un problema de case NP-Hard.

Por otro lado para modelar un problema hay que generar un conjunto de expresiones y relaciones matemáticas que se conoce como modelo de programación matemática. En él se encuentran las especificaciones del sistema que son todas las acciones definidas y normas determinadas que lo definen, los actores que son las entidades con las que trabaja el sistema, tangibles o intangibles, y que pueden estar sometidas a cualquier proceso, y las actividades de decisión del sistema que son las acciones de valor indeterminado que se producen en el mismo y que se definen sobre los actores del sistema.

El modelo matemático está compuesto por tres componentes principales:

1. **Conjunto de variables:** referidas a las actividades que se desarrollan en el sistema que se quiere optimizar. Se distinguen en :
    - Variables continuas: valores reales, identifican medidas.
    - Variables enteras: entidades discretas.
    - Variables binarias: un tipo especial de variable entera, toman únicamente valor entre 1 y 0. Se identifican como variables para la toma de decisión que realice el modelo.
2. **Un conjunto de restricciones:** definen las especificaciones, características y normas del sistema.
3. **Una función objetivo:** criterio que se desea optimizar.

Formato del modelo

$$Min \quad f(x)$$
$$sujeto\ a$$
$$G_i(x) \approx b \quad i = 1...m$$

x: Vector de variables

≈ : Signo de las restricciones [≤; =; ≥]

b: Vector de términos independientes

$G_i(x) \approx b$: Conjunto de *m* restricciones, $i=1...m$

f(*x*) : Función objetivo

Min: El objetivo del problema se considera de forma genérica en modo de minimizar una función. Cualquier expresión de maximizar una función se puede convertir en minimizar por una sencilla transformación:

$$Max\ f(x) \Rightarrow -Min - f(x)$$

Signos posibles en las restricciones: ≤, = y ≥.

En el caso de que un sistema encuentre una restricción que se defina con mayor o menor estricto, se realizar una pequeña modificación para expresarlo de forma correctamente. Tomemos por ejemplo una restricción genérica g(x)<b. El modo de operar es el siguiente:

- Si todas las variables son enteras, el problema se soluciona definiendo como término independiente, el término independiente menos 1, b-1, así pues en el caso menor estricto el valor máximo al que podría llegar la expresión g(x) <b → g(x)≤b-1.

En caso de mayor estricto, g(x)>b, se cambiaría por b+1; g(x)≥b+1

- Por otra parte en el caso de tener variables que tomen valores continuos, es necesario cometer un mínimo error controlado en la especificación, definiéndose un valor ε tan pequeño como queramos y la restricción se define de la siguiente manera, en el caso de menor restricto (<), como g(x)≤b-ε . En caso de mayor estricto (>), g(x)≥b+ε.

El ajuste del valor de ε se realiza según la precisión de los valores que podrían llegar a tomar las variables continuas y por tanto las función g(x).

Signos de las variables:

Aquí pueden aparecer dos casos:

- Si determinada variable tomara únicamente valores negativos $x \leq 0$, se realizaría un simple cambio de variables

  $x' = -x$; $x' \geq 0$

  y de esa manera trabajaríamos el problema con x' en lugar de x

- Si determinada variable *x* fuera libre, es decir, no tuviera restricción de signo, entonces el cambio a

realizar sería:

$x = u - v$; $u \geq 0$; $v \geq 0$.

1.1.3 Componentes del sistema

Nuestro objetivo es convertir la descripción de un sistema, en nuestro caso el sistema de producción de maquinas paralelas no relacionadas, a un lenguaje matemático utilizable por las técnicas de optimización de la programación matemática, lo que en términos matemáticos se conoce como modelar el problema. El resultado será un modelo matemático, que se puede denominar modelo de programación matemática o modelo de optimización, que posteriormente resolveremos para obtener una solución a la gestión de ese sistema.

Los componentes principales de un sistema son :

- **Actores**

Los actores son todos los elementos que participan en el sistema. Son de naturaleza diversa, desde personas, herramientas, lugares, tiempo, etc.

Suelen tener asociada información que denominaremos como atributos o características y que tiene que ser una información numérica. Participan en las acciones que se producen en el sistema y también soportan las especificaciones del mismo.

- **Actividades de decisión**

Acciones directas e indirectas que se producen en el sistema para las que decidiremos su valor, el cual no está determinado. Se encuentran asociadas a actores.

- Características: Las actividades son acciones simples, no pueden ser fruto de un cálculo, son función simple o combinación de otras actividades de decisión, es posible poseer actividades de decisión cuyos valores estén condicionados al valor de otras actividades de decisión.

Las actividades de decisión definen las variables principales del modelo. Los valores que pueden asignarse a estas actividades son valor binario (1/0), valor entero o valor continuo.

- **Especificaciones**

Normas y acciones definidas que se deben cumplir en el sistema.

Dan lugar a las restricciones del problema, pero no debe confundirse especificación con restricción. La restricción es una expresión matemática mientras que la especificación es una característica que debe cumplir el modelo y que generará una o varias restricciones.

- **Cálculos de apoyo**

Son funciones que se aplican sobre variables del sistema para obtener el valor de un cálculo relevante del mismo. Se representan también como variables del problema, conocidas como variables auxiliares.

- **Función Objetivo**

La función objetivo es la ecuación que será optimizada dadas las limitaciones o restricciones determinadas y con variables que necesitan ser minimizadas o maximizadas usando técnicas de programación matemática

1.2. Programación y control de la producción

La programación de la producción es un proceso de toma de decisiones que desempeñan un papel crucial tanto en la fabricación como en los sistemas de servicios e industrias. En el actual entorno competitivo, rápidamente cambiante, una programación efectiva se convierte en una necesidad para la supervivencia en el mercado de la industria. Las empresas tienen que cumplir con los plazos de producción y fechas de entrega comprometidos con los clientes y el no hacerlo puede derivar en una pérdida económica significativa y de confianza del cliente.

En términos generales, la programación de la producción se refiere a la asignación de unos recursos, generalmente limitados, a unas tareas, durante un tiempo. Es un proceso de toma de decisiones cuyo objetivo es la optimización de uno o más objetivos sometidos a limitaciones o restricciones del problema. Los recursos y las tareas pueden tomar muchas formas. Por ejemplo, los recursos pueden ser máquinas en un taller, pistas de aterrizaje en un aeropuerto, los equipos de una obra de construcción, unidades de procesamiento por medio de ordenadores, etc. Las tareas correspondientes a esos recursos pueden adoptar las siguientes formas: operaciones en un proceso de producción, despegues y aterrizajes en el aeropuerto, las etapas de una obra, producción de las maquinas, la ejecución de programas de ordenador, etc. Además, cada tarea puede tener diferente tiempo de procesamiento, nivel de prioridad (o peso), o una fecha de inicio y otra de vencimiento. El objetivo también puede tomar muchas formas. Uno de los objetivos es la reducción al mínimo del tiempo en el que se termina la última tarea (lo denominamos la minimización del makespan) que es el objetivo principal de nuestro análisis .

1.2.1 Origines de la programación de la producción

La programación de la producción comenzó a ser tomada en serio en la industria al principio del pasado siglo con los trabajos de Henry Gantt (Gantt, 1910 y Gantt, 1919) y otros pioneros que siguieron sus ideas. Sin embargo, la primera publicación relacionada con la literatura de la investigación operativa necesito varios años para que surgiera ese hecho. Algunas de estas primeras publicaciones aparecieron en la década de 1950. En concreto, tenemos los artículos de Smith (1956), Johnson (1954), y Jackson (1956). Durante la década de 1960 gran parte de los trabajos se centraron en la programación tanto dinámica como entera para los problemas de programación de la producción. Después de que Richard Karp publicara su famoso artículo sobre teoría de la complejidad (Karp, 1975), la investigación en la década de 1970 se centró principalmente en la jerarquía de la complejidad de problemas de programación de la producción. En la década de 1980 se siguieron diferentes direcciones tanto en la parte teórica como en la práctica, con una atención creciente en la programación estocástica (Pinedo, 2008). Además, como los ordenadores personales empezaron a ser de amplia difusión en el ambiente empresarial, los sistemas de programación se estaban enfocando en el desarrollo de programas que pudieran utilizarse en la práctica.

La planificación y programación de la producción es una herramienta importante en la toma de decisiones para la mayoría de las empresas tanto en el sector industrial como en el de servicios, así

como para la mayoría de los entornos de procesamiento de información donde puede tener un gran impacto en la productividad de un proceso. En la industria, el objetivo principal de la programación de la producción es reducir al mínimo el tiempo de producción y los costos, teniéndose en cuenta que en una planta de producción se deben decidir cosas como lo que se debe hacer, cuándo, con qué personal, y/o con qué equipo. Del mismo modo, la programación de la producción de las empresas de servicios, tales como líneas aéreas y el transporte público, tiene como objetivo maximizar la eficiencia de las operaciones y reducir sus costes.

Las empresas utilizan la programación de la producción para planificar, de la forma más eficiente posible, sus recursos humanos y materiales tanto en sentido temporal hacia atrás como hacia adelante. La programación de la producción hacia atrás es la planificación de las tareas con una fecha de vencimiento para determinar la fecha de inicio y/o cualquier variación de las necesidades requeridas, mientras que la programación hacia adelante es la planificación de las tareas con fecha de inicio para determinar la fecha de envío o la fecha de vencimiento

La programación de la producción puede ser difícil, tanto desde un punto de vista teórico como práctico. Las dificultades teóricas son similares a las encontradas en otras ramas de la optimización combinatoria. Así, los resultados teóricos obtenidos en otros problemas de optimización se pueden aplicar, al menos parcialmente, a los problemas de programación de la producción. Pero, las dificultades encontradas durante la implementación son de un tipo completamente diferente. Están relacionadas con la modelización de los problemas de programación de la producción en el mundo real. En términos generales, se puede decir que cada problema de programación de la producción de la vida real es diferente, y requiere por lo tanto una modelización general, que en la mayoría de los casos se tiene que hacer desde cero. Para poder introducirse de fondo en la teoría de la programación de la producción y sus aplicaciones pueden consultarse los trabajos de Conway, Maxwell y Miller (1967), Brucker (1995), Pinedo (2008) y Brucker (2005), entre muchos otros.

1.2.2 Notaciones

Los problemas de programación de la producción vienen determinados por el número de trabajos y las operaciones a procesar, el número y tipo de máquinas, el flujo de las operaciones en las máquinas y los criterios de optimización, entre otros datos. Normalmente se considera un número de trabajos y máquinas finito y determinista. En los problemas de producción hay un conjunto de N trabajos, donde $N = 1, 2, ..., n$, que hay que procesar en un conjunto de M máquinas, $M = 1, 2, ...,m$. Utilizamos los subíndices j y k para referirnos a los trabajos y el subíndice i para las máquinas y donde pij representa el tiempo de proceso del trabajo j en la máquina i.

Para la clasificación de los diversos problemas de la producción se han propuesto numerosas notaciones, entre las que destaca la notación cuádruple empleada por Conway, Maxwell y Miller (1967), la algo más elaborada presentada por Rinnooy Kan (1976) y la más frecuentemente usada, introducida por Graham et al. (1979), basada en la tripleta $\alpha/\beta/\gamma$.

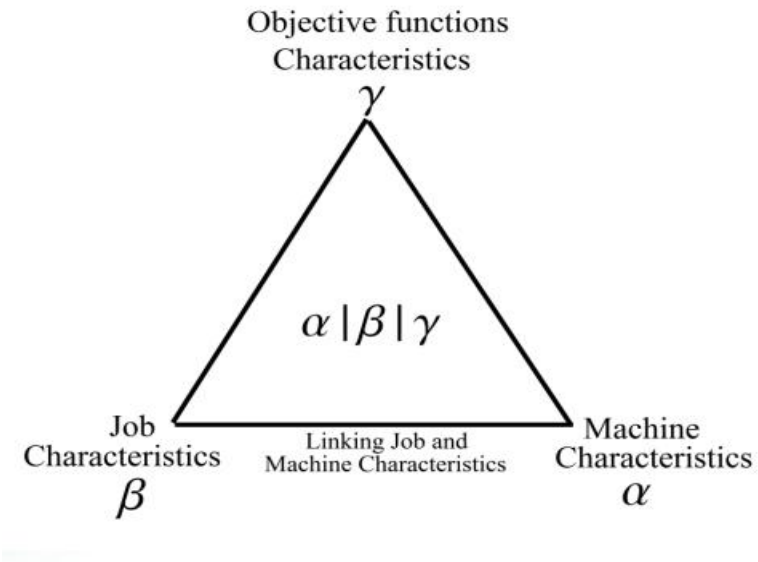


Figura 1. Notación tripleta

El campo α representa las características de la maquina, puede tener los valores de:

1: una sola máquina.

P: máquinas paralelas idénticas.

Q: máquinas paralelas uniformes.

R: máquinas paralelas no relacionadas.

O: taller de flujo regular (flow shop).

F: taller de trabajo (job shop).

J: taller abierto (open shop).

El campo β indica las características de los trabajos, las cuales son muy numerosas. En el caso de no figurar de forma expresa, se entiende que no se da la característica indicada. Los subcampos más usuales son:

- ✓ ***pmtn (Interrupciones):*** se permite interrumpir las operaciones y continuarlas más tarde o en otra máquina.
- ✓ ***Prec (precedencia):*** relaciones de precedencia.
- ✓ ***rj* :** fecha de llegada (disponibilidad) de los trabajos, son distintos de cero.
- ✓ ***Rj:*** indica en qué orden el trabajo "j" va a ser procesado por las distintas maquinas.
- ✓ ***Snsd*:** tiempo de cambio (o setup) independientes de la secuencia. Se denomina como S_{ij} (tiempo de setup en la maquina "i" antes de procesar el trabajo "j").
- ✓ ***Ssd*:** tiempo de cambio (o setup) dependientes de la secuencia. Se denotan por S_{ijk} (tiempo de setup de la maquina "i" para procesar el trabajo "k" después de procesar el trabajo "j").
- ✓ ***dj* :**fecha de entrega del trabajo "j".
- ✓ $\overline{dj}$ **:** fecha de finalización obligatoria (*deadline*).
- ✓ ***Prmu (permutación):*** el orden de entrada de los trabajos (secuencia) es el mismo para todas las máquinas. Solo aplicable cuando $\alpha = F$.
- ✓ ***no*-idle (maquina no ociosa):** no está permitido tiempos ociosos de las maquinas entre trabajos. Una vez que la maquina empieza a procesar su primer trabajo, no puede para.

- ✓ ***batch (lotes)***: el proceso se realiza por lotes. Hay dos variantes:
  - ***p-batch***: todas las maquinas consideran un tamaño máximo de lote. Donde el tiempo de proceso del lote es el mayor de los tiempos de proceso de los trabajo en dicho lote.
  - **s-batch:** todas las maquinas consideran un tamaño máximo de lote. Donde el tiempo de proceso del lote es la suma de los tiempos de proceso de los trabajo en dicho lote.
- ✓ ***Nwt (espera no permitida)*:** sin esperas entre dos máquinas. A veces implica retrasar el inicio de las operaciones de un trabajo.
- ✓ ***pmtn (interrupciones)*:** se trata de interrumpir el proceso del trabajo "j" que se está procesando debido a diferentes causas. Distinguimos tres tipos:
  - ***ptmn-non-resumable***: se pierde el trabajo hecho de la tarea interrumpida, y se empieza otra vez cuando se reinicie.
  - ***ptmn-semi-resumable:*** se pierde parte del trabajo que ha sido interrumpido.
  - ***ptmn-resumable:*** no se pierde el trabajo hecho tras la interrupción, se reinicia por donde se había dejado tras la interrupción.

- ✓ ***Almacenaje*:** Existe una capacidad (o buffer) finita para almacenar el producto en curso. Si se llena el buffer de una máquina, está quedará bloqueada por el último trabajo que ha procesado y no ha podido dejar en el buffer. Por lo tanto la maquina no puede prosear el siguiente trabajo hasta que quede sitio.
- ✓ ***recrc*:** recirculación. Indica que un trabajo (o más) pueden necesitar ser procesados más de una vez por alguna (o varias) de las máquinas. Solo cuando $\alpha = O, F, J$.
- ✓ ***Mj*** : restricción de uso de máquinas. Implica que un trabajo (o más) solo puede ser procesado en determinadas máquinas.
- ✓ ***pij = p***: todos los tiempos de trabajo son iguales a p. Una variante común es hacer $p = 1$.

1.2.3 Clasificación de los modelos de programación de la producción.

En la programación de la producción se distinguen claramente dos fases, la primera fase trata la secuenciación y la segunda fase trata la programación. La secuenciación consiste en la asignación de recursos en el tiempo para la realización de un conjunto de tareas. Previa a la secuenciación, la planificación de la producción se realiza a priori, es decir, se sabe qué productos se van a realizar, las cantidades de cada uno de ellos y la configuración, disposición y cantidad de recursos disponibles. Esta secuenciación es de vital importancia para las empresas.

Para generalizar nos referiremos a los recursos como máquinas y a cada pedido de la producción como trabajo. Así pues, la programación de la producción la podemos ver como una asignación de los trabajos a las máquinas (asignación), la ordenación de los trabajos asignados a cada máquina (secuenciación) y la obtención de los tiempos de inicio y finalización de cada trabajo (programación).

Una vez explicada las dos fases de la programación de la producción, podemos clasificar los distintos entornos de producción atendiendo a la configuración productiva:

- ➢ **Problemas de una máquina (Single machine):** solo existe una máquina, cada trabajo tiene que procesarse en la máquina y cada trabajo solo tiene una operación. No hay problema de asignación pero si de secuenciación.

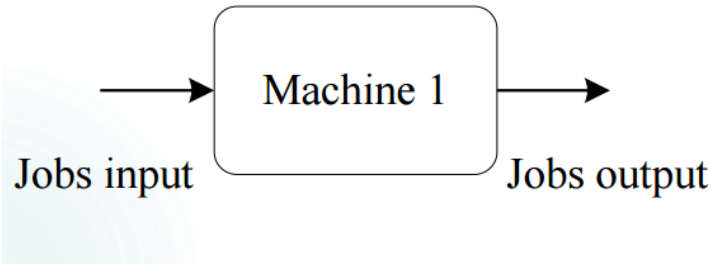


Figura 2. Máquina única

- **Problemas de máquinas paralelas (Parallel machines):** existen dos o más máquinas dispuestas en paralelo, donde los trabajos se procesan en una sola maquina y cada trabajo tiene una única operación. En general, se tienen que asignar y secuenciar los trabajos en las máquinas, aunque dependerá del objetivo optimizar. Los problemas de máquinas paralelas pueden asimismo dividirse según la naturaleza de las máquinas en:

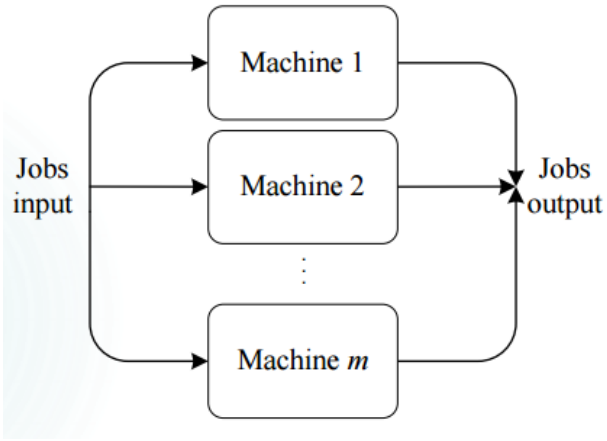


Figura 3. Maquinas paralelas idénticas

  - **Máquinas paralelas idénticas (Identicla parallel machine):** cada trabajo se procesa a la misma velocidad con independencia de la máquina a la que ha sido asignado.

