Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Soluciones exactas para el problema de flowshop de permutación con restricciones de disponibilidad periódicas

Autor: Cristóbal Ramos Salgado
Tutora: Paz Pérez González

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

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Soluciones exactas para el problema de flowshop de permutación con restricciones de disponibilidad periódicas

Autor:
Cristóbal Ramos Salgado

Tutora:
Paz Pérez González

Dep. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: Soluciones exactas para el problema de flowshop de permutación con restricciones de disponibilidad periódicas

Autor: Cristóbal Ramos Salgado

Tutora: Paz Pérez González

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Un problema de programación de la producción consiste en la asignación de los distintos recursos de la empresa (máquinas) a la fabricación de un conjunto de productos.

Este proyecto tiene como objetivo el estudio de un modelo de programación de la producción que trata la problemática de minimizar el tiempo que los trabajos están en un sistema productivo. El escenario que se plantea es un entorno con máquinas en serie (conocido como flowshop) y periodos de indisponibilidad en las mismas. Además, los trabajos se consideran ininterrumpibles, lo que dificulta la programación de la producción.

Este problema de secuenciación no es trivial cuando se pretende optimizar un criterio relacionado con los tiempos de terminación de los trabajos. En este caso el objetivo a considerar es el máximo tiempo de terminación de todos los trabajos en el sistema (makespan).

El marco del problema trata de replicar un entorno productivo real. Las máquinas realizan paradas programadas de las que se conoce tanto su periodicidad como su duración. Estas pueden deberse tanto a descansos en la producción, como es el caso de los fines de semana, o a tareas de mantenimiento preventivo. Es por esto que los intervalos de disponibilidad en las máquinas serán periódicos y siempre iguales en duración.

El problema clásico de secuenciación de trabajos en un flowshop con objetivo makespan (denominado problema clásico en este documento) ha sido estudiado en la literatura profundamente. Sin embargo, el problema propuesto con periodos de indisponibilidad no ha sido estudiado aún.

1.1. Objetivos

El objetivo final del proyecto es la determinación de un modelo capaz de resolver de forma óptima, en el menor tiempo posible, el problema de programación de la producción expuesto anteriormente.

El procedimiento para ello será adaptar de la literatura sobre el problema clásico varios modelos matemáticos, diferentes entre ellos, capaces de establecer una secuencia de los trabajos en las máquinas que minimice el makespan.

Sin embargo, aunque todos los modelos encuentren la solución óptima, el tiempo que necesiten para conseguirla puede no ser el mismo. El tiempo de cómputo para un mismo problema diferirá según el modelo que se esté utilizando.

Por lo tanto, el objetivo principal del trabajo es realizar un análisis computacional para determinar qué modelo es más adecuado emplear en función del tamaño del problema.

Para obtener este objetivo principal se plantean los siguientes objetivos específicos:

- Analizar los modelos de la literatura para el problema clásico.
- Adaptar los modelos existentes al problema propuesto.
- Programar y modelar con el *solver* Gurobi los modelos adaptados.
- Validar los modelos adaptados mediante la resolución de problemas pequeños.
- Resolver una batería de instancias del problema para obtener los tiempos de cómputo.
- Determinar estadísticamente qué modelo es más eficiente para el problema propuesto.

1.2. Estructura del documento

En este primer capítulo se ha introducido el problema que es objeto de investigación, se han establecido los objetivos de este proyecto y se describe la estructura del documento. El resto de capítulos se estructuran siguiendo la línea de los objetivos específicos explicados anteriormente.

En el siguiente capítulo se describe en detalle el problema y se presenta el problema clásico, que servirá de base para el estudio del problema objeto.

En el capítulo 3, se describen cinco modelos que resuelven el problema clásico de forma óptima, en tiempos de cómputo razonables para instancias determinadas, y se adaptan para que cumplan las restricciones de indisponibilidad en las máquinas, propias del problema propuesto.

En el cuarto capítulo, se realiza un análisis computacional para comparar la eficiencia de los modelos diseñados, midiendo el tiempo de cómputo requerido para alcanzar la solución óptima.

En el capítulo 5, se exponen las conclusiones derivadas del análisis.

Por último, en un anexo se incluyen el código fuente de los modelos desarrollados en lenguaje Python.