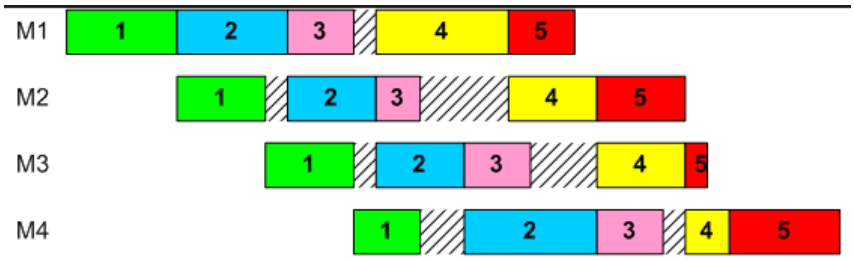
  - **Máquinas paralelas uniformes (Uniform parallel machines)**: maquinas con diferentes velocidades de procesado (v_i), es decir algunas máquinas son capaces de procesar todos los trabajos más rápido o más lento que las demás.

  - **Máquinas paralelas no relacionadas (Unrelated parallel machine):** la velocidad de proceso depende de la máquina y del trabajo que está procesando. Es un caso más general que engloba a los otros dos y nuestro caso de estudio.

- **Entornos tipo taller:** Están compuestos por maquinas en serie, cada máquina tiene una función diferente, cada trabajo visita todas las maquinas (cada trabajo tiene una operación en cada máquina). Además, existe una relación de precedencia entre las etapas para cada trabajo. A esta relación se la denomina ruta. Existen distintos tipos de problemas tipo taller

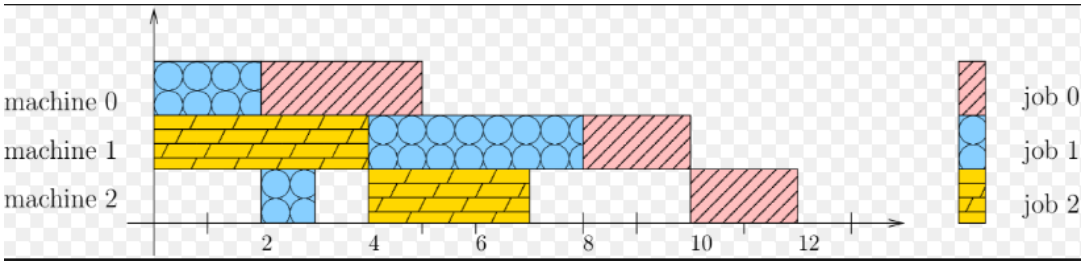
dependiendo de su tipo de ruta, por lo tanto podemos clasificar los problema según su ruta en:

- **Taller de flujo regular (flow shop):** todos los trabajos tienen la misma ruta predeterminada, es decir, existe una ordenación previa de etapas o máquinas que es la misma para todos los trabajos.



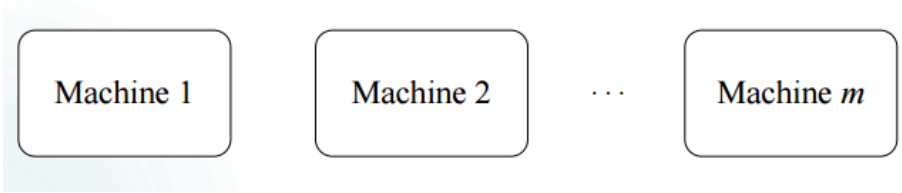
Figiura 4. Flow shop

- **Taller de trabajo (job shop):** cada trabajo tiene su propia ruta a seguir dentro de las máquinas.



Figiura 5. Job shop

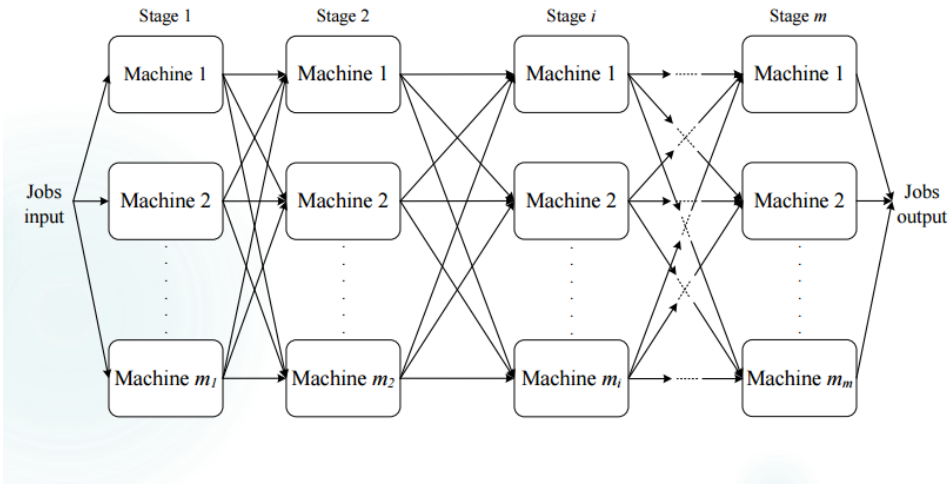
- **Taller abierto (Open shop):** el más general y complejo de los talleres, no existe ninguna restricción en la ruta, por lo que las operaciones pueden efectuarse en cualquier orden.



Figiura 6. Open shop

❖ **Caso particular**

➢ **Entorno híbridos:** es la unión de maquinas paralelas con algún taller. Los talleres están formados por estados (stages), en vez de maquinas (flow shop, job shop, open shop). Cada estado está formado por máquinas paralelas.(idénticas, uniformes, no relacionadas).



Figiura 7. Entornos hibridos

Finalmente, el campo γ nos da información acerca del criterio de optimización que se ha empleado. Antes definiremos una serie de conceptos y parámetros adicionales, que nos permitirán definir mejor las medidas de optimización

- C_j : tiempo de terminación del trabajo "j" (completion time). Instante en el que el trabajo finaliza su proceso en el entorno.
- F_j : tiempo de flujo del trabajo (flowtime). Tiempo que el trabajo está en el entorno. $\boldsymbol{F_j = C_j - r_j}$
- L_j: retraso del trabajo "j" (lateness). $\boldsymbol{L_j = C_j - d_j}$
- T_j :tardanza del trabajo "j" (tardiness). $\boldsymbol{Tj = \max\{L_j, 0\}}$.
- E_j : adelanto del trabajo "j"(earliness). $\boldsymbol{E_j = \max\{-L_j, 0\}}$.
- U_j :trabajo tardío (tardy job). $\mathbf{U_j=1}$ **si** $\boldsymbol{T_j > 0}$ es decir $\mathbf{C_j > d_j}$. $\mathbf{U_j=0}$ en caso contrario.

Procedamos a definir ahora algunos de los objetivos de optimización más comunes:

- **No relacionados con las fechas de entregas:**

- $\boldsymbol{C_{máx}}$**:** minimización del tiempo máximo de finalización de todos los trabajos o makespan.
- $\boldsymbol{F_{máx}}$**:** minimización del tiempo máximo de flujo.
- Total completion time: $\sum C_j = \sum_{j=1}^{n} C_j$
- Total flowtime: $\sum F_j = \sum_{j=1}^{n} F_j$

- **Relacionadas con fechas de entrega:**

- $\boldsymbol{L_{máx}}$: minimización del máximo retraso.
- $\boldsymbol{T_{máx}}$: minimización del la tardanza máximo.
- $\boldsymbol{E_{máx}}$: minim+ización del adelanto máximo.
- Total lateness: $\sum L_j = \sum_{j=1}^{n} L_j$
- Total tardiness: $\sum T_j = \sum_{j=1}^{n} T_j$
- Total earliness: $\sum E_j = \sum_{j=1}^{n} E_j$
- Number of tardy jobs: $\sum U_j = \sum_{j=1}^{n} U_j$

Muchos de estos objetivos admiten una ponderación de los mismos sin más que añadir el correspondiente peso (denominado ω_j) a sus componentes.

❖ **Objetivos ponderados:**

- máx $\omega_j C_j$ (maximum weighted completion time)
- máx $\omega_j F_j$ (maximum weighted Flowtime)
- máx $\omega_j L_j$ (maximum weighted lateness)
- máx $\omega_j T_j$ (maximum weighted tardiness)
- máx $\omega_j E_j$ (maximum weighted earliness)
- Total weighted completion time: $\sum \omega_j C_j = \sum_{j=1}^{n} \omega_j C_j$
- Total weighted Flowtime: $\sum \omega_j F_j = \sum_{j=1}^{n} \omega_j F_j$
- Total weighted lateness: $\sum \omega_j L_j = \sum_{j=1}^{n} \omega_j L_j$
- Total weighted tardiness: $\sum \omega_j T_j = \sum_{j=1}^{n} \omega_j T_j$
- Total weighted earliness: $\sum \omega_j E_j = \sum_{j=1}^{n} \omega_j E_j$
- Weighted number tardy jobs: $\sum \omega_j U_j = \sum_{j=1}^{n} \omega_j U_j$

1.3. Motivación

A la hora de iniciarse el análisis del presente Trabajo Fin de Grado buscamos un problema de producción relativamente sencillo que tuviera amplia aplicación en distintos ámbitos y del que pudieran derivar problemas más complejos. Nos centramos en el problema máquinas paralelas, y dentro de él, en el de máquinas paralelas no relacionadas, ya que éste último es el más general de los tres problemas de máquinas paralelas. Este entorno, a pesar de parecer de carácter marcadamente teórico, tiene aplicaciones directas en muchos entornos de las industrias manufactureras, en empresas de cerámicas, comunes en la industria de la producción de pintura y de plástico (donde se requiere una completa limpieza entre la producción de una orden de producción y otra), como también en la industria de la producción de textiles, de vidrios, de químicos, de papel, y de servicios.

2 DESCRIPCIÓN DEL PROBLEMA

El problema de máquinas paralelas no relacionadas (R_m) consiste en *n* trabajos disponibles en el tiempo cero que deben ser programados en *m* máquinas en paralelo que pueden procesar trabajos con tiempos de procesamiento arbitrarios. Cada trabajo es asignado a una máquina, la cual será la menos cargada de todas las (R_m) maquinas, y cada máquina puede procesar un trabajo a la vez. Esto significa que si el trabajo *j* es programado en la máquina *i*, el tiempo necesario para procesar ese trabajo es P_{ij}, que depende del trabajo *j*, y la máquina *i*. Adicionalmente, el problema considera tiempos de setup dependientes de la secuencia (S_{ijk}), donde el tiempo necesario de setup para procesar el trabajo *k* después de procesar el trabajo *j* en la máquina *i* puede ser diferente de aquel que se necesita para procesar el trabajo *j* después del trabajo *k* en la misma máquina. La idea es analizar el programa de producción que minimice el tiempo de terminación del último trabajo (también llamado makespan o *Cmax*), además de analizar otros tipos de metodologías. En términos de la notación de tres parámetros propuesta por Graham *et al.* (1979) el problema en estudio en este trabajo puede ser representado por la tripleta $R_m \,|S_{ijk}|C_{max}$

2.1. Modelado matemático del problema

Como se menciono con anterioridad, en primer lugar llevaremos a cabo un análisis del caso más simple, dentro de los sistemas de maquinas paralelas no relacionadas, el cual es $R_m \mid |C_{max}$ y posteriormente llevaremos a cabo el análisis del mismo caso añadiendo la restricción S_{ijk} . De esta forma podremos observar la evolución matemática de dicho caso.

2.1.1 Modelo básico

$R_m \mid |C_{max}$

En este caso básico vamos a modelar matemáticamente el problema, construyendo en primer lugar su tabla de actores correspondiente, en la cual distinguiremos los distintos **Actores** que intervienen en el problema, que tipo de actores son (Unitarios o Medibles) y los atributos de cada actor. Seguidamente desarrollaremos la **Actividad de Decisión** del problema, así como sus **Especificaciones** correspondiente y por último la **Función Objetivo**. El objetivo, es el ya mencionado, minimizar el makespan ($C_{\max}$), cuyo valor será siempre no negativo.

Tabla 1. Tabla de actores modelo básico

Actores	Tipo	Atributos				
		Nombre	**Parámetro**	**Tipo Valor**	**Pertenencia**	**Aplicación**
Maquinas	Unitario i=1…m	Tiempo de proceso del trabajo j en la maquina i	P_{ij}	Continuo	Compartida	Firme
Trabajos	Unitario J=1…n		P_{ij}			

Actividad de decisión

Asignar (Trabajos, Maquinas)

$$\alpha_{ij} \begin{cases} 1 \; si \; asigno \; el \; trabajo \text{ "j" a la máquina "i"} \\ 0 \; en \; otro \; caso \end{cases} \quad (1)$$

Especificaciones

$$\forall_j : \sum_{i=1}^{m} \alpha_{ij} = 1 \quad (2)$$

Función Objetivo

$$Min \; C_{máx}$$

$$C_i \leq C_{max} \quad \forall_i \quad (3)$$

$$\forall_i : \; C_i = \sum_{j=1}^{n} p_{ij} * \alpha_{ij} \quad (4)$$

$$\alpha_{ij} \in \{0,1\} \quad \forall j \in N \; , \forall i \in N$$

Como podemos observar, los actores que intervienen en el modelo básico son las maquinas y los trabajos. El conjunto **(1)** hace referencia a una variable binaria, con valores 0 o 1, la cual indica que el trabajo "j" va asignado a la máquina "i". El conjunto **(2)** es una especificación de asignación la cual especifica que cada trabajo "j" va a ser asignado a una única maquina "i". Para finalizar en el conjunto **(3)** calculamos la función objetivo la cual es, como ya se ha dicho anteriormente, minimizar el tiempo de terminación del último trabajo o C_{max}. Para ello necesitamos saber el tiempo total de trabajo de cada máquina "i", el cual se obtiene en el conjunto **(4)**, e imponemos que sea menor igual que C_{max}, que es el tiempo de la maquina que más trabaja.

2.1.2 Modelo objetivo 1

R_m $\|S_{ijk}$ $\|C_{max}$

Ahora pasamos a analizar el segundo caso añadiendo al problema la restricción de tiempos de Setup dependientes de la secuencia (S_{ijk}). Al igual que en el problema anterior desarrollaremos su **tabla de Actores**, **Actividades de decisión, Restricciones y Función Objetivo.**

Tabla 2. Tabla de actores modelo objetivo 1

Actores	Tipo	Atributos				
		Nombre	**Parámetro**	**Tipo Valor**	**Pertenencia**	**Aplicación**
Maquinas	Unitario i=1…m	Tiempo de proceso del trabajo j en la maquina i	P_{ij} S_{ijk}	Entero	Compartida	Firme
Trabajos	Unitario j,k=1…n	Setup	P_{ij} S_{ijk}	Entero	Compartida	Firme
Posición	Unitario r=1…m/n					

Actividad de decisión

Asignar (Trabajos, Maquinas, Posición)

(1)

$$\alpha_{ijr} \begin{cases} 1 \; si \; asigno \; el \; trabajo \text{ "j" a la máquina "i" en la posición "r"} \\ 0 \; en \; otro \; caso \end{cases}$$

Especificaciones

Especificación de asignación

$$\forall_j : \sum_{r=1}^{m/n} \sum_{i=1}^{m} \alpha_{ijr} = 1 \quad (2)$$

Especificacion de posición

$$\forall_i, \forall_r : \sum_{j}^{n} \alpha_{ijr} \leq 1$$

Especificación de contiguidad

$$\forall_i, \forall_r \backslash r > 1: \sum_{j}^{n} \alpha_{ijr} \leq \sum_{j}^{n} \alpha_{ijr-1}$$

Función Objetivo

$$Min\ C_{máx} \tag{3}$$
$$C_{máx} \geq C_i \quad \forall_i$$

$$\beta_{ijk} \begin{cases} 1\ Si\ aplico\ Setup\ de\ los\ trabajos\ j,k\ en\ la\ maquina\ "i" \\ 0\ en\ otro\ caso \end{cases} \tag{4}$$

$$\forall_i, \forall_j, \forall_k, \forall_r \qquad \text{Si } \alpha_{ijr} = 1\ y\ \alpha_{ikr+1} = 1\ Entonces\ \beta_{ijk} = 1$$

Modelamos :

$$\alpha_{ijr} + \alpha_{ikr+1} \geq 2\ Entonces\ \beta_{ijk} = 1$$

$$\beta_{ijk} = 0\ Entonces\ \alpha_{ijr} + \alpha_{ikr+1} \leq 1$$

$$\alpha_{ijr} + \alpha_{ikr+1} \leq 1 + \left(CS_{\alpha_{ijr} + \alpha_{ikr+1}} - 1\right) * \beta_{ijk}$$

$$\alpha_{ijr} + \alpha_{ikr+1} \leq 1 + (2-1) * \beta_{ijk}$$

$$\alpha_{ijr} + \alpha_{ikr+1} \leq 1 + \beta_{ijk}$$

$$\forall_i:\ C_i = \sum_{j=1}^{n} \sum_{r=1}^{m/n} p_{ij} * \alpha_{ijr} + \sum_{j}^{n} \sum_{k}^{n} S_{ijk} * \beta_{ijk} \tag{5}$$

$$\alpha_{ijr} \in \{0,1\}, \beta_{ijk} \in \{0,1\}\ , \forall j \in N\ , \forall k \in N, \forall i \in N, \forall r \in N$$

CS= cota superior

En este caso al tratarse de un problema con tiempos de setup dependientes de la secuencia, los actores que interviene en el modelo son las maquinas, los trabajos y las posiciones de cada trabajo, este último es fundamental ya que nos informa la posición de cada trabajo en cada máquina y por tanto sabremos qué tiempo de setup le corresponde a cada trabajo.

El conjunto **(1)** hace referencia a una variable binaria, con valores 0 o 1, la cual asigna el trabajo "j" en la posición "r" en la máquina "i". El conjunto **(2)** está formado por tres especificaciones, la primera es una especificación de asignación la cual especifica que cada trabajo "j" en la posición "r", va a ser asignado a una única maquina "i", la segunda es una especificación posición la cual obliga que en la maquina "i" y en la posición "k" solo puede haber un trabajo asignado y la tercera es una especificación de contigüidad, es decir, con esto conseguimos que en la maquina "i" se asignen los trabajos en posiciones "r" contiguas. Para finalizar en los conjuntos **(3)**, **(4)** y **(5)** calculamos la función objetivo la cual es, como ya se ha dicho anteriormente, minimizar el tiempo de terminación del último trabajo o C_{max}. Por lo tanto es necesario saber los tiempos de setup correspondientes a cada trabajo, los cuales obtenemos mediante la variable binaria "α_{ijk}," la cual aplica el tiempo de setup del trabajo "k" después de procesarse el trabajo "j" en la maquina "i" **conjunto (4**). Para finalizar calculamos el tiempo total de trabajo de cada máquina "i" que es la suma de los tiempo de proceso de cada trabajo mas sus tiempos de setup correspondientes **conjunto (5).**

2.1.3 Modelo objetivo 2

De forma adicional hemos modelado el mismo problema, con tiempos de setup, sustituyendo el actor **Posición** (unitario) por el actor **Tiempo** (medible continuo) ya que de esta forma el modelo puede llegar a ser más estable. Más adelante estudiaremos el comportamiento de cada modelo y compararemos el resultado de cada uno.

Tabla 3. Tabla de actores modelo objetivo 2

Actores	Tipo	Atributos				
		Nombre	**Parámetro**	**Tipo Valor**	**Pertenencia**	**Aplicación**
Maquinas	Unitario i=1…m	Tiempo de proceso del trabajo j en la maquina i	P_{ij} S_{ijk}	Continuo	Compartida	Firme
Trabajos	Unitario J,k=1…n	Setup	P_{ij} S_{ijk}	Continuos	Compartida	Firme
Tiempo	MC					

Actividad de decisión

Programar (Trabajos, Maquinas, Tiempo)

X_{ij}: Uds de tiempo en el que se programa el trabajo "j" en la máquina "i" (1)

Asignar (Trabajos, Maquinas)

α_{ij} $\begin{cases} 1\ si\ asigno\ el\ trabajo \text{ "j" a la máquina "i"} \\ 0\ en\ otro\ caso \end{cases}$

$$\forall_i \forall_j \quad \text{si } X_{ij} \geq 1 \text{ Entonces } \alpha_{ij} = 1$$

$$\alpha_{ij} = 0\ Entonces\ X_{ij} \leq 0$$

$$X_{ij} \leq 0 + (CS_{X_{ij}} - 0) * \alpha_{ij}$$

Especificaciones (2)

$$\forall_{\boldsymbol{j}} \quad \sum_{i}^{m} \alpha_{ij} = 1$$

Especificación de solape de trabajos

Calculo simple : $\forall \boldsymbol{i}, \forall \boldsymbol{j} \quad F_{ij} \geq X_{ij} + P_{ij} * \alpha_{ij}$

$$\forall_i \forall_j \forall_k \ \ Si\ \alpha_{ij} + \alpha_{ik} \geq 2\ Entonces\ (F_{ij} \leq X_{ik}\ O\ \ F_{ik} \leq X_{ij})$$

Modelamos

$$F_{ij} \leq X_{ik} \rightarrow \omega_{ijk} = 1$$

$$F_{ik} \leq X_{ij} \rightarrow \omega 1_{ijk} = 1$$

❖ $Si\ \omega_{ijk} = 1\ Entonces\ \ F_{ij} - X_{ik} \leq 0$

$$F_{ij} - X_{ik} \leq 0 + \left(CS_{F_{ij}-X_{ik}} - 0\right) * (1 - \omega_{ijk})$$

$$F_{ij} - X_{ik} \leq CS_{F_{ij}-X_{ik}} - CS_{F_{ij}-X_{ik}} * \omega_{ijk}$$

❖ $Si\ \omega 1_{ijk} = 1\ Entonces\ \ F_{ik} - X_{ij} \leq 0$

$$F_{ik} - X_{ij} \leq 0 + \left(CS_{F_{ik}-X_{ij}} - 0\right) * (1 - \omega 1_{ijk})$$

$$F_{ik} - X_{ij} \leq CS_{F_{ik}-X_{ij}} - CS_{F_{ik}-X_{ij}} * \omega 1_{ijk}$$

$$Si\ \alpha_{ij} + \alpha_{ik} \geq 2\ Entonces\ (\omega_{ijk} = 1\ O\ \omega 1_{ijk} = 1)$$

$$Si\ \alpha_{ij} + \alpha_{ik} \geq 2\ Entonces\ (\omega_{ijk} + \omega 1_{ijk} \geq 1)$$

Modelamos

- $\alpha_{ij} + \alpha_{ik} \geq 2 * \omega 2_{ijk} + CI_{\alpha_{ij}+\alpha_{ik}} * (1 - \omega 2_{ijk})$

$$CI_{\alpha_{ij}+\alpha_{ik}} = 0 \rightarrow \alpha_{ij} + \alpha_{ik} \geq 2 * \omega 2_{ijk}$$

- $\alpha_{ij} + \alpha_{ik} \leq (2 - A) * (1 - \omega 2_{ijk}) + CS_{\alpha_{ij}+\alpha_{ik}} * \omega 2_{ijk}$

$$CS_{\alpha_{ij}+\alpha_{ik}} = 2;\ A = 1 \rightarrow \alpha_{ij} + \alpha_{ik} \leq 1 + \omega 2_{ijk}$$

- $\omega_{ijk} + \omega 1_{ijk} \geq 1 * \omega 2_{ijk} + CI_{\omega_{ijk}+\omega 1_{ijk}} * (1 - \omega 2_{ijk})$

$$CI_{\omega_{ijk}+\omega 1_{ijk}} = 0 \rightarrow \omega_{ijk} + \omega 1_{ijk} \geq \omega 2_{ijk}$$

Función Objetivo

(3)

$$Min\ C_{máx}$$

$$C_{máx} \geq C_i \quad \forall_i$$

Calculo de precedencia

(4)

$$\beta_{ijk} \begin{cases} 1\ Si\ el\ trabajo\ \text{"j"}\ es\ precede\ al\ trabajo\ \text{"k"}\ en\ la\ maquina\ i \\ 0\ en\ otro\ caso \end{cases}$$

$$\forall_i \forall_k \quad Si\ X_{ik} \geq 1\ Entonces \sum_{j}^{n} \beta_{ijk} \geq 1$$

Modelamos

- $X_{ik} \geq 1 * \omega 3_{ik} + CI_{X_{ik}} * (1 - \omega 3_{ik})$

$$CI_{X_{ik}} = 0 \rightarrow X_{ik} \geq 1 * \omega 3_{ik}$$

❖ $X_{ik} \leq (1 - A) * (1 - \omega 3_{ik}) + CS_{X_{ik}} * \omega 3_{ik}$

$$A = 1 \rightarrow X_{ik} \leq CS_{X_{ik}} * \omega 3_{ik}$$

❖ $\sum_j^n \beta_{ijk} \geq 1 * \omega 3_{ik} + CI_{\sum_j^n \beta_{ijk}} * (1 - \omega 3_{ik})$

$$CI_{\sum_j^n \beta_{ijk}} = 0 \rightarrow \sum_j^n \beta_{ijk} \geq 1 * \omega 3_{ik}$$

$$\forall_i, \forall_j, \forall_k \ \ si\ \beta_{ijk} = 1\ Entonces\ X_{ik} - X_{ij} = P_{ij}$$

Modelamos

❖ $X_{ik} - X_{ij} \leq P_{ij} + \left(CS_{X_{ik-X_{ij}}} - P_{ij}\right) * (1 - \beta_{ijk})$

$$X_{ik} - X_{ij} \leq CS_{X_{ik-X_{ij}}} - CS_{X_{ik-X_{ij}}} * \beta_{ijk} + P_{ij} * \beta_{ijk}$$

❖ $X_{ik} - X_{ij} \geq P_{ij} * \beta_{ijk} + CI_{X_{ik-X_{ij}}} * (1 - \beta_{ijk})$

$$\forall_i: \ C_i = \sum_{j=1}^{n} F_{ij} + \sum_j^n \sum_k^n S_{ijk} * \beta_{ijk}$$

$$X_{ij} \geq 0, \alpha_{ij} \in \{0,1\}, \ \beta_{ijk} \in \{0,1\}\ , \forall j \in N\ , \forall k \in N, \forall i \in N \quad (5)$$

$F_{ij} = \ finalización\ del\ porceso\ del\ trabajo\ "j"\ en\ la\ maquina"i"$

$CS = Cota\ superior$

$CI = Cota\ inferior$

En este modelo, al igual que el anterior, actúan los actores “Maquias” y “Trabajos”, con la variante del actor “Tiempo” el cual nos indica el ínstate de tiempo en el que cada trabajo “j” es programado.

Con diferencia en el modelo anterior, el conjunto **(1)** lo forman dos actividades de decisión. La primera, una variable continua, la cual nos informa el instante de tiempo en el que se inicia trabajo “j” en una maquina “i”, mientras que la segunda variable es binaria e informa a que máquina es asignado cada trabajo. El conjunto **(2)** lo forman dos especificaciones. La primera especificación es de asignación, es decir, cada trabajo es asignado a una única maquina, mientras que la segunda es una especificación de solape la cual especifica que si dos trabajos “j” y “k”, uno precedente del otro, han sido seleccionados para procesarse en la misma máquina “i”, tiene que cumplirse que el tiempo de terminación del trabajo “j” sea menor igual que el tiempo de comienzo de proceso del trabajo “k”, o viceversa. Para finalizar con el modelo, los conjuntos **(3)**, **(4)** y **(5)** formalizan la función objetivo la cual es, como ya se ha dicho anteriormente, minimizar el tiempo de terminación del último trabajo o C_{max}. Por lo tanto se ha tenido en cuenta la precedencia de cada trabajo, para así de ese modo aplicar el tiempo de setup correspondiente a cada trabajo **Conjunto (4)**. Por último hemos calculado

el tiempo de terminación o "completion time" de cada máquina **(C_i),** mediante la suma de los tiempos de procesos de los trabajos asignados a cada una de las mismas, más los tiempos de setup correspondientes a cada trabajo **Conjunto (5).**

2.2. Implantación de los modelos en LINGO

En este apartado veremos la implantación de cada modelo objetivo en "LINGO". Para ello hemos creado 4 baterías de problemas, cada batería mantiene un número fijo de máquinas, en este caso 5 máquinas, variando el número de trabajos: 10, 15, 20 y 25. Cada batería está formada por, los tiempos de proceso y tiempos de Setup de cada trabajo para cada máquina.

De cada batería obtendremos un Cmax, el cual será el óptimo, y una secuencia de asignación de cada trabajo a cada máquina los cuales los plasmaremos más adelante en graficas de Gantt.

2.2.1 Modelo objetivo 1 LINGO

En este apartado se desarrolla el código lingo con el modelo objetivo 1, para una mejor ilustración utilizaremos los datos de la primera batería de problemas, es decir, utilizando 5 máquinas y 10 trabajos. El desarrollo de las restantes baterías de problemas se realiza de la misma forma, únicamente varia el tamaño del problema.

Iniciamos LINGO con el comando de apertura "MODEL", entres los comandos "SETS" y "ENDSETS" se definen los actores del modelo con sus correspondientes atributos. Esa manera de definir los actores se denominan "Conjuntos" que se distinguen en dos tipos:

- Conjunto primitivo: formado por un cada actor y cada atributo … **Maquinas /1..5/: C;**
- Conjunto derivado: formado por varios conjuntos primitivos y los atributos que comparten entre ellos… **Maq_Tbj (Maquinas, Trabajos): P;**

MODEL:

SETS:

Maquinas /1..5/: C;

Trabajos /1..10/;

Posicion /1..11/;

Precedencia (Maquinas,Trabajos,Trabajos): S, beta;

Maq_Tbj (Maquinas, Trabajos): P;

Asignar (Maquinas, Trabajos, Posicion): alfa;

ENDSETS

Seguidamente declaramos los comandos "DATA" Y "ENDDATA" entre los cuales se definen los valores de algunos atributos, en este caso los tiempos de proceso (P) y los tiempos de Setup (S).

DATA:

P=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4;

S=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, …;

ENDDATA

Luego introducimos el tipo de variables que interviene en el modelo. En lingo por defecto las variables son continuas a no ser que se diga lo contrario : variable entera (**@GIN(X)**), variable binaria (**@BIN(X)**), variable libre (**@FREE(X)**)

Para finalizar declaramos las restricciones a las que está sujeto el modelo, su función objetivo y cerramos con el comando "**end**"

```
!Tipo de variables;

@for(Asignar(i,j,r):@BIN(alfa(i,j,r)));

@for(Precedencia(i,j,k):@BIN(beta(i,j,k)));

!Restricciones;

@for(Trabajos(j):@sum(Maquinas(i):@sum(Posicion(r): alfa(i,j,r))) =1);

@for(Maquinas(i):@for(Posicion(r): @sum(Trabajos(j): alfa(i,j,r)) <=1));

@for(Maquinas(i):@for(Posicion(r)|r#GT#1: @sum(Trabajos(j): alfa(i,j,r)) <=
@sum(Trabajos(j): alfa(i,j,r-1))));

@for(Maquinas (i):@for(Trabajos(j):@for(Trabajos(k)
|k#NE#j:@for(Posicion(r)|r#LT#21: alfa(i,j,r)+alfa(i,k,r+1) <= 1+beta(i,j,k)))));

@for(Maquinas(i): C(i) = @sum(Trabajos(j):@sum(Posicion(r) |r#LT#21:
P(i,j)*alfa(i,j,r))) + @sum(Trabajos(j):@sum(trabajos(k)|k#NE#j: S(i,j,k)*beta(i,j,k))));

MIn=Cmax;

@for(Maquinas(i): Cmax >= C(i));

end
```

Como resultado hemos obtenido un **Cmax** igual a "**10**"

2.2.2 Modelo objetivo 2 LINGO

La implantación del modelo 2 en LINGO con 5 maquinas y 10 trabajos tienen la misma estructura

del modelo anterior, sin embargo al utilizar en este modelo del actor "tiempo" nos genera más restricciones que en el modelo anterior.

Inicialización con el código "MODEL" y declaración de los actores entre la estructura "SET" y ENDSET":

MODEL:

SETS:

Maquinas /1..5/: C;

Trabajos /1..10/;

Precedencia (Maquinas,Trabajos,Trabajos):Setup,beta,W,W1,W2;

Maq_Tbj (Maquinas, Trabajos): P, alfa, X, F,W3;

ENDSETS

Introducción de los datos de la batería de problemas:

DATA:

P=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4;

Setup=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, 7, 7, 9, 5, 9, 3, 2, 5, 2, 9, 7, 8, 2, 8, 8, 8, 9, 5, 9, 5, 6, 2, 6, 7, 7, 6, 1, 2, 6, 7, 1, 9, 8, 1, 6, 9, 2, 4, 6, 8, 9, 10, 9, 8, 4, 1, 5, 9, 2, 5, 3, 3, 4, 10, 5, 4, 3, 10, 2, 10, 2, 4, 8, 9, 1, 7, 10, 5, 10, 9, 9, 5, 10, 7, 1, 5, 7, 5, 2, 8, 9, 10, 2, 5, 7, 1, 2, 8, 9, 10, 3, 2, 7, 3, 1, 2, 4, 8, 6, 7, 4, 4, 5, 2, 3, 6, 5, 6, 5, 7, 3, 1, 4, 9, 4, 1, 3, 1, 3, 3, 8, 9, 5, 7, 8, 5, 6, 8, 8, 2, 7,...;

CSx= 15000;

CSf=15000;

CIx=-15000;

ENDDATA

Declaración de las variables y restricciones del modelo

```
!Tipo de variables;

@for(Maq_Tbj(i,j):@BIN(alfa(i,j)));

@for(Precedencia(i,j,k):@BIN(beta(i,j,k)));

@for(Precedencia (i,j,k):@BIN(w(i,j,k)));

@for(Precedencia (i,j,k):@BIN(w1(i,j,k)));

@for(Precedencia (i,j,k):@BIN(w2(i,j,k)));

@for(Maq_Tbj(i,j):@BIN(W3(i,j)));

!Restricciones;

@for(Maquinas(i):@for(Trabajos(j): X(i,j) <= CSx*alfa(i,j)));

@for(Trabajos(j):@sum(Maquinas(i):alfa(i,j))=1);

@for(Maquinas(i):@for(Trabajos(j): F(i,j) >= X(i,j)+P(i,j)*alfa(i,j)));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: F(i,j) - X(i,k)<= CSf-
CSf*W(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: F(i,k) - X(i,j) <= CSf-
CSf*W1(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: alfa(i,j) + alfa(i,k) >=
2*W2(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: alfa(i,j) + alfa(i,k) <=
1+W2(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: W(i,j,k) + W1(i,j,k) >=
W2(i,j,k))));

@for(Maquinas(i):@for(Trabajos(k): X(i,k) >= W3(i,k)));

@for(Maquinas(i):@for(Trabajos(k): X(i,k) <= CSx*W3(i,k)));

@for(Maquinas(i):@for(Trabajos(k):@sum(Trabajos(j): beta(i,j,k)) >= W3(i,k)));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: X(i,k)-X(i,j) <=  CSx*(1-
beta(i,j,k))+P(i,j)*beta(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j:X(i,k)-X(i,j) >= P(i,j)*
beta(i,j,k)+ CIx*(1-beta(i,j,k)))));

@for(Maquinas(i): C(i) = @sum(Trabajos(j):F(i,j)) + @sum(Trabajos(j):
@sum(Trabajos(k): Setup(i,j,k)*beta(i,j,k))));
```

Para finalizar declaramos la función objetivo y cerramos la estructura con el código “end”:

```
Min=Cmax;
@for(Maquinas(i): Cmax >= C(i));
end
```

Como es de esperar nos da un resultado de un **Cmax** igual a “**10**” igual a la del modelo objetivo 1

3 HEURÍSTICA DE RESOLUCIÓN

En este capítulo desarrollaremos y explicaremos la heurística de resolución formado mediante lenguaje C++. Así mismo se implementaran las secuencias de cada batería de problemas que se aplicaron en la aplicación LINGO y se creara una tabla de resultados. La idea básica de la heurística se centra en la asignación de cada trabajo a la máquina menos cargada. Para dicha asignación hemos creado una secuencia aplicando la regla de despacho LPT. Es decir, se han ordenado los trabajos de forma decreciente en función de sus tiempos de procesos en cada máquina, por lo tanto los primero trabajos en procesarse serán los que tiene un tiempo de proceso mayor.