Capítulo 2

Caracterización del problema

2.1. Introducción y Notación

Se indican a continuación algunos términos y definiciones generales extraídos de [Pinedo, 2005] y [Framinan et al., 2014]:

- Máquina: Recurso productivo capaz de realizar operaciones de transformación o de transporte a un material.
- Trabajo: Producto que es objeto de alguna operación de transformación o de transporte en alguna de las máquinas de la fábrica.
- Tiempo de proceso: Duración que requiere la operación de transformación de un trabajo en una determinada máquina.

La solución a un problema de programación de la producción es lo que se conoce como programa (*production schedule*). El programa asigna los trabajos a las máquinas y los ordena en la escala temporal. Esto es, qué máquina procesa qué trabajo, su orden de procesamiento en su paso por las máquinas (secuenciación) y los tiempos de inicio y fin de cada uno de ellos.

Todo problema de planificación de la producción es clasificado según tres aspectos que lo diferencian de cualquier otro Siguiendo la notación establecida por [Graham et al., 1979]:

- Criterio (γ). Función objetivo del problema. Existen múltiples criterios para optimizar la producción según sea el interés de la empresa. En general, los criterios dependen de los tiempos de terminación de los trabajos en el sistema. En algunos casos dependen de la fecha de entrega de los trabajos. Algunos ejemplos son: el tiempo de flujo total, el retraso máximo o el número de trabajos no entregados a tiempo.

- Entorno (α). Entorno de fabricación que indica la disposición de las máquinas. Los entornos se distinguen según el número de máquinas y su disposición en la fábrica. Entornos habituales son una única máquina, máquinas en paralelo o máquinas en serie.

- Restricciones (β). Condiciones sobre los trabajos o las máquinas. Algunas de las restricciones que se pueden considerar pueden ser: tiempos de llegada de los trabajos, tiempos de *setup*, interrupción de los trabajos, etc. Existen unas suposiciones generales que se consideran sin necesidad de ser indicadas en las restricciones: los trabajos, disponibles al principio del horizonte de programación, se consideran ininterrumpibles y las máquinas siempre disponibles. Cada máquina puede hacer un trabajo, y un trabajo puede ser realizado solo en una máquina. El buffer entre máquinas se supone infinito y el tiempo de transporte, despreciable.

Con estos tres conceptos, queda caracterizado cualquier problema, al que se puede hacer referencia con la notación:

$$\alpha \mid \beta \mid \gamma$$

2.2. Caracterización del problema

- Criterio (γ). En este trabajo el objetivo es minimizar la duración total del programa, es decir, la diferencia entre el tiempo de inicio del primer trabajo en la primera máquina y el de finalización del último trabajo en la última máquina. Este tiempo se conoce como *makespan* y su notación viene dada por C_{max}.

- Entorno (α). En este trabajo el conjunto de las máquinas considerado forma un entorno de tipo taller de flujo (*flowshop*), que consiste en que todos los trabajos siguen la misma secuencia para ser procesados a través de las máquinas. Se denota F_m, siendo m el número de máquinas existentes.

- Restricciones (β). En este trabajo se añaden dos restricciones más a las suposiciones generales descritas anteriormente. Estas restricciones particulares son:

  - *prmu*. Esta restricción solo se considera para el flowshop, e implica que el taller de flujo es de permutación. Esto quiere decir que la secuencia con la que los trabajos son procesados es la misma en todas las máquinas.

  - *nr-pm*. Al realizar tareas de mantenimiento y parar las máquinas, se pierde el trabajo hecho de la tarea interrumpida y, por lo tanto, esta debe comenzar de nuevo cuando se reinicia la máquina. A los periodos de disponibilidad de las máquinas se les denominará *bin*. Su duración, que es conocida y constante, se denota con el parámetro T. Naturalmente, este tiempo debe ser mayor que la duración del tiempo de proceso de cualquier trabajo en cualquier máquina. El tiempo de indisponibilidad de las máquinas se supone también el mismo para todas las paradas e igual a cero, ya que no influye en la elección de la solución óptima del problema.

Con todo esto, el problema de programación de la producción considerado se denota, empleando la notación previamente introducida, como $F_m \mid prmu, nr-pm \mid C_{max}$.

Para clarificar la influencia de esta última restricción y entender mejor en qué consisten los bines, se va a hacer uso de un ejemplo gráfico. Se va a resolver una pequeña instancia de tres máquinas y cinco trabajos. En primer lugar se podrá disponer de las máquinas sin necesidad de pararlas. En el segundo caso las máquinas verán restringida su disponibilidad, con un periodo de funcionamiento entre las paradas de 10 unidades de tiempo (u.t.).