3.1. Desarrollo de la heurística

Para la creación del algoritmo hemos utilizado diferentes librerías, que facilitan su desarrollo. Las principales son:

- "**stdio.h**": contiene las definiciones de las macros, las constantes, las declaraciones de **funciones de la biblioteca** estándar del lenguaje de programación C para hacer operaciones, estándar, de entrada y salida.
- "**stdlib.h**": Contiene los prototipos **de** funciones **de** C **para** gestión **de** memoria dinámica, control **de** procesos y otras funciones.
- "**Schedule_lib.h** : contiene las funciones necesarias para poder trabajar con "n" maquinas, creación de matrices, lectura de datos…etc

A continuación veremos el desarrollo paso a paso del algoritmo:

```
#include <stdio.h>
#include <stdlib.h>                                                    (1)
#include <schedule_lib.h>
```

```
int cmx(int n,int m, MAT_INT pij,  VECTOR_INT sec, MAT_INT sjk);       (2)
```

```
void ordenDecrecienteTiemposSM(int numeroTrabajos,int
ordenTrabajos[],int tiemposProceso[]);
```

El conjunto (1) hace referencia a las librerías necesarias, como se ha explicado anteriormente, para realizar todas las operaciones imprescindibles y así obtener un resultado. El conjunto (2) se trata de dos funciones, la primera "int" función se utiliza para calcular $\mathbf{C_{max}}$, en el se introduce en número de

trabajos (**n**), el numero de máquinas (**m**), los tiempos de proceso ($\mathbf{p_{ij}}$), la secuencia en la que hay que procesar los trabajos (**sec**) y los tiempos de setup (**sjk**), la segunda función "void" se utiliza para la creación de la secuencia, siguiendo la regla de despacho LPT, en función de los tiempos de proceso de cada trabajo "j" en cada máquina "i".

```
int main()                                                    (3)
{
int n=10;
int m=5;                                                      (3.1)
int semilla=12;
int j;
int i;
int sec[n];
int sumaTiemP[n];
printf("Secuencia:");
print_int_vector(sec,n);

// Tiempos de proceso                                         (3.2)
  MAT_INT pij=DIM_MAT_INT(m,n);
srand(semilla);
for(i=0;i<m;i++)
{
   for(j=0;j<n;j++)
   {
      pij[i][j]=rand()%10+1;
       sec[j]=j;
   }

}
printf("\nPij:\n");
print_int_matrix(pij,m,n);
```

El conjunto (3) es una función en la que declaramos el conjunto de instrucciones para calcular el objetivo deseado. En el declaramos los parámetro enteros (3.1), que son el numero de maquinas (**m**) y trabajos(**n**), mediante la instrucción "**int**", así mismo declararemos las variables entera "**i**" y "**j**" las cual utilizaremos para generar los tiempos de proceso de cada trabajo en cada máquina (**pij**), así como sus tiempos de setup (**sjk**). Estos datos los hemos utilizado para la implantación en **LINGO** el cual nos ha proporcionado una secuencia la cual la hemos implantado en el comando (**sec**).

El conjunto (3.2) genera los tiempos de proceso de cada trabajo en cada máquina. Utilizando la función "rand" podemos generar números aleatorios con valores entre un intervalo, en este caso, entre 1 y 10. Para que genere siempre los mismo tiempos de procesos independientes del numero de maquinas o de trabajos, hemos utilizado la función "srand" igual al valor de una semilla previamente declarada.

// tiempos de setup **(3.3)**

```
 MAT_INT sjk=DIM_MAT_INT(m*n,n);
srand(semilla);
for(i=0;i<m*n;i++)
{
   for(j=0;j<n;j++)
   {
      sjk[i][j]=rand()%10+1;
      }
}
printf("\nSjk:\n");
print_int_matrix(sjk,m*n,n);
```

//Calculo de cmax **(3.4)**

```
int r1=cmx(n,m,pij,sec,sjk);
printf("\ncmax:%d",r1);
```

//Liberacion de las librerias **(3.5)**

```
FREE_MAT_INT(pij, m);
FREE_MAT_INT(sjk, m);
```

El conjunto (3.3) es la generación de los tiempos de setup. Los tiempos de setup de la máquina *i* para el trabajo *j* precedido por el trabajo *k*, es tridimensional, en este caso usaremos una matriz bidimensional. Para ello el número de filas será igual al número de máquinas por el número de trabajos que hay (MxN). Con esto conseguimos que cada "*M*" filas correspondan a los tiempos de setup de un trabajo en cada máquina, y cada columna corresponde al trabajo que le precede. Los valores que corresponden al trabajo *j* precedido por el trabajo *k*.

A continuación declaramos un entero "**r**", apartado (3.4), el cual recogerá el resultado de la función **cmx** y seguidamente imprimiremos el valor de "**r**" por pantalla utilizando el comando ***printf***. Para finalizar en el conjunto (3.5) liberamos el comando que genera la matriz de tiempos de procesos "$\mathbf{p_{ij}}$" y la matriz de tiempos de setup "**sjk**".

```
//llamada de la  función para el cálculo de cmx                    (4)
int cmx(int n, int m, MAT_INT pij,  VECTOR_INT sec, MAT_INT sjk)


{                                                                    (4.1)
int j;
int i;
int k1;
int k2;

VECTOR_INT maq=DIM_VECTOR_INT(m);                                    (4.2)
setval_Ivector(maq,m,0);
VECTOR_INT maq_aux=DIM_VECTOR_INT(m);
VECTOR_INT pos=DIM_VECTOR_INT(m);
setval_Ivector(pos,m,0);
VECTOR_INT pos_aux=DIM_VECTOR_INT(m);
```

A continuación llamamos a la función "**cmx**", código (4), y estructuramos las instrucciones de la misma la cual se encargará de calcular la función objetivo. En el conjunto (4.1) se ha declarado nuevas variables enteras, (i,j,k,k1), necesarias para el cálculo del objetivo.

El conjunto (4.2) es la creación del vector "maq" mediante el comando "VECTOR INT" con un tamaño igual a "m" e inicializado todos sus valores igual a cero mediante el comando "setval_Ivector". Este vector

"maq" almacenará el completion time de la maquina asignada, mientras que del mismo modo hemos creado el vector "pos" el cual almacenará la suma de la secuencia de cada máquina, es decir por ejemplo si la maquina 1 tiene asignados los trabajos 1 y 2 "pos=3". Además se ha creado los vectores "maq_aux" y "pos_aux" los cuales almacenan e informan la maquina seleccionada para asignar el trabajo "j".

```
for(j=0;j<n;j++)                                                     (4.3)
```

```
{
copyIVector(maq,maq_aux,m);
sortIVector(maq_aux,m,'A');
printf("\n Maquina:%d",maq_aux[0]);
printf("\n");
pos_aux[0]=maq_aux[0];
pos[pos_aux[0]]=pos[pos_aux[0]]+sec[j];
printf("\n");
k1=pos[pos_aux[0]];
```

En el conjunto (4.3) implantamos la metodología para asignar los trabajos "j" a la maquina menos cargada. Para dicho fin se ha creado un bucle "for" el cual recorre todos los trabajos "j". En el hemos copiado todos los valores del vector "maq", al vector "maq_aux" mediante el comando "copyIVector", así mismo se ha ordenando los valores del vector "maq_aux" de forma ascendente mediante el comando "setval_Ivector". De esta forma podemos saber cuál es la maquina menos cargada y asignar el trabajo "j".

```
if (maq[maq_aux[0]]==0)                                    (4.4)
{

maq[maq_aux[0]]=maq[maq_aux[0]]+pij[maq_aux[0]][sec[j]];
printf("\n");
print_int_vector(maq,m);

}
else
{
K2=k1-sec[j];
maq[maq_aux[0]]=maq[maq_aux[0]]+pij[maq_aux[0]][sec[j]]+sjk[maq_a
ux[0]+m*sec[j]][k2];
printf("\n");
print_int_vector(maq,m);
// Actualizacion
```

```
if(pos[pos_aux[0]]<1000000000)
{
   pos[pos_aux[0]]=sec[j];
   k1=pos[pos_aux[0]];
   printf("\n k1 actualizado:%d",k1);
}
}
```

Como sabemos que los primeros trabajos asignados no tienen precedentes, por lo tanto no les imponen tiempos de setup, se ha creado una condición "if" (conjunto (4.4)) dentro del bucle "for" y es que si el completion time de la máquina elegida es igual a cero, a dicha maquina solo se le implantara el tiempo de proceso del trabajos asignado a la misma. En caso contrario "else" a dicha maquina se le implantara el tiempo de proceso del trabajo "j" asignado más el tiempo de setup del trabajo precedente "k" Para saber el trabajo precedente "k" imponemos que "pos_aux[0] = maq_aux[0];"(conjunto (4.3)) de esta manera el vector "pos_aux" almacena que máquina es la menos cargada. Seguidamente almacenamos la suma de la secuencia de dicha maquina en el vector "pos" con ayuda del vector "pos_aux" mediante el comando "pos[pos_aux[0]] = pos[pos_aux[0]]+sec[j];". Dicha suma la almacenamos en una variable "k1" previamente declarada "k1= pos[pos_aux[0]]. Para finalizar con el cálculo del trabajo precedente "k" imponemos que "k2", variable previamente declarada, sea igual a "k1" menos la secuencia del último trabajo "j" asignado en ese momento a la maquina "i" "k2=k1-sec[j]" (conjunto (4.4)). De esta forma si por ejemplo en la maquina 1 se han signado los trabajos 3 y 8, el trabajo 3 es precedente del trabajo 8 el cual es el último trabajo asignado, por lo tanto "k1 =3+8" y "k2=3+8-8" lo que nos da "k2=3" el trabajo precedente del trabajo 8.

Como se ha dicho con anterioridad los tiempos de setup están compuestos en una matriz con filas igual a (MxN). Donde cada "*M*" filas correspondan a los tiempos de setup de un trabajo en cada máquina y cada columna corresponde al trabajo que le precede. Para explorara cada fila y columna dependiendo de la maquina seleccionada y el trabajo precedente al trabajo que se le asignará en dicha maquina, hemos utilizado la siguiente expresión "[maq_aux[0]+m*sec[j]][k2]" donde "maq_aux[0]+m*sec[j]" recorrerá la fila de la matriz en busca de la maquina seleccionada para la asignación del trabajo "j" y "[k2]" el trabajo precedente de "j" (conjunto(4.4)). Seguidamente actualizamos el vector "pos", de esta forma conseguimos que en dicho vector se almacene la suma de la secuencia de los dos últimos trabajos "j" asignados a la maquina "i".

```
int cmax=0;                                                    (5)

      j=0;
for(j=0;j<m;j++)
      {
        if(maq[j]>cmax)
        {
```

```
            cmax=maq[j];
        }
    }

  return cmax;

}
```

Para finalizar en la función que calcula el makespan declaramos "**cmax**"(conjunto(5)), variable que almacenará el total completion time de la máquina más cargada, para ello imponemos inicialmente que cmax sea igual a cero, y mediante un bucle "for" y una condición "if", con cada iteración, se actualiza con el valor más alto de cmax obtenido.

```
//llamada de la  función secuencia                    (6)

int i,j;                                               (6.1)
int posicionMaxima,maximo;
int tiemposProcesoAuxiliar[numeroTrabajos];

for(i=0;i<numeroTrabajos;i++)                          (6.2)
  {
    tiemposProcesoAuxiliar[i]=tiemposProceso[i];
  }
for(i=0;i<numeroTrabajos;i++)                          (6.3)
  {
    //Determinamos el i-esimo menor tiempo de proceso
    int a=1;
    for(j=0;j<numeroTrabajos;j++)
    {
      if(tiemposProcesoAuxiliar[j]!=-1)
      {
        if(a==1)
        {
```

```
            a=0;
            posicionMaxima=j;
            maximo=tiemposProcesoAuxiliar[j];
          }
          else if(tiemposProcesoAuxiliar[j]>maximo)
          {
            posicionMaxima=j;
            maximo=tiemposProcesoAuxiliar[j];
          }
        }
      }
      ordenTrabajos[i]
//llamada de la  función para el cálculo de cmx=posicionMaxima;
      tiemposProcesoAuxiliar[posicionMaxima]=-1;
   }
}
```

Para finalizar con la heurística llamamos a la función secuencia, la cual se encarga de ordenar los trabajos de forma decreciente en función de sus tiempos de procesos según la regla de despacho LPT.

En el conjunto (6.1) declaramos las variables necesarias para el cálculo como la **posición máxima**, la cual guarda la posición del trabajo con mayor tiempo de proceso; **máximo**, el cual almacena el tiempo de proceso máximo y **tiemposProcesoAuxiliar**, el cual almacena los tiempos de proceso. El conjunto (6.2) se trata de un bucle "for" en el cual se impone que en la variable **tiemposProcesoAuxiliar** almacene los tiempos de procesos de cada trabajo.

Para finalizar en el conjunto (6.3) se crea una bucle "for" dentro de otro bucle "for", con esto conseguimos que la heurística recorra los tiempos de proceso de cada trabajo en cada máquina. Seguidamente declaramos una variable entera denominada "**a**" e imponemos que su valor inicial sea 1, una vez hecho esto imponemos dos condiciones "if" las cuales establecen que si **tiemposProcesoAuxiliar** es distinto que -1 y si "**a**" es igual a 1, "**a**" se actualiza igual a 0, "**posicionMaxima"** es igual al trabajo "j" de esa iteración y "**máximo"** es igual al tiempo de proceso de dicho trabajo "j". Si no se cumplen dichas condiciones pasamos a la siguiente condición establecida por el comando "**else if**", es decir, al no cumplirse que **tiemposProcesoAuxiliar** sea distinto que –1 y que "**máximo**" sea igual a 1 entonces si **tiemposProcesoAuxiliar** es mayor que "**máximo**" actualizamos la posición y máximo. Para finalizar la heurística declaramos la llamada para el cálculo del cmax con la secuencia obtenida mediante este proceso.

Veamos un ejemplo de aplicación con 2 máquinas y 4 trabajos para ver mejor la asignación de los trabajos a la máquina menos cargada.

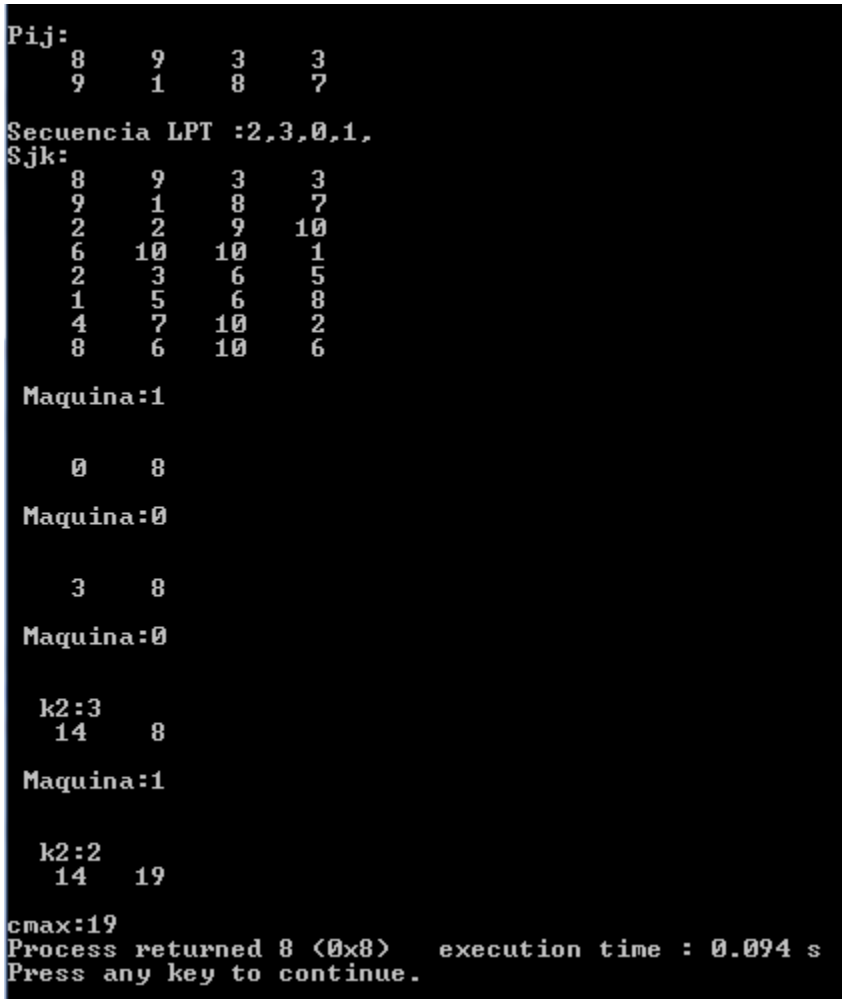


Imagen 8. Impresión por pantalla ejemplo m=2; n=4

En la imagen 8 podemos ver la implementación del ejemplo en la heurística, en ella se aprecia una secuencia LPT, los tiempos de proceso "P_{ij}"de cada trabajo en cada máquina, los tiempos de Setup "S_{jk}" y el proceso de asignación de los trabajos a cada máquina.

Los tiempos de proceso "P_{ij}" se ordenan la siguiente forma; la primera fila corresponde a la primera máquina, la segunda fila a la segunda maquina y así sucesivamente hasta "m" máquinas.

Los tiempos de Setup (S_{jk}) se ordenan de la siguiente formas: las primeras dos filas corresponden a los tiempos de setup del trabajo "1" en la maquina 0 y 1 sucesivamente; las siguientes dos al trabajo 2…y así sucesivamente. Ver imagen 9

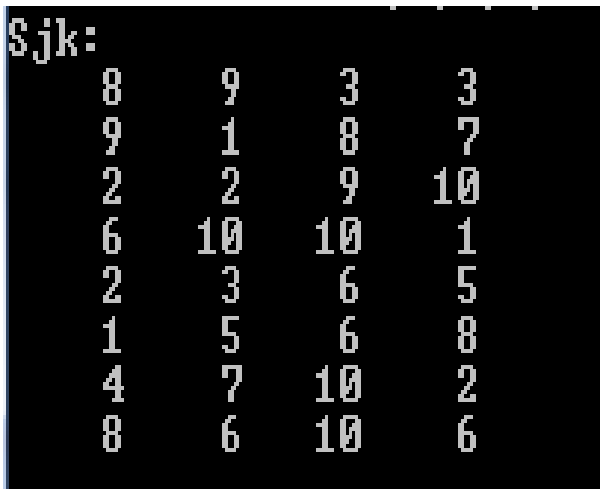


Imagen 9. Matriz Setup

Si nos encontramos en la situación de 5 máquinas, las primeras 5 filas de la matriz de los tiempos de setup corresponden al trabajo 1 en cada máquina, las siguientes 5 al trabajo 2 y así sucesivamente para “m” maquinas y “n” trabajos.

Para finalizar se realiza la asignación de los trabajos a las máquinas siguiendo la secuencia asignada dándonos como resultado el mejor Cmax encontrado y el tiempo que ha tardado la heurística en encontrarlo. Ver imagen 8

4 RESULTADOS

En este capítulo vamos a estudiar los resultados obtenidos de cada modelo. Como se ha dicho anteriormente los datos son el resultado de la creación de modelos matemáticos y la implantación de dichos modelos en la aplicación LINGO, así mismo vamos a comparar los resultados obtenidos en LINGO con los obtenidos en la heurística que hemos desarrollado. Los resultados a analizar van a ser el tiempo que tarda la heurística en obtener la solución del makespan, cuyo valor nos aportará, mediante el cálculo PDR, el cual explicaremos con más detalle más adelante, lo cerca que esta dicho valor del óptimo. Como se ha dicho anteriormente se va a analizar los resultados de las 4 baterías de problemas (anexo), se ilustraran diagramas de Gantt con la asignación dada por los modelos matemáticos y se generara una trabal de resultados para la comparación de los modelos con la heurística de cada batería de problemas.

4.1. Tablas de resultados

En este apartado podremos visualizar los resultados obtenidos por los modelos matemáticos:

En la siguiente figura podemos observar la distribución de los trabajos en las maquinas, dando como resultado un cmax de 10, el cual es el óptimo.

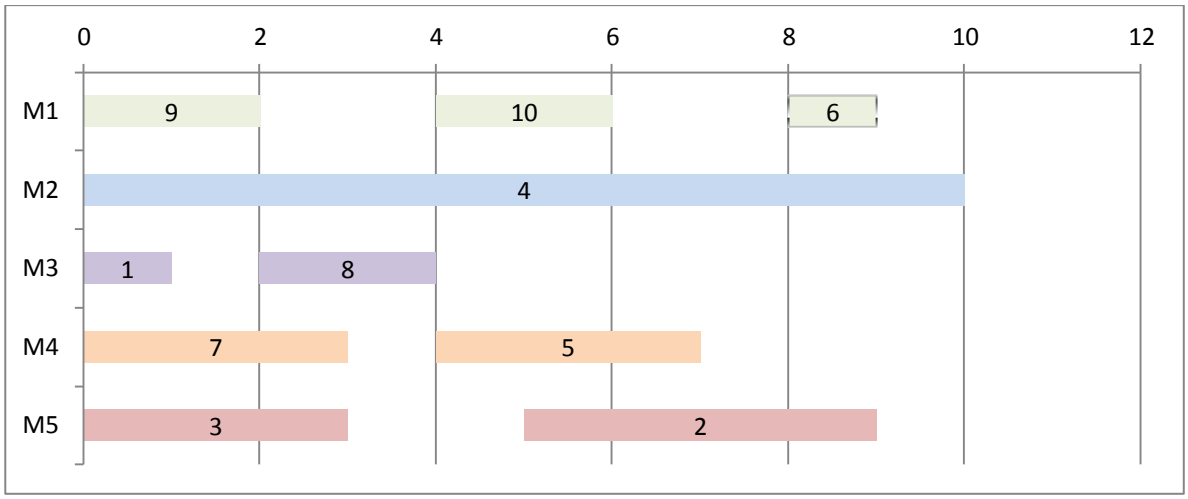


Figura 10. Solución m=5 y n=10

En la siguiente figura podemos observar la distribución de los trabajos en las maquinas, dando como resultado un cmax de 16, el cual es el óptimo.

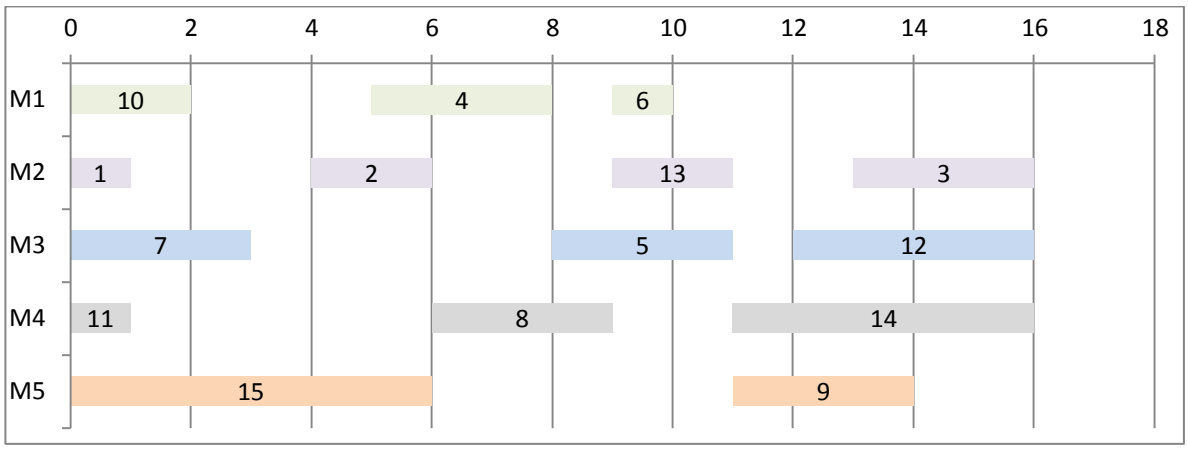


Figura 11. Solución m=5 y n=15

En la siguiente figura podemos observar la distribución de los trabajos en las maquinas, dando como resultado un cmax de 25, el cual es el óptimo.

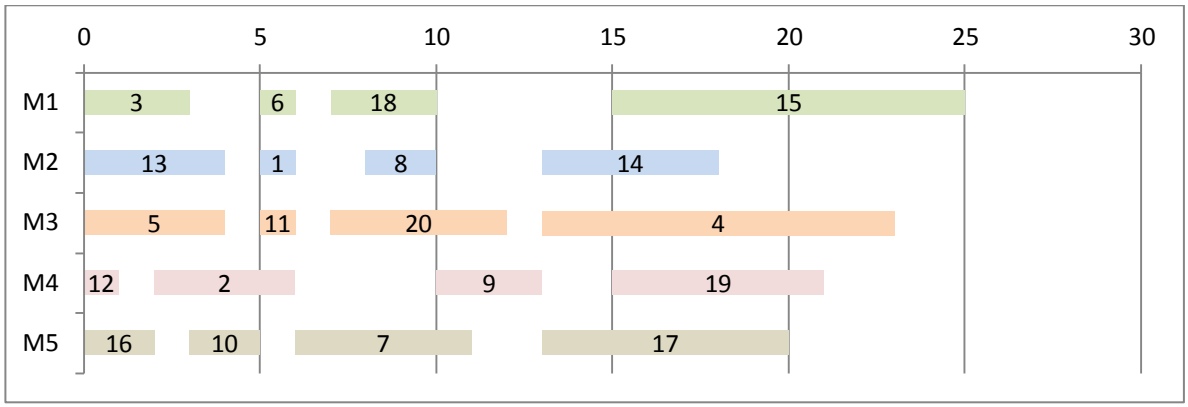


Figura 12. Solución m=5 y n=20

En la siguiente figura podemos observar la distribución de los trabajos en las maquinas, dando como resultado un cmax de 38, el cual es el óptimo.

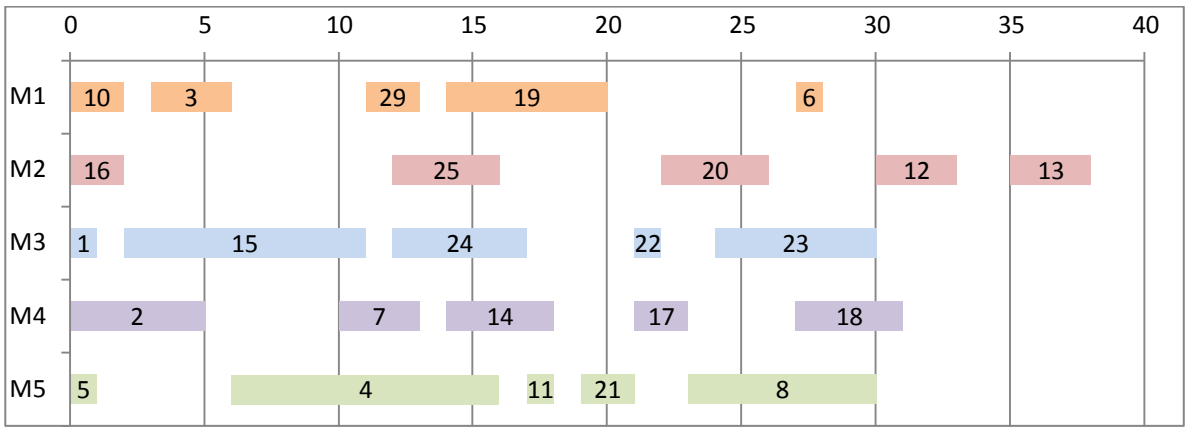


Figura 13. Solución m=5 y n=25

En la siguiente **tabla 4** podemos ver los resultados obtenidos en la heuritica aplicada a cada batería de problemas:

m / n	5	
	Cmax	Tiempo (seg)
10	15	0,5645
15	30	0,751
20	48	1,248
25	78	1,994

Tabla 4. Resultados heurística

4.2. Cálculo PDR

Para comprobar de manera exacta la desviación que tienen los resultados obtenidos mediante la heurística respecto al óptimo obtenido mediante los modelos, vamos a utilizar el porcentaje de desviación relativa (PDR) (Martí, 2010) que se desarrolla por la siguiente expresión:

$$PDR = \frac{valor\ heurística - valor\ óptimo}{valor\ óptimo} \times 100$$

n \ m	5					
	Tiempos Modelo1	Tiempos Modelo 2	Tiempos Heurística	Cmax modelos	Cmax Heurística	PDR Medio
10	9,97 (min)	22,15 (min)	0,5645 (seg)	10	15	50%
15	5,15 (h)	8,23 (h)	0,751 (seg)	16	30	87.5%
20	5,5 (h)	12,4 (h)	1,248 (seg)	25	48	92%
25	19,89 (h)	27,8 (h)	1,994 (seg)	38	78	105,26%%
Total						83.69%

Tabla 5. Resultado tiempos modelo1, tiempos modelo 2, cmax modelos, cmax heurística y PDR

En la tabla 5 podemos observar que la heurística que hemos desarrollado se aleja del óptimo cuando aumentan el número de trabajos. Así mismo podemos observar el tiempo medio en llegar al óptimo para cada batería de problemas. Tanto en el modelo 1 como en el modelo 2 se observa que los tiempos crecen de forma exponencial conforme crecen el número de trabajos. Sin embargo en la heurística observamos que tenemos una respuesta mucho más rápida.

En la siguiente gráfica "Figura 13" podemos visualizar de forma más clara la diferencia de tiempo de computo entre los dos modelos y de la heurística. Como se menciono con anterioridad en ella podemos observar que la heurística nos proporciona soluciones en tiempos más cortos que los dos modelos desarrollados.

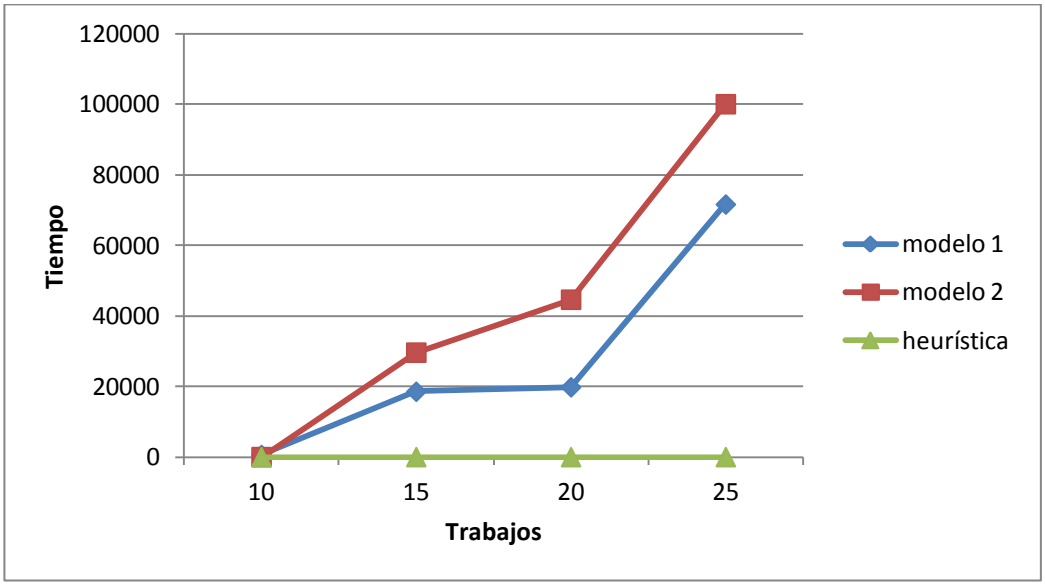


Figura 13. Diferencia de tiempos de cómputo

4.3. Conclusiones

En este trabajo fin de grado se ha estudiado el problema de máquinas paralelas no relacionadas con tiempos de setup dependientes de la secuencia. Para poder llevarlo a cabo se ha estudiado el problema mediante dos modelos matemático diferente y una heurística, desarrollados con los conocimientos obtenidos en "Investigación operativa (modelado)" y " programación y control de la producción" respectivamente. La estructura del trabajo fin de grado se compone en un principio de unos capítulos en los que se explica brevemente en que se fundamenta tanto la "investigación operativa" como la" programación y control de la producción".

Una vez finalizado el capítulo teóricos, seguimos con el capitulo 2 en donde describimos las características del problemas que se aborda. Desarrollamos los dos modelos matemáticos para su resolución del problema e implantamos dichos modelos en la aplicación "LINGO". Así mismo explicamos paso a paso el desarrollo tanto de los modelos matemáticos como del código de la aplicación LINGO.

Finalizado el capitulo 2, seguimos con el capitulo 3 donde desarrollamos la heurística. Para ello tomamos como base principal la asignación de cada trabajo a la máquina menos cargada y la creación de una secuencia siguiendo la regla de despacho LPT. Así mismo, como en el capitulo anterior, el desarrollo de la heurística y su código es explicado paso a paso.

Para finalizar en el capítulo 4 se visualiza los resultados tanto de los dos modelos como de la heurística. Hemos observado que los modelos dan los óptimos del problema que se aborda, sin embargo el tiempo de computo que se invierte en obtener dicho resultado es enorme, por dicha razón hemos reducido las baterías de problemas a 5 máquinas y 25 trabajos como mucho, ya que la ser un numero de trabajos superior, la inversión de tiempo de computo seria de días. Por otro lado la heurística nos da resultados con tiempos de cómputo mínimos, sin embargo los resultados obtenidos se desvían del óptimo una media de 84%, lo cual nos da la conclusión que la heurística se desvía bastante del óptimo. Sabemos que existen distintas heurísticas, como MILP, las cuales dan un resultado más cercano al optimo, sin embargo la inspiración que hemos tenido a la hora de realizar este trabajo es la de desarrollar nuestra propia heurística aplicando los conocimientos obtenidos a lo largo de la carrera.

Para finalizar con la conclusión, decir que el desarrollo de los modelos, su implantación en LINGO y el desarrollo de la heurística no ha sido un trabajo fácil, ya que ha requerido de muchas modificaciones, detección de errores y pruebas hasta conseguir tanto los modelos como la heurística necesaria para resolver el problema. Con todos estos datos podemos concluir que para problemas grandes los modelos no son capaces de dar los óptimos sin tener una gran inversión en tiempo de cómputo, en cambio la heurística es una buena forma de obtener un resultado en un tiempo más razonable.

REFERENCIAS

1. Al-Salem, A. (2004). *Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times*. Engineering Journal of the University of Qatar, Vol. 17, pp 177-188.

2. Anagnostopoulos, G. & Rabadi, G. (2002). *A simulated annealing algorithm for the unrelated parallel machine scheduling problem*. In: Proceedings of the World Automation Congress 2002, Orlando, FL, June 09-13
.
3. Cheng, T. & Sin, C. (1990). *A state-of-the Art review of parallel-machine scheduling research*. European Journal of Operation Research, 47, 271-292.

4. Dhaenens-Flipo, C. (2001). *A bicriterion approach to deal with a constrained singleobjective problem*. International Journal of Production Economics, Vol. 74, 93-101.

5. Franca, P., Gendreau, M., Laporte, G. & Müller, F. (1996). *A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times*. International Journal of Production Economics, Vol. 43, 79-89.

6. 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*. Annals of discrete mathematics.

7. Graves, S.C. (1981). *A review of Production Scheduling*, Operation Research, 29, 646- 675.

8. Guinet, A. (1993). *Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria*. International Journal of Production Research, Vol. 31, Nº7, 1579-1594.

9. Guinet, A. (1991). *Textile production systems: a succession of non-identical parallel processor shops*. Journal of the Operational Research Society, Vol. 42, Nº 8, 655-671.

10. Helal, M. & Hosni, Y. (2003). *A Tabu Search Approach for the Non-identical Parallel-Machines Scheduling Problem With Sequence-Dependent Setup Times*. In: Proceedings of the 2003 Industrial Engineering Research Conference, Portland, OR, May 18-21.

11.[Arango, Giraldo, & Castrillón, 2013] Arango, J. a., Giraldo, J. a., & Castrillón, O. D. (2013). *Programación de máquinas paralelas no relacionadas con tiempos de montaje dependientes de la secuencia y entrada dinámica usando algoritmos genéticos.*Información Tecnológica, Vol. 24 (3), 73-84.

12.[Companys Pascual, 1989] Companys Pascual, R. (1989). *Planificación y programación de la producción.*Marcombo-Boixerau.

13.[Martí, 2010] Martí, Rafael (2010). *Algoritmos Heurísticos en Optimización Combinatoria.*Departamento de Estadística e Investigación Operativa. Facultad de Ciencias Matemáticas. Universidad de Valencia.

14.[Framiñan, Leisten, & Ruiz García, 2014] Framinan, J. M., Leisten, R., & Ruiz García, R. (2014). *Manufacturing Scheduling System : An Integrated View on Models, Methods and Tools*. Springer

15.[Gantt, 1910] Gantt, H. (1910). *Work, Wages and Profit. Their influence on the cost of living.*The Engineering Magazine, New York.

16.[Miranda Zambrana, 2010] Miranda Zambrana, A. (2010). *Diseño del Software de control de instrumento HP4142B para caracterización de dispositivos.*Proyecto Fin de Carrera. Universidad Politécnica de Cataluña.

17.[Pérez González, 2014] Pérez González, P. (2014). *Programación y Control de la Producción.*
Universidad de Sevilla.