Los tiempos de proceso de los trabajos en las máquinas se recogen en la Tabla 2.1.

	Trabajos				
Máquina	1	2	3	4	5
1	7	8	1	1	7
2	2	3	8	3	5
3	9	7	5	7	6

Tabla 2.1: Tiempos de proceso de una instancia de ejemplo

Sin restricciones de disponibilidad en las máquinas y para, por ejemplo, la secuencia (2, 4, 5, 1, 3) se tiene que el makespan es 45 u.t. El plan de producción se muestra en la Figura 2.1.

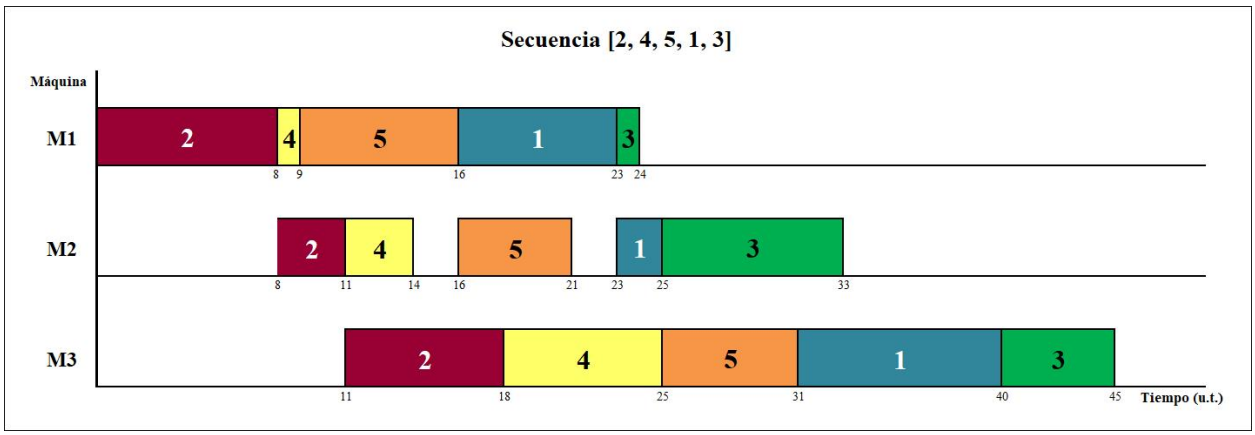

Figura 2.1: Diagrama de Gantt del ejemplo sin restricción de disponibilidad en las máquinas

Al realizar paradas en las máquinas aparecen los bines, aumenta la complejidad del problema y el tiempo requerido para procesar todos los trabajos. La adición de las restricciones periódicas en las máquinas alarga el tiempo de fabricación hasta las 55 u.t., lo que supone aproximadamente un 20 % más de tiempo que en el caso anterior. El plan de producción resultante se muestra en la Figura 2.2.

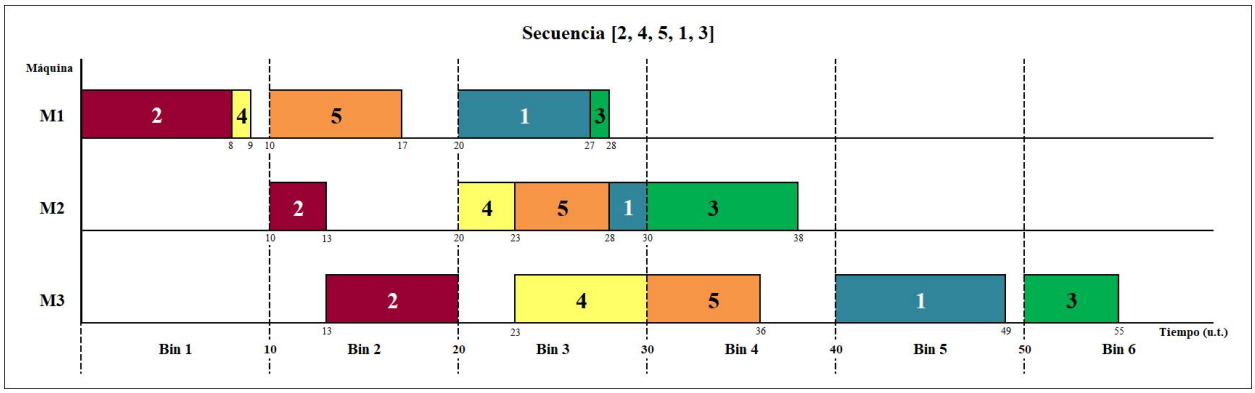

Figura 2.2: Diagrama de Gantt del ejemplo con restricción de disponibilidad en las máquinas