Anexo

Batería de problemas

5 maquinas y 10 trabajos:

P_{ij} =8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4,

S_{ijk} =8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, 7, 7, 9, 5, 9, 3, 2, 5, 2, 9, 7, 8, 2, 8, 8, 8, 9, 5, 9, 5, 6, 2, 6, 7, 7, 6, 1, 2, 6, 7, 1, 9, 8, 1, 6, 9, 2, 4, 6, 8, 9, 10, 9, 8, 4, 1, 5, 9, 2, 5, 3, 3, 4, 10, 5, 4, 3, 10, 2, 10, 2, 4, 8, 9, 1, 7, 10, 5, 10, 9, 9, 5, 10, 7, 1, 5, 7, 5, 2, 8, 9, 10, 2, 5, 7, 1, 2, 8, 9, 10, 3, 2, 7, 3, 1, 2, 4, 8, 6, 7, 4, 4, 5, 2, 3, 6, 5, 6, 5, 7, 3, 1, 4, 9, 4, 1, 3, 1, 3, 3, 8, 9, 5, 7, 8, 5, 6, 8, 8, 2, 7, 5, 9, 4, 5, 8, 8, 8, 1, 7, 7, 6, 3, 9, 9, 7, 5, 6, 5, 6, 2, 10, 2, 8, 5, 5, 2, 3, 2, 2, 10, 8, 5, 3, 7, 7, 10, 10, 1, 10, 6, 6, 8, 6, 3, 6, 4, 9, 6, 4, 1, 7, 2, 9, 10, 7, 10, 3, 7, 6, 10, 6, 4, 10, 10, 4, 8, 6, 9, 4, 10, 9, 3, 7, 10, 6, 3, 9, 7, 1, 4, 2, 6, 10, 10, 5, 2, 1, 4, 9, 9, 5, 2, 9, 8, 9, 5, 8, 8, 2, 4, 1, 8, 10, 5, 10, 5, 2, 9, 2, 4, 2, 3, 6, 10, 8, 4, 9, 4, 5, 7, 1, 8, 7, 7, 8, 4, 6, 9, 6, 1, 8, 4, 5, 2, 8, 4, 9, 4, 8, 2, 8, 5, 6, 2, 5, 5, 3, 9, 7, 5, 8, 10, 8, 10, 4, 2, 5, 2, 9, 2, 3, 6, 5, 3, 4, 8, 8, 6, 2, 7, 5, 9, 5, 9, 6, 6, 3, 4, 4, 8, 7, 5, 3, 8, 8, 5, 7, 9, 8, 5, 5, 7, 9, 1, 6, 8, 4, 2, 4, 8, 3, 10, 6, 5, 7, 5, 2, 7, 10, 7, 4, 9, 7, 2, 9, 8, 10, 3, 4, 5, 1, 10, 2, 7, 6, 4, 8, 8, 7, 9, 5, 3, 5, 2, 1, 7,

Batería de problemas

5 maquinas y 15 trabajos :

P_{ij}=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6,

S_{ijk}=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, 7, 7, 9, 5, 9, 3, 2, 5, 2, 9, 7, 8, 2, 8, 8, 8, 9, 5, 9, 5, 6, 2, 6, 7, 7, 6, 1, 2, 6, 7, 1, 9, 8, 1, 6, 9, 2, 4, 6, 8, 9, 10, 9, 8, 4, 1, 5, 9, 2, 5, 3, 3, 4, 10, 5, 4, 3, 10, 2, 10, 2, 4, 8, 9, 1, 7, 10, 5, 10, 9, 9, 5, 10, 7, 1, 5, 7, 5, 2, 8, 9, 10, 2, 5, 7, 1, 2, 8, 9, 10, 3, 2, 7, 3, 1, 2, 4, 8, 6, 7, 4, 4, 5, 2, 3, 6, 5, 6, 5, 7, 3, 1, 4, 9, 4, 1, 3, 1, 3, 3, 8, 9, 5, 7, 8, 5, 6, 8, 8, 2, 7, 5, 9, 4, 5, 8, 8, 8, 1, 7, 7, 6, 3, 9, 9, 7, 5, 6, 5, 6, 2, 10, 2, 8, 5, 5, 2, 3, 2, 2, 10, 8, 5, 3, 7, 7, 10, 10, 1, 10, 6, 6, 8, 6, 3, 6, 4, 9, 6, 4, 1, 7, 2, 9, 10, 7, 10, 3, 7, 6, 10, 6, 4, 10, 10, 4, 8, 6, 9, 4, 10, 9, 3, 7, 10, 6, 3, 9, 7, 1, 4, 2, 6, 10, 10, 5, 2, 1, 4, 9, 9, 5, 2, 9, 8, 9, 5, 8, 8, 2, 4, 1, 8, 10, 5, 10, 5, 2, 9, 2, 4, 2, 3, 6, 10, 8, 4, 9, 4, 5, 7, 1, 8, 7, 7, 8, 4, 6, 9, 6, 1, 8, 4, 5, 2, 8, 4, 9, 4, 8, 2, 8, 5, 6, 2, 5, 5, 3, 9, 7, 5, 8, 10, 8, 10, 4, 2, 5, 2, 9, 2, 3, 6, 5, 3, 4, 8, 8, 6, 2, 7, 5, 9, 5, 9, 6, 6, 3, 4, 4, 8, 7, 5, 3, 8, 8, 5, 7, 9, 8, 5, 5, 7, 9, 1, 6, 8, 4, 2, 4, 8, 3, 10, 6, 5, 7, 5, 2, 7, 10, 7, 4, 9, 7, 2, 9, 8, 10, 3, 4, 5, 1, 10, 2, 7, 6, 4, 8, 8, 7, 9, 5, 3, 5, 2, 1, 7, 5, 2, 9, 2, 2, 10, 6, 3, 7, 5, 5, 10, 8, 10, 2, 5, 8, 7, 9, 2, 5, 1, 9, 5, 9, 1, 6, 5, 2, 5, 6, 3, 5, 8, 2, 6, 3, 2, 5, 5, 10, 6, 1, 6, 5, 8, 3, 3, 8, 2, 10, 5, 7, 3, 8, 4, 9, 10, 2, 1, 2, 7, 4, 3, 2, 6, 5, 9, 10, 3, 2, 9, 7, 3, 2, 6, 3, 7, 10, 10, 3, 10, 8, 7, 2, 6, 2, 3, 7, 9, 8, 10, 1, 7, 1, 6, 8, 4, 5, 6, 3, 1, 10, 6, 6, 5, 4, 4, 2, 6, 6, 1, 6, 3, 6, 8, 5, 10, 4, 5, 8, 2, 7, 4, 3, 6, 4, 7, 10, 6, 4, 2, 1, 3, 7, 9, 2, 8, 4, 4, 1, 6, 1, 8, 3, 7, 9, 4, 8, 4, 1, 7, 3, 9, 10, 10, 6, 6, 6,

6, 2, 9, 3, 5, 6, 1, 2, 10, 1, 7, 6, 8, 5, 7, 8, 1, 5, 1, 8, 10, 5, 7, 6, 3, 8, 7, 3, 5, 6, 5, 6, 8, 4, 7, 2, 5, 3, 2, 3, 7, 5, 8, 4, 6, 2, 1, 1, 5, 4, 5, 4, 9, 2, 1, 2, 6, 3, 10, 1, 9, 6, 7, 5, 8, 10, 8, 4, 9, 4, 10, 10, 3, 4, 9, 10, 6, 9, 7, 5, 7, 6, 7, 4, 4, 9, 6, 1, 3, 10, 9, 7, 9, 10, 4, 8, 4, 8, 10, 10, 10, 9, 1, 8, 1, 10, 5, 3, 1, 2, 1, 2, 8, 2, 6, 2, 10, 6, 9, 2, 2, 5, 4, 6, 10, 10, 5, 4, 1, 7, 8, 5, 5, 5, 2, 2, 4, 2, 4, 9, 6, 8, 3, 1, 1, 2, 4, 1, 5, 3, 6, 7, 8, 1, 5, 1, 10, 1, 3, 7, 8, 2, 1, 4, 9, 10, 5, 6, 2, 8, 5, 4, 2, 5, 3, 7, 10, 3, 6, 2, 1, 4, 3, 8, 6, 4, 9, 5, 1, 7, 2, 4, 7, 9, 7, 9, 2, 7, 10, 4, 9, 1, 6, 7, 5, 10, 6, 6, 7, 4, 10, 8, 3, 3, 6, 6, 2, 4, 2, 5, 10, 10, 2, 9, 5, 5, 2, 7, 8, 8, 2, 1, 9, 10, 3, 8, 3, 8, 5, 10, 1, 7, 2, 8, 10, 7, 4, 4, 8, 9, 8, 4, 4, 2, 10, 4, 6, 8, 8, 6, 4, 8, 8, 7, 6, 1, 5, 2, 3, 5, 4, 2, 5, 7, 5, 9, 6, 2, 8, 6, 9, 1, 6, 1, 6, 10, 6, 2, 7, 10, 10, 1, 7, 1, 5, 1, 2, 10, 2, 4, 2, 10, 5, 2, 1, 5, 9, 2, 10, 10, 4, 8, 10, 2, 1, 4, 4, 2, 6, 4, 5, 1, 9, 8, 1, 7, 2, 4, 3, 9, 4, 2, 5, 5, 2, 5, 4, 10, 5, 5, 3, 9, 4, 3, 2, 8, 2, 6, 8, 4, 1, 2, 2, 7, 6, 10, 9, 6, 4, 7, 1, 6, 8, 6, 4, 1, 8, 8, 9, 9, 9, 1, 7, 1, 8, 5, 6, 5, 1, 10, 8, 1, 1, 2, 1, 9, 6, 4, 8, 4, 4, 5, 6, 4, 6, 5, 9, 10, 10, 2, 2, 8, 9, 8, 1, 3, 1, 9, 8, 7, 10, 5, 4, 7, 4, 5, 10, 9, 4, 8, 10, 8, 5, 7, 9, 2, 4, 7, 4, 10, 10, 2, 9, 2, 10, 10, 10, 8, 8, 4, 2, 8, 5, 10, 2, 4, 9, 1, 10, 2, 9, 7, 4, 5, 10, 8, 5, 10, 9, 5, 5, 5, 8, 7, 3, 3,

Batería de problemas

5 maquinas y 20 trabajos :

P_{ij} =8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4,

S_{ijk}=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, 7, 7, 9, 5, 9, 3, 2, 5, 2, 9, 7, 8, 2, 8, 8, 8, 9, 5, 9, 5, 6, 2, 6, 7, 7, 6, 1, 2, 6, 7, 1, 9, 8, 1, 6, 9, 2, 4, 6, 8, 9, 10, 9, 8, 4, 1, 5, 9, 2, 5, 3, 3, 4, 10, 5, 4, 3, 10, 2, 10, 2, 4, 8, 9, 1, 7, 10, 5, 10, 9, 9, 5, 10, 7, 1, 5, 7, 5, 2, 8, 9, 10, 2, 5, 7, 1, 2, 8, 9, 10, 3, 2, 7, 3, 1, 2, 4, 8, 6, 7, 4, 4, 5, 2, 3, 6, 5, 6, 5, 7, 3, 1, 4, 9, 4, 1, 3, 1, 3, 3, 8, 9, 5, 7, 8, 5, 6, 8, 8, 2, 7, 5, 9, 4, 5, 8, 8, 8, 1, 7, 7, 6, 3, 9, 9, 7, 5, 6, 5, 6, 2, 10, 2, 8, 5, 5, 2, 3, 2, 2, 10, 8, 5, 3, 7, 7, 10, 10, 1, 10, 6, 6, 8, 6, 3, 6, 4, 9, 6, 4, 1, 7, 2, 9, 10, 7, 10, 3, 7, 6, 10, 6, 4, 10, 10, 4, 8, 6, 9, 4, 10, 9, 3, 7, 10, 6, 3, 9, 7, 1, 4, 2, 6, 10, 10, 5, 2, 1, 4, 9, 9, 5, 2, 9, 8, 9, 5, 8, 8, 2, 4, 1, 8, 10, 5, 10, 5, 2, 9, 2, 4, 2, 3, 6, 10, 8, 4, 9, 4, 5, 7, 1, 8, 7, 7, 8, 4, 6, 9, 6, 1, 8, 4, 5, 2, 8, 4, 9, 4, 8, 2, 8, 5, 6, 2, 5, 5, 3, 9, 7, 5, 8, 10, 8, 10, 4, 2, 5, 2, 9, 2, 3, 6, 5, 3, 4, 8, 8, 6, 2, 7, 5, 9, 5, 9, 6, 6, 3, 4, 4, 8, 7, 5, 3, 8, 8, 5, 7, 9, 8, 5, 5, 7, 9, 1, 6, 8, 4, 2, 4, 8, 3, 10, 6, 5, 7, 5, 2, 7, 10, 7, 4, 9, 7, 2, 9, 8, 10, 3, 4, 5, 1, 10, 2, 7, 6, 4, 8, 8, 7, 9, 5, 3, 5, 2, 1, 7, 5, 2, 9, 2, 2, 10, 6, 3, 7, 5, 5, 10, 8, 10, 2, 5, 8, 7, 9, 2, 5, 1, 9, 5, 9, 1, 6, 5, 2, 5, 6, 3, 5, 8, 2, 6, 3, 2, 5, 5, 10, 6, 1, 6, 5, 8, 3, 3, 8, 2, 10, 5, 7, 3, 8, 4, 9, 10, 2, 1, 2, 7, 4, 3, 2, 6, 5, 9, 10, 3, 2, 9, 7, 3, 2, 6, 3, 7, 10, 10, 3, 10, 8, 7, 2, 6, 2, 3, 7, 9, 8, 10, 1, 7, 1, 6, 8, 4, 5, 6, 3, 1, 10, 6, 6, 5, 4, 4, 2, 6, 6, 1, 6, 3, 6, 8, 5, 10, 4, 5, 8, 2, 7, 4, 3, 6, 4, 7, 10, 6, 4, 2, 1, 3, 7, 9, 2, 8, 4, 4, 1, 6, 1, 8, 3, 7, 9, 4, 8, 4, 1, 7, 3, 9, 10, 10, 6, 6, 6, 6, 2, 9, 3, 5, 6, 1, 2, 10, 1, 7, 6, 8, 5, 7, 8, 1, 5, 1, 8, 10, 5, 7, 6, 3, 8, 7, 3, 5, 6, 5, 6, 8, 4, 7, 2, 5, 3, 2, 3, 7, 5, 8, 4, 6, 2, 1, 1, 5, 4, 5, 4, 9, 2, 1, 2, 6, 3, 10, 1, 9, 6, 7, 5, 8, 10, 8, 4, 9, 4, 10, 10, 3, 4, 9, 10, 6, 9, 7, 5, 7, 6, 7, 4, 4, 9, 6, 1, 3, 10, 9, 7, 9, 10, 4, 8, 4, 8, 10, 10, 10, 9, 1, 8, 1, 10, 5, 3, 1, 2, 1, 2, 8, 2, 6, 2, 10, 6, 9, 2, 2, 5, 4, 6, 10, 10, 5, 4, 1, 7, 8, 5, 5, 5, 2, 2, 4, 2, 4, 9, 6, 8, 3, 1, 1, 2, 4, 1 , 5, 3, 6, 7, 8, 1, 5, 1, 10, 1, 3, 7, 8, 2, 1, 4, 9, 10, 5, 6, 2, 8, 5, 4, 2, 5, 3, 7, 10, 3, 6, 2, 1, 4, 3, 8, 6, 4, 9, 5, 1, 7, 2, 4, 7, 9, 7, 9, 2, 7, 10, 4, 9, 1, 6, 7, 5, 10, 6, 6, 7, 4, 10, 8, 3, 3, 6, 6, 2, 4, 2, 5, 10, 10, 2, 9, 5, 5, 2, 7, 8, 8, 2, 1, 9, 10, 3, 8, 3, 8, 5, 10, 1, 7, 2, 8, 10, 7, 4, 4, 8, 9, 8, 4, 4, 2, 10, 4, 6, 8, 8, 6, 4, 8, 8, 7, 6, 1, 5, 2, 3, 5, 4,