En el gráfico se identifica claramente el bin como los espacios de tiempo existentes entre una parada y otra de la máquina (periodos de disponibilidad). Cada bin tiene, para este caso, una duración de 10 u.t. y una operación de un trabajo en una máquina se inicia en un bin si y solo si se puede terminar de procesar en dicho bin. En caso contrario, se debe programar en el bin siguiente. Por ejemplo, la operación en M1 del trabajo 5 se programa en el Bin 2 porque el Bin 1 tiene disponibilidad de una unidad de tiempo, y sin embargo, la operación que se desea programar necesita 7 u.t. Por esa razón se ha programado en el bin siguiente, ya que no está permitida la interrupción. En total se han requerido 6 bines para llevar a cabo el procesamiento de todos los trabajos.

El problema propuesto se asemeja a otro problema clásico de optimización combinatoria en el que el objetivo es asignar objetos de diferentes medidas en bines de tamaño fijo con el objetivo de maximizar la utilización de los bines.

Para poder abordar el problema anteriormente descrito y diseñar un modelo de programación lineal entera que lo resuelva de forma óptima, se van a revisar los modelos existentes para el problema clásico, denotado $F_m \mid prmu \mid C_{max}$ de acuerdo con la notación introducida anteriormente.

Capítulo 3

Modelización

3.1. Introducción

En este capítulo se presentarán los modelos de programación lineal entera que resuelven de forma óptima el problema clásico, se dará a conocer la notación de los mismos y se indicarán las modificaciones necesarias para adaptarlos en los modelos que resuelvan el problema propuesto $F_m \mid prmu, nr-pm \mid C_{max}$. Los modelos y la notación seguida han sido extraídos de [Stafford et al., 2005], referencia que hace una revisión de los modelos de la literatura.

Para mostrar cómo influye en las máquinas la restricción de disponibilidad, se resolverá una instancia del problema, con un número pequeño de máquinas y trabajos, a modo de ejemplo. La misma instancia será resuelta para el problema clásico, con los modelos originales de la literatura, y con los modelos adaptados para el problema propuesto. Esto permitirá evidenciar, tanto la validación de los modelos, como las diferencias en la solución debidas a la existencia o no de las restricciones en las máquinas.

Hallar la solución óptima a tal problema no es computacionalmente fácil. Conforme aumenta el tamaño del mismo, crece considerablemente el número de posibles soluciones y, por tanto, el tiempo requerido para resolverlo. En un caso de flowshop de permutación, el número de soluciones factibles asciende a $n!$ y es conocido que el problema con el objetivo makespan es *NP-hard*. Se puede probar de forma sencilla que el problema propuesto también lo es, ya que $F_m \mid prmu \mid C_{max}$ es un caso particular de $F_m \mid prmu, nr-pm \mid C_{max}$ cuando T es suficientemente grande.

Para su resolución, se ha utilizado el solver *Gurobi*, un optimizador comercial para programación lineal entera mixta (*MILP*), entre otros modelos de optimización matemática que es capaz de tratar. Los modelos han sido implementados en lenguaje de programación *Python* y para su correcta codificación se ha consultado un tutorial de Python [Rossum and Drake, 2010]. Se ha trabajado sobre una plataforma *PC* con procesador *Intel(R) Core(TM) i5 de 1.80 GHz* y sistema operativo de *64 bits Windows 10 Home* de *Microsoft Corporation*. Las versiones utilizadas de los softwares son: *Gurobi-7.0.2-win64* y *Python - 3.5.2 (64-bit)*.