2, 5, 7, 5, 9, 6, 2, 8, 6, 9, 1, 6, 1, 6, 10, 6, 2, 7, 10, 10, 1, 7, 1, 5, 1, 2, 10, 2, 4, 2, 10, 5, 2, 1, 5, 9, 2, 10, 10, 4, 8, 10, 2, 1, 4, 4, 2, 6, 4, 5, 1, 9, 8, 1, 7, 2, 4, 3, 9, 4, 2, 5, 5, 2, 5, 4, 10, 5, 5, 3, 9, 4, 3, 2, 8, 2, 6, 8, 4, 1, 2, 2, 7, 6, 10, 9, 6, 4, 7, 1, 6, 8, 6, 4, 1, 8, 8, 9, 9, 9, 1, 7, 1, 8, 5, 6, 5, 1, 10, 8, 1, 1, 2, 1, 9, 6, 4, 8, 4, 4, 5, 6, 4, 6, 5, 9, 10, 10, 2, 2, 8, 9, 8, 1, 3, 1, 9, 8, 7, 10, 5, 4, 7, 4, 5, 10, 9, 4, 8, 10, 8, 5, 7, 9, 2, 4, 7, 4, 10, 10, 2, 9, 2, 10, 10, 10, 8, 8, 4, 2, 8, 5, 10, 2, 4, 9, 1, 10, 2, 9, 7, 4, 5, 10, 8, 5, 10, 9, 5, 5, 5, 8, 7, 3, 3, 4, 10, 10, 4, 2, 9, 5, 7, 6, 4, 8, 10, 10, 1, 10, 8, 8, 9, 10, 9, 5, 4, 9, 1, 6, 6, 6, 3, 4, 4, 9, 1, 5, 1, 2, 8, 6, 7, 4, 2, 6, 4, 2, 7, 4, 4, 8, 8, 8, 7, 5, 1, 2, 10, 3, 9, 9, 5, 1, 8, 7, 6, 2, 8, 6, 6, 4, 2, 7, 7, 9, 5, 1, 6, 10, 8, 5, 5, 3, 8, 5, 5, 10, 3, 9, 7, 10, 4, 8, 5, 4, 6, 3, 1, 3, 6, 3, 2, 9, 6, 8, 4, 9, 4, 5, 7, 6, 6, 9, 7, 4, 8, 3, 6, 9, 1, 7, 8, 2, 6, 7, 7, 1, 4, 6, 8, 2, 1, 6, 5, 10, 6, 4, 5, 5, 9, 10, 9, 10, 1, 5, 1, 5, 9, 3, 1, 1, 1, 5, 4, 1, 1, 4, 6, 7, 2, 5, 2, 5, 1, 4, 6, 9, 4, 2, 10, 3, 7, 1, 8, 10, 6, 9, 4, 3, 2, 1, 1, 3, 1, 7, 8, 9, 8, 4, 9, 5, 2, 5, 4, 3, 4, 9, 10, 7, 8, 8, 10, 8, 1, 5, 7, 5, 3, 6, 2, 9, 3, 3, 9, 7, 2, 9, 7, 5, 9, 8, 10, 8, 6, 8, 3, 3, 8, 6, 9, 6, 7, 6, 9, 1, 9, 7, 5, 10, 2, 6, 8, 1, 7, 6, 9, 1, 2, 7, 10, 9, 5, 2, 4, 10, 6, 4, 2, 2, 8, 3, 3, 1, 7, 5, 3, 8, 6, 8, 6, 3, 7, 5, 10, 10, 2, 2, 4, 5, 6, 10, 7, 9, 3, 4, 1, 3, 10, 9, 6, 2, 4, 2, 3, 5, 1, 9, 3, 9, 8, 1, 8, 3, 10, 4, 6, 8, 6, 4, 6, 5, 1, 10, 1, 8, 5, 3, 7, 6, 4, 5, 6, 7, 4, 4, 8, 5, 2, 8, 8, 9, 9, 7, 10, 9, 10, 7, 3, 2, 9, 8, 7, 10, 7, 1, 10, 3, 9, 1, 8, 6, 10, 2, 6, 7, 10, 3, 6, 7, 4, 7, 2, 7, 7, 5, 10, 3, 9, 6, 10, 9, 5, 6, 6, 1, 8, 4, 10, 10, 3, 10, 8, 10, 4, 2, 3, 6, 2, 5, 3, 9, 3, 6, 2, 8, 2, 10, 7, 6, 8, 10, 1, 9, 6, 1, 4, 1, 6, 10, 2, 10, 9, 7, 2, 6, 1, 1, 3, 4, 9, 5, 10, 1, 5, 10, 6, 7, 6, 10, 4, 10, 7, 7, 8, 1, 6, 3, 8, 3, 9, 7, 1, 6, 8, 5, 3, 6, 1, 9, 8, 10, 2, 6, 7, 3, 2, 1, 4, 6, 2, 3, 8, 6, 9, 7, 6, 7, 6, 2, 7, 1, 4, 5, 9, 6, 3, 1, 6, 2, 6, 8, 3, 7, 3, 10, 1, 7, 1, 3, 1, 7, 10, 6, 2, 9, 4, 9, 9, 10, 8, 5, 7, 1, 6, 8, 5, 6, 8, 9, 9, 6, 4, 5, 9, 2, 7, 1, 7, 4, 6, 9, 2, 7, 7, 9, 1, 5, 2, 7, 8, 9, 7, 8, 10, 7, 9, 1, 1, 3, 2, 3, 3, 4, 9, 1, 8, 5, 10, 1, 4, 1, 6, 9, 2, 3, 4, 6, 8, 7, 6, 6, 2, 4, 2, 7, 7, 10, 4, 9, 6, 8, 9, 5, 9, 8, 8, 10, 9, 7, 8, 4, 5, 1, 5, 9, 6, 5, 7, 10, 3, 6, 5, 2, 4, 8, 5, 5, 6, 1, 7, 5, 8, 10, 3, 5, 2, 6, 4, 10, 10, 10, 9, 5, 4, 6, 2, 8, 6, 2, 6, 9, 5, 10, 9, 3, 9, 8, 2, 3, 4, 3, 6, 6, 4, 10, 2, 3, 3, 6, 7, 2, 4, 4, 3, 5, 6, 10, 9, 2, 2, 5, 4, 8, 6, 6, 9, 2, 8, 2, 9, 7, 10, 10, 5, 6, 1, 8, 10, 5, 8, 5, 6, 10, 5, 1, 4, 2, 3, 10, 5, 7, 7, 6, 4, 5, 4, 4, 5, 1, 6, 3, 4, 3, 7, 8, 8, 1, 6, 4, 2, 7, 9, 2, 5, 4, 8, 10, 4, 8, 2, 3, 2, 6, 3, 10, 10, 2, 8, 3, 5, 1, 1, 2, 6, 9, 4, 4, 6, 5, 10, 7, 2, 9, 7, 2, 10, 8, 4, 9, 10, 2, 1, 2, 1, 7, 5, 7, 9, 4, 1, 8, 3, 6, 6, 9, 5, 3, 9, 9, 9, 8, 9, 6, 9, 7, 10, 1, 1, 3, 3, 6, 4, 6, 1, 7, 8, 5, 4, 10, 1, 9, 3, 7, 7, 6, 5, 5, 3, 1, 1, 5, 10, 7, 6, 7, 10, 1, 5, 5, 10, 4, 9, 4, 10, 6, 5, 10, 8, 10, 5, 8, 4, 10, 6, 5, 5, 8, 4, 10, 6, 7, 9, 9, 6, 3, 1, 7, 7, 9, 1, 1, 7, 1, 5, 6, 2, 1, 2, 9, 1, 4, 7, 5, 10, 5, 5, 8, 9, 2, 2, 3, 6, 3, 6, 10, 1, 9, 6, 5, 2, 8, 3, 10, 8, 10, 5, 8, 4, 5, 7, 6, 3, 10, 2, 6, 1, 10, 4, 3,

Batería de problemas

5 maquinas y 25 trabajos :

P_{ij}=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9,

S_{ijk}=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, 7, 7, 9, 5, 9, 3, 2, 5, 2, 9, 7, 8, 2, 8, 8, 8, 9, 5, 9, 5, 6, 2, 6, 7, 7, 6, 1, 2, 6, 7, 1, 9, 8, 1, 6, 9, 2, 4, 6, 8, 9, 10, 9, 8, 4, 1, 5, 9, 2, 5, 3, 3, 4, 10, 5, 4, 3, 10, 2, 10, 2, 4, 8, 9, 1, 7, 10, 5, 10, 9, 9, 5, 10, 7, 1, 5, 7, 5, 2, 8, 9, 10, 2, 5, 7, 1, 2, 8, 9, 10, 3, 2, 7, 3, 1 , 2, 4, 8, 6, 7, 4, 4, 5, 2, 3, 6, 5, 6, 5, 7, 3, 1, 4, 9, 4, 1, 3, 1, 3, 3, 8, 9, 5, 7, 8, 5, 6, 8, 8, 2, 7, 5, 9, 4, 5, 8, 8, 8, 1, 7, 7, 6, 3, 9, 9, 7, 5, 6, 5, 6, 2, 10, 2, 8, 5, 5, 2, 3, 2, 2, 10, 8, 5, 3, 7, 7, 10, 10, 1, 10, 6, 6, 8, 6, 3, 6, 4, 9, 6, 4, 1, 7, 2, 9, 10, 7, 10, 3, 7, 6, 10, 6, 4, 10, 10, 4, 8, 6, 9, 4, 10, 9, 3, 7, 10, 6, 3, 9, 7, 1, 4, 2, 6,

10, 10, 5, 2, 1, 4, 9, 9, 5, 2, 9, 8, 9, 5, 8, 8, 2, 4, 1, 8, 10, 5, 10, 5, 2, 9, 2, 4, 2, 3, 6, 10, 8, 4, 9, 4, 5, 7, 1, 8, 7, 7, 8, 4, 6, 9, 6, 1, 8, 4, 5, 2, 8, 4, 9, 4, 8, 2, 8, 5, 6, 2, 5, 5, 3, 9, 7, 5, 8, 10, 8, 10, 4, 2, 5, 2, 9, 2, 3, 6, 5, 3, 4, 8, 8, 6, 2, 7, 5, 9, 5, 9, 6, 6, 3, 4, 4, 8, 7, 5, 3, 8, 8, 5, 7, 9, 8, 5, 5, 7, 9, 1, 6, 8, 4, 2, 4, 8, 3, 10, 6, 5, 7, 5, 2, 7, 10, 7, 4, 9, 7, 2, 9, 8, 10, 3, 4, 5, 1, 10, 2, 7, 6, 4, 8, 8, 7, 9, 5, 3, 5, 2, 1, 7, 5, 2, 9, 2, 2, 10, 6, 3, 7, 5, 5, 10, 8, 10, 2, 5, 8, 7, 9, 2, 5, 1, 9, 5, 9, 1, 6, 5, 2, 5, 6, 3, 5, 8, 2, 6, 3, 2, 5, 5, 10, 6, 1, 6, 5, 8, 3, 3, 8, 2, 10, 5, 7, 3, 8, 4, 9, 10, 2, 1, 2, 7, 4, 3, 2, 6, 5, 9, 10, 3, 2, 9, 7, 3, 2, 6, 3, 7, 10, 10, 3, 10, 8, 7, 2, 6, 2, 3, 7, 9, 8, 10, 1, 7, 1, 6, 8, 4, 5, 6, 3, 1, 10, 6, 6, 5, 4, 4, 2, 6, 6, 1, 6, 3, 6, 8, 5, 10, 4, 5, 8, 2, 7, 4, 3, 6, 4, 7, 10, 6, 4, 2, 1, 3, 7, 9, 2, 8, 4, 4, 1, 6, 1, 8, 3, 7, 9, 4, 8, 4, 1, 7, 3, 9, 10, 10, 6, 6, 6, 6, 2, 9, 3, 5, 6, 1, 2, 10, 1, 7, 6, 8, 5, 7, 8, 1, 5, 1, 8, 10, 5, 7, 6, 3, 8, 7, 3, 5, 6, 5, 6, 8, 4, 7, 2, 5, 3, 2, 3, 7, 5, 8, 4, 6, 2, 1, 1, 5, 4, 5, 4, 9, 2, 1, 2, 6, 3, 10, 1, 9, 6, 7, 5, 8, 10, 8, 4, 9, 4, 10, 10, 3, 4, 9, 10, 6, 9, 7, 5, 7, 6, 7, 4, 4, 9, 6, 1, 3, 10, 9, 7, 9, 10, 4, 8, 4, 8, 10, 10, 10, 9, 1, 8, 1, 10, 5, 3, 1, 2, 1, 2, 8, 2, 6, 2, 10, 6, 9, 2, 2, 5, 4, 6, 10, 10, 5, 4, 1, 7, 8, 5, 5, 5, 2, 2, 4, 2, 4, 9, 6, 8, 3, 1, 1, 2, 4, 1, 5, 3, 6, 7, 8, 1, 5, 1, 10, 1, 3, 7, 8, 2, 1, 4, 9, 10, 5, 6, 2, 8, 5, 4, 2, 5, 3, 7, 10, 3, 6, 2, 1, 4, 3, 8, 6, 4, 9, 5, 1, 7, 2, 4, 7, 9, 7, 9, 2, 7, 10, 4, 9, 1, 6, 7, 5, 10, 6, 6, 7, 4, 10, 8, 3, 3, 6, 6, 2, 4, 2, 5, 10, 10, 2, 9, 5, 5, 2, 7, 8, 8, 2, 1, 9, 10, 3, 8, 3, 8, 5, 10, 1, 7, 2, 8, 10, 7, 4, 4, 8, 9, 8, 4, 4, 2, 10, 4, 6, 8, 8, 6, 4, 8, 8, 7, 6, 1, 5, 2, 3, 5, 4, 2, 5, 7, 5, 9, 6, 2, 8, 6, 9, 1, 6, 1, 6, 10, 6, 2, 7, 10, 10, 1, 7, 1, 5, 1, 2, 10, 2, 4, 2, 10, 5, 2, 1, 5, 9, 2, 10, 10, 4, 8, 10, 2, 1, 4, 4, 2, 6, 4, 5, 1, 9, 8, 1, 7, 2, 4, 3, 9, 4, 2, 5, 5, 2, 5, 4, 10, 5, 5, 3, 9, 4, 3, 2, 8, 2, 6, 8, 4, 1, 2, 2, 7, 6, 10, 9, 6, 4, 7, 1, 6, 8, 6, 4, 1, 8, 8, 9, 9, 9, 1, 7, 1, 8, 5, 6, 5, 1, 10, 8, 1, 1, 2, 1, 9, 6, 4, 8, 4, 4, 5, 6, 4, 6, 5, 9, 10, 10, 2, 2, 8, 9, 8, 1, 3, 1, 9, 8, 7, 10, 5, 4, 7, 4, 5, 10, 9, 4, 8, 10, 8, 5, 7, 9, 2, 4, 7, 4, 10, 10, 2, 9, 2, 10, 10, 10, 8, 8, 4, 2, 8, 5, 10, 2, 4, 9, 1, 10, 2, 9, 7, 4, 5, 10, 8, 5, 10, 9, 5, 5, 5, 8, 7, 3, 3, 4, 10, 10, 4, 2, 9, 5, 7, 6, 4, 8, 10, 10, 1, 10, 8, 8, 9, 10, 9, 5, 4, 9, 1, 6, 6, 6, 3, 4, 4, 9, 1, 5, 1, 2, 8, 6, 7, 4, 2, 6, 4, 2, 7, 4, 4, 8, 8, 8, 7, 5, 1, 2, 10, 3, 9, 9, 5, 1, 8, 7, 6, 2, 8, 6, 6, 4, 2, 7, 7, 9, 5, 1, 6, 10, 8, 5, 5, 3, 8, 5, 5, 10, 3, 9, 7, 10, 4, 8, 5, 4, 6, 3, 1, 3, 6, 3, 2, 9, 6, 8, 4, 9, 4, 5, 7, 6, 6, 9, 7, 4, 8, 3, 6, 9, 1, 7, 8, 2, 6, 7, 7, 1, 4, 6, 8, 2, 1, 6, 5, 10, 6, 4, 5, 5, 9, 10, 9, 10, 1, 5, 1, 5, 9, 3, 1, 1, 1, 5, 4, 1, 1, 4, 6, 7, 2, 5, 2, 5, 1, 4, 6, 9, 4, 2, 10, 3, 7, 1, 8, 10, 6, 9, 4, 3, 2, 1, 1, 3, 1, 7, 8, 9, 8, 4, 9, 5, 2, 5, 4, 3, 4, 9, 10, 7, 8, 8, 10, 8, 1, 5, 7, 5, 3, 6, 2, 9, 3, 3, 9, 7, 2, 9, 7, 5, 9, 8, 10, 8, 6, 8, 3, 3, 8, 6, 9, 6, 7, 6, 9, 1, 9, 7, 5, 10, 2, 6, 8, 1, 7, 6, 9, 1, 2, 7, 10, 9, 5, 2, 4, 10, 6, 4, 2, 2, 8, 3, 3, 1, 7, 5, 3, 8, 6, 8, 6, 3, 7, 5, 10, 10, 2, 2, 4, 5, 6, 10, 7, 9, 3, 4, 1, 3, 10, 9, 6, 2, 4, 2, 3, 5, 1, 9, 3, 9, 8, 1, 8, 3, 10, 4, 6, 8, 6, 4, 6, 5, 1, 10, 1, 8, 5, 3, 7, 6, 4, 5, 6, 7, 4, 4, 8, 5, 2, 8, 8, 9, 9, 7, 10, 9, 10, 7, 3, 2, 9, 8, 7, 10, 7, 1, 10, 3, 9, 1, 8, 6, 10, 2, 6, 7, 10, 3, 6, 7, 4, 7, 2, 7, 7, 5, 10, 3, 9, 6, 10, 9, 5, 6, 6, 1, 8, 4, 10, 10, 3, 10, 8, 10, 4, 2, 3, 6, 2, 5, 3, 9, 3, 6, 2, 8, 2, 10, 7, 6, 8, 10, 1, 9, 6, 1, 4, 1, 6, 10, 2, 10, 9, 7, 2, 6, 1, 1, 3, 4, 9, 5, 10, 1, 5, 10, 6, 7, 6, 10, 4, 10, 7, 7, 8, 1, 6, 3, 8, 3, 9, 7, 1, 6, 8, 5, 3, 6, 1, 9, 8, 10, 2, 6, 7, 3, 2, 1, 4, 6, 2, 3, 8, 6, 9, 7, 6, 7, 6, 2, 7, 1, 4, 5, 9, 6, 3, 1, 6, 2, 6, 8, 3, 7, 3, 10, 1, 7, 1, 3, 1, 7, 10, 6, 2, 9, 4, 9, 9, 10, 8, 5, 7, 1, 6, 8, 5, 6, 8, 9, 9, 6, 4, 5, 9, 2, 7, 1, 7, 4, 6, 9, 2, 7, 7, 9, 1, 5, 2, 7, 8, 9, 7, 8, 10, 7, 9, 1, 1, 3, 2, 3, 3, 4, 9, 1, 8, 5, 10, 1, 4, 1, 6, 9, 2, 3, 4, 6, 8, 7, 6, 6, 2, 4, 2, 7, 7, 10, 4, 9, 6, 8, 9, 5, 9, 8, 8, 10, 9, 7, 8, 4, 5, 1, 5, 9, 6, 5, 7, 10, 3, 6, 5, 2, 4, 8, 5, 5, 6, 1, 7, 5, 8, 10, 3, 5, 2, 6, 4, 10, 10, 10, 9, 5, 4, 6, 2, 8, 6, 2, 6, 9, 5, 10, 9, 3, 9, 8, 2, 3, 4, 3, 6, 6, 4, 10, 2, 3, 3, 6, 7, 2, 4, 4, 3, 5, 6, 10, 9, 2, 2, 5, 4, 8, 6, 6, 9, 2, 8, 2, 9, 7, 10, 10, 5, 6, 1, 8, 10, 5, 8, 5, 6, 10, 5, 1, 4, 2, 3, 10, 5, 7, 7, 6, 4, 5, 4, 4, 5, 1, 6, 3, 4, 3, 7, 8, 8, 1, 6, 4, 2, 7, 9, 2, 5, 4, 8, 10, 4, 8, 2, 3, 2, 6, 3, 10, 10, 2, 8, 3, 5, 1, 1, 2, 6, 9, 4, 4, 6, 5, 10, 7, 2, 9, 7, 2, 10, 8, 4, 9, 10, 2, 1, 2, 1, 7, 5, 7, 9, 4, 1, 8, 3, 6, 6, 9, 5, 3, 9, 9, 9, 8, 9, 6, 9, 7, 10, 1, 1, 3, 3, 6, 4, 6, 1, 7, 8, 5, 4, 10, 1, 9, 3, 7, 7, 6, 5, 5, 3, 1, 1, 5, 10, 7, 6, 7, 10, 1, 5, 5, 10, 4, 9, 4, 10, 6, 5, 10, 8, 10, 5, 8, 4, 10, 6, 5, 5, 8, 4, 10, 6, 7, 9, 9, 6, 3, 1, 7, 7, 9, 1, 1, 7, 1, 5, 6, 2, 1, 2, 9, 1, 4, 7, 5, 10, 5, 5, 8, 9, 2, 2, 3, 6, 3, 6, 10, 1, 9, 6, 5, 2, 8, 3, 10, 8, 10, 5, 8, 4, 5, 7, 6, 3, 10, 2, 6, 1, 10, 4, 3, 9, 4, 4, 7, 1, 3, 8, 9, 1, 1, 10, 7, 7, 1, 4, 1, 6, 5, 10, 3, 8, 4, 2, 9, 8, 2, 7, 4, 1, 2, 4, 3, 2, 7, 6, 10, 3, 3, 1, 10, 6, 7, 4, 6, 8, 2, 1, 10, 1, 9, 5, 10, 10, 7, 8, 4, 6, 9, 4, 7, 8, 9, 3, 1, 10, 7, 10, 3, 8, 6, 9, 1, 8, 2, 1, 3, 4, 5, 8, 7, 8, 7, 5, 9, 3, 1, 6, 3, 1, 9, 6, 1, 8, 7, 9, 3, 3, 4, 5, 2, 8, 9, 10, 5, 10, 10, 6, 6, 1, 9, 2, 2, 2, 5, 5, 2, 6, 5, 5, 2, 9, 2, 5, 8, 2, 4, 4, 1, 5, 9, 7, 8, 10, 6, 1, 9, 1, 5, 7, 2, 8, 6, 9, 7, 9, 7, 10, 8, 9, 2, 1, 4, 9, 5, 2, 3, 6, 9, 10, 7, 10, 4, 5, 2, 4, 3, 7, 4, 1, 10, 4, 7, 9, 2, 7, 2, 5, 7, 1, 5, 1, 5, 8, 2, 5, 10, 4, 2, 7, 7, 2, 3, 6, 4, 2, 3, 10, 3, 4, 4, 5, 1, 10, 2, 5, 3, 7, 3, 5, 4, 9, 1, 6, 1, 8, 5, 3, 1, 2, 1, 8, 2, 9, 1, 9, 7, 3, 5, 2, 8, 5, 6, 1, 9, 7, 8, 2, 8, 8, 2, 4, 2, 2, 8, 6, 1, 7, 3, 9, 5, 5, 1, 1, 8, 7, 2, 2, 7, 5, 4, 4, 10, 10, 6, 9, 6, 10, 1, 6, 8, 5, 5, 2, 7, 1, 7, 5, 8, 2, 9, 9, 7, 9, 8, 8, 9, 7, 7, 3, 3, 7, 7, 4, 9, 7, 1, 3, 10, 8, 10, 4, 7, 7, 1, 6, 6, 9, 10, 3, 10, 5, 7, 7, 4, 7, 5, 2, 10, 3,