3.2. Modelos de programación lineal entera

Se partirá de cinco modelos matemáticos distintos capaces de resolver el problema clásico. Los nombres de los modelos son los mismos que dan los autores de [Stafford et al., 2005]. Tres de ellos pertenecen a la familia Wagner, denominados *WST*, *Wilson* y *TS2*. Estos incorporan el problema de asignación, mientras que los otros dos, de la familia Manne, utilizan pares de restricciones dicotómicas, o sus equivalentes matemáticos, para asignar trabajos a posiciones de secuencia: *SGST* y *LYeq*. En los siguientes apartados se definirá la notación empleada para los datos, conjuntos de índices y variables de todos los modelos, de forma que la notación sea uniforme para todos los modelos.

3.2.1. Datos

M, número de máquinas.

N, número de trabajos.

T_{ri}, indica el tiempo que tarda el trabajo i en ser procesado en la máquina r.

P, representa un valor lo suficientemente grande como para asegurar que se cumple solamente una relación del par de restricciones dicotómicas. El valor que se le asigna a P es para cada instancia la suma de los tiempos de proceso de todos los trabajos en todas las máquinas: $P = \sum_{r=1}^{M} \sum_{i=1}^{N} T_{ri}$

T, periodo de disponibilidad. Tiempo constante de funcionamiento entre dos paradas consecutivas en las máquinas.

$Bines$, es una cota superior del número de bines. No es un dato propio del problema, sino un valor que se calcula a partir de los valores de P y T, según $Bines = \lceil \frac{P}{T} \rceil$, es decir, el entero inmediatamente superior al valor que resulta de dividir P entre T.

3.2.2. Conjunto de índices

r, para las máquinas $(1 \leq r \leq M)$

i, k, para los trabajos $(1 \leq i < k \leq N)$

j, para la posición en la secuencia $(1 \leq j \leq N)$

b, para los bines en el horizonte temporal $(1 \leq b \leq Bines)$

3.2.3. Variables

$B_{r,j}$, tiempo de comienzo del trabajo en la posición j de la secuencia en la máquina r $(1 \leq r \leq M, 1 \leq j \leq N)$.

$$Bin_{r,i(j),b} = \begin{cases} 1 & \text{El trabajo i (que está en la posición j de la secuencia)} \\ & \text{es procesado en la máquina r y en el bin b} \\ 0 & \text{En caso contrario} \end{cases}$$

$(1 \leq r \leq M, 1 \leq i \leq N, 1 \leq j \leq N, 1 \leq b \leq Bines)$.

$C_{r,i}$, tiempo de finalización del trabajo i en la máquina r $(1 \leq r \leq M, 1 \leq i \leq N)$.

$$D_{i,k} = \begin{cases} 1 & \text{El trabajo i es procesado antes que el trabajo k} \\ 0 & \text{En caso contrario} \end{cases}$$

$(1 \leq i < k \leq N)$.

$E_{r,j}$, tiempo de finalización del trabajo en la posición j en la máquina r $(1 \leq r \leq M, 1 \leq j \leq N)$.

$S_{r,i}$, tiempo de comienzo del trabajo i en la máquina r $(1 \leq r \leq M, 1 \leq i \leq N)$.

$X_{r,j}$, tiempo ocioso en la máquina r antes del comienzo del procesamiento del trabajo en la posición j de la secuencia ($1 \leq r \leq M, 1 \leq j \leq N$).

$Y_{r,j}$, tiempo de espera del trabajo en la posición j de la secuencia después de ser procesado en la máquina r ($1 \leq r \leq M, 1 \leq j \leq N$).

$$Z_{i,j} = \begin{cases} 1 & \text{el trabajo i es asignado a la posición j de la secuencia} \\ 0 & \text{En caso contrario} \end{cases}$$

($1 \leq i \leq N, 1 \leq j \leq N$).

3.3. Modelos originales y sus adaptaciones

En esta sección se van a presentar los modelos originales del problema clásico, y la adaptación realizada de los mismos añadiendo en cada caso nuevas variables y/o restricciones según corresponda.

Para ilustrar los cambios realizados se hará uso de un ejemplo. A medida que se vayan describiendo los modelos, se resolverá la misma instancia para el problema clásico y para el problema propuesto. Así se podrá contrastar cómo las modificaciones en las restricciones afectan a las soluciones de uno y otro problema.

El ejemplo será una instancia sencilla, con un número reducido de máquinas y trabajos, y unos valores para los tiempos de proceso no muy dispares entre ellos. Cabe decir que, aunque todos los modelos para un mismo problema (clásico o propuesto) sean igualmente válidos y alcancen el mismo valor de la función objetivo, la secuencia que devuelvan podrá ser diferente. El valor mínimo alcanzable del makespan puede estar asociado a más de una secuencia, esto es, un problema con soluciones óptimas alternativas.

Datos

- Número de máquinas, $M = 3$
- Número de trabajos, $N = 6$
- Tiempo de disponibilidad, $T = 10$

- La Tabla 3.1 contiene los tiempos de proceso.

	Trabajos (i)					
Máquina (r)	1	2	3	4	5	6
1	1	5	4	5	8	1
2	2	9	9	3	2	1
3	1	9	7	2	2	3

Tabla 3.1: Tiempos de proceso de ejemplo de instancia para los modelos

3.3.1. La familia de modelos Wagner

En este apartado se trabaja con los tres modelos de la familia Wagner (*WST*, *Wilson* y *TS2*). Se describen en su versión original, se adaptan al problema propuesto y se presenta de forma gráfica la solución proporcionada para el ejemplo anteriormente mencionado.

Modelo WST original

El primer modelo de la familia Wagner utiliza en sus expresiones las variables X_{rj}, Y_{rj} y Z_{ij}.

$$\text{Min } C_{max} = \sum_{i=1}^{N} T_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{3.1}$$

$$\text{sa: } \sum_{j=1}^{N} Z_{ij} = 1 \quad (1 \leq i \leq N) \tag{3.2}$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \quad (1 \leq j \leq N) \tag{3.3}$$

$$\sum_{i=1}^{N} T_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{N} T_{r+1,i} Z_{i,j} + \\ + X_{r+1,j+1} + Y_{r,j} \quad (1 \leq r \leq M-1, 1 \leq j \leq N-1) \tag{3.4}$$

$$X_{r+1,1} = X_{r,1} + Y_{r,1} + \sum_{i=1}^{N} T_{r+1} Z_{i,1} \quad (1 \leq r \leq M-1) \tag{3.5}$$

$$Y_{r,1} = 0 \quad (1 \leq r \leq M-1) \tag{3.6}$$

$$X_{rj}, Y_{rj} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq j \leq N$$
$$Z_{ij} \in \{0,1\}; \quad 1 \leq i \leq N,\ 1 \leq j \leq N$$

El makespan corresponde con el tiempo de finzalización del último trabajo en la última máquina, y viene dado por la suma de los tiempos de proceso y tiempos ociosos que tienen lugar en la última máquina (3.1). Los conjuntos de restricciones (3.2 y 3.3) aseguran, respectivamente, que cada trabajo sea asignado a una única posición en la secuencia y que cada posición de la secuencia esté ocupada por un solo trabajo. Las restricciones (3.4 y 3.5) aseguran que el procesado en la máquina r del trabajo que está en la posición $j+1$ de la secuencia no comience hasta que el

trabajo en la posición j haya terminado su proceso en la misma máquina. Además, garantizan que el trabajo que está en la posición j de la secuencia no empiece a ser procesado en la máquina $r+1$ hasta que haya terminado de ser procesado en la máquina r. Por último, el conjunto de restricciones (3.6) obliga a que el primer trabajo en ser procesado no espere en ninguna máquina. Es decir, una vez el trabajo que está en primera posición de la secuencia termine de ser procesado, este pasará inmediatamente a la siguiente máquina.

Resolución del ejemplo: En la Tabla 3.2 se expone la solución al ejemplo mediante el modelo WST en su versión original. Esta refleja todas las variables necesarias para que la secuencia quede definida. Por ejemplo, la variable $X_{21} = 5$ indica que existen 5 u.t. de tiempo ocioso antes de que el trabajo que está en la posición 1 de la secuencia sea procesado en la máquina 2. $Y_{13} = 5$, significa que el trabajo que ocupa la tercera posición en la secuencia espera durante 5 u.t. después de ser procesado en la máquina 1. En cuanto a las variables binarias Z_{ij}, solo se muestran las que valen 1. El caso de la variable $Z_{61} = 1$ indica que el trabajo 6 es procesado en la posición 1 de la secuencia.