5, 5, 9, 2, 7, 6, 10, 2, 2, 3, 2, 3, 5, 4, 7, 8, 9, 8, 6, 7, 2, 7, 1, 7, 1, 1, 8, 8, 1, 9, 6, 1, 4, 4, 4, 2, 1, 3, 10, 7, 7, 6, 4, 3, 9, 7, 8, 2, 2, 2, 5, 8, 8, 7, 5, 7, 7, 7, 5, 3, 4, 4, 1, 1, 7, 5, 8, 8, 9, 1, 9, 9, 9, 4, 5, 9, 3, 9, 7, 7, 6, 8, 1, 7, 3, 2, 7, 8, 2, 8, 5, 7, 1, 5, 1, 5, 8, 5, 3, 6, 10, 9, 5, 10, 8, 2, 9, 7, 2, 1, 4, 10, 1, 6, 2, 5, 5, 1, 4, 5, 2, 10, 4, 1, 1, 5, 6, 10, 7, 4, 2, 8, 9, 3, 4, 3, 3, 3, 8, 4, 10, 7, 3, 9, 6, 2, 4, 1, 2, 2, 2, 3, 7, 8, 8, 8, 3, 2, 3, 8, 2, 9, 9, 7, 8, 6, 7, 5, 1, 1, 3, 5, 3, 9, 9, 10, 2, 4, 7, 2, 6, 5, 7, 7, 10, 7, 3, 5, 7, 10, 8, 3, 10, 8, 8, 4, 1, 9, 1, 2, 4, 6, 6, 4, 7, 1, 5, 7, 10, 3, 1, 2, 1, 8, 7, 6, 8, 4, 9, 7, 9, 3, 3, 9, 1, 9, 4, 9, 3, 6, 8, 2, 5, 4, 1, 9, 4, 5, 5, 3, 2, 1, 4, 6, 9, 9, 5, 4, 7, 2, 4, 6, 4, 8, 7, 3, 1, 1, 7, 2, 3, 10, 6, 1, 5, 1, 1, 10, 8, 2, 7, 4, 10, 2, 6, 9, 5, 9, 5, 2, 2, 3, 6, 5, 5, 1, 3, 10, 9, 10, 9, 2, 5, 9, 7, 7, 10, 5, 2, 9, 4, 6, 2, 1, 4, 10, 5, 2, 8, 10, 9, 6, 1, 8, 3, 7, 3, 1, 5, 5, 10, 4, 7, 5, 2, 3, 9, 9, 7, 10, 2, 7, 3, 2, 8, 4, 8, 1, 4, 2, 9, 8, 8, 8, 7, 6, 5, 9, 2, 10, 7, 9, 10, 1, 3, 2, 1, 1, 5, 9, 7, 3, 10, 6, 6, 6, 1, 6, 6, 5, 2, 5, 8, 4, 7, 3, 5, 3, 9, 6, 2, 1, 9, 8, 2, 5, 6, 2, 5, 1, 2, 6, 4, 10, 10, 3, 7, 5, 5, 6, 1, 10, 9, 4, 10, 2, 10, 10, 1, 4, 6, 9, 7, 4, 9, 10, 7, 9, 10, 4, 4, 1, 4, 10, 2, 9, 3, 6, 4, 1, 7, 9, 2, 1, 7, 5, 7, 5, 1, 3, 6, 3, 8, 5, 5, 9, 7, 8, 1, 6, 3, 1, 8, 1, 3, 5, 4, 10, 9, 7, 10, 2, 4, 6, 6, 3, 7, 1, 8, 8, 1, 9, 3, 4, 3, 4, 10, 9, 1, 10, 4, 4, 8, 8, 1, 9, 10, 2, 1, 3, 8, 5, 6, 5, 2, 4, 7, 10, 1, 8, 10, 7, 10, 8, 1, 1, 10, 5, 6, 5, 1, 7, 9, 10, 7, 10, 7, 4, 7, 6, 6, 6, 1, 3, 7, 10, 3, 6, 2, 1, 3, 9, 9, 9, 8, 7, 2, 9, 3, 2, 7, 5, 7, 10, 5, 10, 1, 9, 2, 2, 8, 1, 7, 9, 2, 9, 4, 6, 7, 8, 8, 8, 1, 5, 10, 8, 3, 6, 2, 8, 2, 1, 9, 4, 9, 6, 7, 2, 4, 4, 1, 6, 2, 9, 4, 1, 5, 1, 9, 7, 7, 3, 5, 7, 10, 5, 1, 3, 8, 6, 6, 5, 7, 1, 1, 10, 8, 2, 8, 2, 6, 1, 5, 8, 9, 9, 5, 8, 9, 10, 4, 6, 1, 8, 7, 10, 3, 8, 5, 8, 4, 8, 9, 4, 1, 9, 2, 5, 7, 10, 4, 6, 4, 6, 7, 3, 4, 7, 9, 9, 1, 8, 10, 9, 5, 8, 3, 9, 4, 4, 5, 7, 9, 2, 1, 6, 7, 1, 7, 7, 2, 7, 9, 1, 10, 8, 1, 3, 9, 8, 10, 7, 10, 6, 2, 10, 10, 10, 2, 1, 5, 4, 9, 6, 7, 3, 7, 7, 7, 3, 2, 8, 9, 1, 6, 1, 8, 10, 9, 8, 10, 9, 6, 4, 7, 5, 8, 5, 3, 9, 4, 4, 2, 1, 6, 3, 2, 9, 10, 1, 1, 10, 5, 8, 7, 1, 10, 7, 9, 10, 8, 6, 8, 5, 6, 2, 7, 10, 1, 6, 6, 8, 8, 1, 4, 4, 9, 8, 9, 9, 10, 5, 7, 6, 3, 7, 2, 3, 2, 1, 1, 10, 5, 6, 5, 7, 1, 8, 5, 2, 9, 2, 9, 4, 7, 9, 2, 2, 4, 9, 8, 7, 8, 8, 4, 10, 5, 5, 2, 8, 10, 10, 8, 9, 9, 10,

Código lingo modelo objetivo 1

```
MODEL:

SETS:

Maquinas /1..5/: C;

Trabajos /1..25/;

Posicion /1..6/;

Precedencia (Maquinas,Trabajos,Trabajos): S, beta;

Maq_Tbj (Maquinas, Trabajos): P;

Asignar (Maquinas, Trabajos, Posicion): alfa;

ENDSETS

DATA:

P=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9;
```

S=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5, 1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5, 10, 9, 9, 7, 8, 10, 5, 9, 10, 10, 7, 7, 9, 5, 9, 3, 2, 5, 2, 9, 7, 8, 2, 8, 8, 8, 9, 5, 9, 5, 6, 2, 6, 7, 7, 6, 1, 2, 6, 7, 1, 9, 8, 1, 6, 9, 2, 4, 6, 8, 9, 10, 9, 8, 4, 1, 5, 9, 2, 5, 3, 3, 4, 10, 5, 4, 3, 10, 2, 10, 2, 4, 8, 9, 1, 7, 10, 5, 10, 9, 9, 5, 10, 7, 1, 5, 7, 5, 2, 8, 9, 10, 2, 5, 7, 1, 2, 8, 9, 10, 3, 2, 7, 3, 1, 2, 4, 8, 6, 7, 4, 4, 5, 2, 3, 6, 5, 6, 5, 7, 3, 1, 4, 9, 4, 1, 3, 1, 3, 3, 8, 9, 5, 7, 8, 5, 6, 8, 8, 2, 7, 5, 9, 4, 5, 8, 8, 8, 1, 7, 7, 6, 3, 9, 9, 7, 5, 6, 5, 6, 2, 10, 2, 8, 5, 5, 2, 3, 2, 2, 10, 8, 5, 3, 7, 7, 10, 10, 1, 10, 6, 6, 8, 6, 3, 6, 4, 9, 6, 4, 1, 7, 2, 9, 10, 7, 10, 3, 7, 6, 10, 6, 4, 10, 10, 4, 8, 6, 9, 4, 10, 9, 3, 7, 10, 6, 3, 9, 7, 1, 4, 2, 6, 10, 10, 5, 2, 1, 4, 9, 9, 5, 2, 9, 8, 9, 5, 8, 8, 2, 4, 1, 8, 10, 5, 10, 5, 2, 9, 2, 4, 2, 3, 6, 10, 8, …;

ENDDATA

!Tipo de variables;

@for(Asignar(i,j,r):@BIN(alfa(i,j,r)));

@for(Precedencia(i,j,k):@BIN(beta(i,j,k)));

!Restricciones;

@for(Trabajos(j):@sum(Maquinas(i):@sum(Posicion(r): alfa(i,j,r))) =1);

@for(Maquinas(i):@for(Posicion(r): @sum(Trabajos(j): alfa(i,j,r)) <=1));

@for(Maquinas(i):@for(Posicion(r)|r#GT#1: @sum(Trabajos(j): alfa(i,j,r)) <= @sum(Trabajos(j): alfa(i,j,r-1))));

@for(Maquinas (i):@for(Trabajos(j):@for(Trabajos(k) |k#NE#j:@for(Posicion(r)|r#LT#6: alfa(i,j,r)+alfa(i,k,r+1) <= 1+beta(i,j,k)))));

@for(Maquinas(i): C(i) = @sum(Trabajos(j):@sum(Posicion(r) |r#LT#6: P(i,j)*alfa(i,j,r))) + @sum(Trabajos(j):@sum(trabajos(k)|k#NE#j: S(i,j,k)*beta(i,j,k))));

MIn=Cmax;

@for(Maquinas(i): Cmax >= C(i));
end

Codigo lingo modelo objetivo 2

```
MODEL:

SETS:

Maquinas /1..5/: C;

Trabajos /1..15/;

Precedencia (Maquinas,Trabajos,Trabajos):Setup,beta,W,W1,W2;

Maq_Tbj (Maquinas, Trabajos): P, alfa, X, F,W3;

ENDSETS

DATA:

P=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5, 3, 5,
3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1, 6, 5,
6;

Setup=8, 9, 3, 3, 9, 1, 8, 7, 2, 2, 9, 10, 6, 10, 10, 1, 2, 3, 6, 5, 1, 5, 6, 8, 4, 7, 10, 2, 8, 6, 10, 6, 4, 5,
3, 5, 3, 3, 9, 9, 2, 4, 3, 10, 4, 2, 9, 6, 6, 4, 1, 6, 3, 9, 6, 1, 4, 7, 5, 5, 2, 4, 10, 10, 9, 5, 7, 6, 3, 2, 4, 1,
6, 5, 6, 9, 5, 8, 6, 4, 8, 3, 9, 10, 9, 3, 5, 8, 4, 2, 8, 2, 4, 9, 10, 2, 7, 9, 4, 4, 4, 8, 1, 10, 1, 4, 5, 7, 8, 5,
1, 5, 9, 3, 7, 8, 3, 1, 5, 1, 2, 1, 5,…;

CSx= 15000;

CSf=15000;

Clx=-15000;

ENDDATA

!Tipo de variables;

@for(Maq_Tbj(i,j):@BIN(alfa(i,j)));

@for(Precedencia(i,j,k):@BIN(beta(i,j,k)));

@for(Precedencia (i,j,k):@BIN(w(i,j,k)));

@for(Precedencia (i,j,k):@BIN(w1(i,j,k)));

@for(Precedencia (i,j,k):@BIN(w2(i,j,k)));

@for(Maq_Tbj(i,j):@BIN(W3(i,j)));
```

```
!Restricciones;

@for(Maquinas(i):@for(Trabajos(j): X(i,j) <= CSx*alfa(i,j)));

@for(Trabajos(j):@sum(Maquinas(i):alfa(i,j))=1);

@for(Maquinas(i):@for(Trabajos(j): F(i,j) >= X(i,j)+P(i,j)*alfa(i,j)));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: F(i,j) - X(i,k)<= CSf-CSf*W(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: F(i,k) - X(i,j) <= CSf-
CSf*W1(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: alfa(i,j) + alfa(i,k) >=
2*W2(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: alfa(i,j) + alfa(i,k) <=
1+W2(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: W(i,j,k) + W1(i,j,k) >= W2(i,j,k))));

@for(Maquinas(i):@for(Trabajos(k): X(i,k) >= W3(i,k)));

@for(Maquinas(i):@for(Trabajos(k): X(i,k) <= CSx*W3(i,k)));

@for(Maquinas(i):@for(Trabajos(k):@sum(Trabajos(j): beta(i,j,k)) >= W3(i,k)));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j: X(i,k)-X(i,j) <=  CSx*(1-
beta(i,j,k))+P(i,j)*beta(i,j,k))));

@for(Maquinas(i):@for(Trabajos(j):@for(Trabajos(k)|k#NE#j:X(i,k)-X(i,j) >= P(i,j)+ CIx*(1-
beta(i,j,k)))));

@for(Maquinas(i): C(i) = @sum(Trabajos(j):F(i,j)) + @sum(Trabajos(j): @sum(Trabajos(k):
Setup(i,j,k)*beta(i,j,k))));

Min=Cmax;

@for(Maquinas(i): Cmax >= C(i));

End
```

Código heuristica

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
```

```
int cmx(int n,int m, MAT_INT pij,  VECTOR_INT sec, MAT_INT sjk);

void ordenDecrecienteTiemposSM(int numeroTrabajos,int ordenTrabajos[],int tiemposProceso[]);
int main()
{
int n=10;
int m=5;
int semilla=12;
int j;
int i;
int sec[n];
int sumaTiemP[n];
printf("Secuencia:");
print_int_vector(sec,n);

// Tiempos de proceso
  MAT_INT pij=DIM_MAT_INT(m,n);
srand(semilla);
for(i=0;i<m;i++)
{
   for(j=0;j<n;j++)
   {
      pij[i][j]=rand()%10+1;
       sec[j]=j;
   }

}
printf("\nPij:\n");
print_int_matrix(pij,m,n);

// tiempos de setup
 MAT_INT sjk=DIM_MAT_INT(m*n,n);
srand(semilla);
for(i=0;i<m*n;i++)
```

```
{
   for(j=0;j<n;j++)
   {
      sjk[i][j]=rand()%10+1;
      }
}
printf("\nSjk:\n");
print_int_matrix(sjk,m*n,n);

//Calculo de cmax
int r1=cmx(n,m,pij,sec,sjk);
printf("\ncmax:%d",r1);

//Liberacion de las librerias

FREE_MAT_INT(pij, m);
FREE_MAT_INT(sjk, m);
// llamada de la funcción para el cálculo de cmax
int cmx(int n, int m, MAT_INT pij,  int secuenciaLPT[], MAT_INT sjk)
{
int j;
int i;
int k1;
int k2;



VECTOR_INT maq=DIM_VECTOR_INT(m);
setval_Ivector(maq,m,0);
VECTOR_INT maq_aux=DIM_VECTOR_INT(m);

VECTOR_INT pos=DIM_VECTOR_INT(m);
setval_Ivector(pos,m,0);
VECTOR_INT pos_aux=DIM_VECTOR_INT(m);
```

```
for(j=0;j<n;j++)
{

copyIVector(maq,maq_aux,m);
sortIVector(maq_aux,m,'A');
printf("\n Maquina:%d",maq_aux[0]);
printf("\n");


pos_aux[0]=maq_aux[0];



pos[pos_aux[0]]=pos[pos_aux[0]]+secuenciaLPT[j];

printf("\n");


k1=pos[pos_aux[0]];


if (maq[maq_aux[0]]==0)
{

maq[maq_aux[0]]=maq[maq_aux[0]]+pij[maq_aux[0]][secuenciaLPT[j]];
printf("\n");
print_int_vector(maq,m);


}
else
{

k2=k1-secuenciaLPT[j];
```

```
printf("\n  k2:%d",k2);
maq[maq_aux[0]]=maq[maq_aux[0]]+pij[maq_aux[0]][secuenciaLPT[j]]+sjk[maq_aux[0]+m*secuenciaLPT[j]][k2];

printf("\n");
print_int_vector(maq,m);

}
if(pos[pos_aux[0]]<1000000000)
{
   pos[pos_aux[0]]=secuenciaLPT[j];
   k1=pos[pos_aux[0]];

}

}


 int cmax=0;

      j=0;
      for(j=0;j<m;j++)
      {
         if(maq[j]>cmax)
         {
            cmax=maq[j];
         }
      }

   return cmax;

}
 //llamada de la función secuencia

void ordenDecrecienteTiemposSM(int numeroTrabajos,int ordenTrabajos[],int tiemposProceso[])
{
```

```
    int i,j;
    int posicionMaxima,maximo;
    int tiemposProcesoAuxiliar[numeroTrabajos];
    for(i=0;i<numeroTrabajos;i++)
    {
       tiemposProcesoAuxiliar[i]=tiemposProceso[i];
    }
    for(i=0;i<numeroTrabajos;i++)
    {
       //Determinamos el i-esimo menor tiempo de proceso
       int primeraVez=1;
       for(j=0;j<numeroTrabajos;j++)
       {
          if(tiemposProcesoAuxiliar[j]!=-1)
          {
             if(primeraVez==1)
             {
                primeraVez=0;
                posicionMaxima=j;
                maximo=tiemposProcesoAuxiliar[j];
             }
             else if(tiemposProcesoAuxiliar[j]>maximo)
             {
                posicionMaxima=j;
                maximo=tiemposProcesoAuxiliar[j];
             }
          }
       }
       ordenTrabajos[i]=posicionMaxima;
       tiemposProcesoAuxiliar[posicionMaxima]=-1;
    }
}
```