X_{rj}			Y_{rj}		$Z_{ij} = 1$
$X_{11} = 0$	$X_{21} = 5$	$X_{31} = 6$	$Y_{11} = 4$	$Y_{21} = 0$	Z_{16}
$X_{12} = 0$	$X_{22} = 0$	$X_{32} = 6$	$Y_{12} = 0$	$Y_{22} = 0$	Z_{22}
$X_{13} = 0$	$X_{23} = 0$	$X_{33} = 0$	$Y_{13} = 5$	$Y_{23} = 0$	Z_{33}
$X_{14} = 0$	$X_{24} = 0$	$X_{34} = 0$	$Y_{14} = 9$	$Y_{24} = 4$	Z_{44}
$X_{15} = 0$	$X_{25} = 4$	$X_{35} = 0$	$Y_{15} = 8$	$Y_{25} = 0$	Z_{55}
$X_{16} = 9$	$X_{26} = 0$	$X_{36} = 0$	$Y_{16} = 0$	$Y_{26} = 0$	Z_{61}

Tabla 3.2: Solución del ejemplo mediante el modelo WST original

Con la información recabada de las variables, se obtiene fácilmente la secuencia que es la (6, 2, 3, 4, 5, 1) y el valor del makespan, que en este caso vale 36 u.t. La Figura 3.1 muestra el diagrama de Gantt.

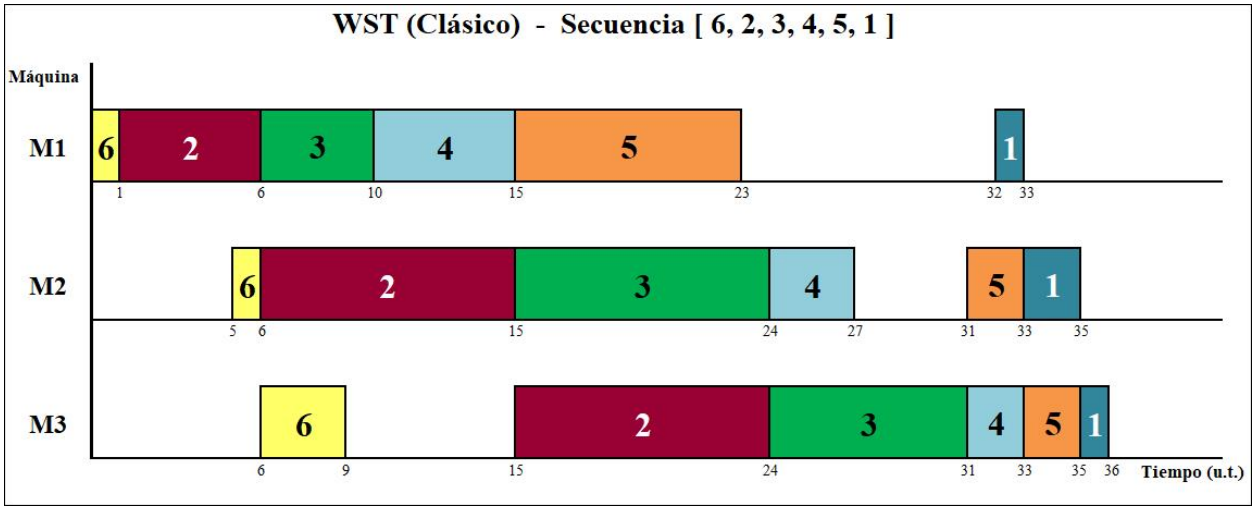

Figura 3.1: D. Gantt de la solución del ejemplo mediante el modelo WST para el problema clásico

Modelo WST adaptado

Una vez analizado con detalle el modelo clásico, se adapta para que resuelva la complejidad añadida de las restricciones en la disponibilidad de las máquinas.

En primer lugar, se han incluido las variables definidas anteriormente como E_{rj} que hacen el modelo más intuitivo. A partir de los tiempos de espera y de proceso se obtiene el tiempo de finalización de los trabajos (3.7). Cada trabajo debe realizarse íntegramente dentro de los límites de un bin, para lo que es necesaria la incorporación de la variable Bin_{rjb}. Se añade el par de restricciones dicotómicas (3.8 y 3.9) al modelo original que establezcan un valor mínimo en el que el trabajo comience, y un valor máximo en el que finalice. Mediante el conjunto de restricciones (3.10) se asegura que un trabajo que es procesado en la máquina r, en la posición j, solo se asigne a un bin.

Siendo l un índice auxiliar para los trabajos:

$$E_{rj} = \sum_{l=1}^{j}(X_{rj} + \sum_{i=1}^{N} T_{ri}Z_{il}) \quad (1 \leq r \leq M, 1 \leq j \leq N), \tag{3.7}$$

$$E_{rj} \leq bT + P(1 - Bin_{rjb}) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines) \tag{3.8}$$

$$E_{rj} - \sum_{i=1}^{N} T_{ri}Z_{ij} + P(1 - Bin_{rjb}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines) \tag{3.9}$$

$$\sum_{b=1}^{Bines} Bin_{rjb} = 1, \quad (1 \leq r \leq M, 1 \leq j \leq N) \tag{3.10}$$

Finalmente, se prescinde del conjunto de restricciones (3.6). Esta restricción fuerza al primer trabajo de la secuencia a no esperar en ninguna máquina después de ser procesado. Sin embargo, en el problema propuesto esto no es posible, ya que las interrupciones debidas a las indisponibilidades hacen que la imposición de este conjunto de restricciones implique infactibilidad.

Con todas estas modificaciones, el modelo WST adaptado al problema propuesto queda:

$$\text{Min } C_{max} = \sum_{i=1}^{N} T_{Mi} + \sum_{p=1}^{N} X_{Mp}$$

$$\text{sa: } \sum_{j=1}^{N} Z_{ij} = 1 \quad (1 \leq i \leq N)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \quad (1 \leq j \leq N)$$

$$\sum_{i=1}^{N} T_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{N} T_{r+1,i} Z_{i,j} +$$

$$+ X_{r+1,j+1} + Y_{r,j} \quad (1 \leq r \leq M-1, 1 \leq j \leq N-1)$$

$$X_{r+1,1} = X_{r,1} + Y_{r,1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \quad (1 \leq r \leq M-1)$$

$$E_{rj} = \sum_{l=1}^{j} (X_{rj} + \sum_{i=1}^{N} T_{ri} Z_{il}) \quad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$E_{rj} \leq bT + P(1 - Bin_{rjb}) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines)$$

$$E_{rj} - \sum_{i=1}^{N} T_{ri} Z_{ij} + P(1 - Bin_{rjb}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines)$$

$$\sum_{b=1}^{Bines} Bin_{rjb} = 1 \quad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$X_{rj}, Y_{rj}, E_{r,j} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq j \leq N$$

$$Z_{ij}, Bin_{rjb} \in \{0,1\}; \quad 1 \leq r \leq M,\ 1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq b \leq Bines$$

Resolución del ejemplo: La solución para el mismo ejemplo por este modelo viene dada en la Tabla 3.3. Solo se hace mención a las variables Bin_{rjb} que tienen valor igual a 1. Por ejemplo, $Bin_{254} = 1$ indica que el trabajo que ocupa la posición 5 de la máquina 2 se procesa en el bin 4.

En la Figura 3.2 se muestra de forma gráfica el plan de producción resultante. La secuencia es (6, 4, 3, 2, 1, 5) y el makespan, 42 unidades. Este valor se corresponde con el de la variable E_{36}, que es el tiempo de finalización del último trabajo en la última máquina. Notar que el valor obtenido es distinto que para el modelo original, que eran 36 u.t. Este incremento es debido a los tiempos ociosos en las máquinas producido por las indisponibilidades de estas.

X_{rj}			Y_{rj}		$Z_{ij}=1$
$X_{11}=0$	$X_{21}=6$	$X_{31}=17$	$Y_{11}=5$	$Y_{21}=10$	Z_{15}
$X_{12}=0$	$X_{22}=0$	$X_{32}=1$	$Y_{12}=1$	$Y_{22}=11$	Z_{24}
$X_{13}=0$	$X_{23}=0$	$X_{33}=0$	$Y_{13}=0$	$Y_{23}=4$	Z_{33}
$X_{14}=4$	$X_{24}=2$	$X_{34}=0$	$Y_{14}=2$	$Y_{24}=0$	Z_{42}
$X_{15}=0$	$X_{25}=0$	$X_{35}=0$	$Y_{15}=10$	$Y_{25}=7$	Z_{56}
$X_{16}=0$	$X_{26}=0$	$X_{36}=0$	$Y_{16}=4$	$Y_{26}=6$	Z_{61}

E_{rj}			$Bin_{rjb}=1$		
$E_{11}=1$	$E_{21}=7$	$E_{31}=20$	Bin_{111}	Bin_{211}	Bin_{312}
$E_{21}=6$	$E_{22}=10$	$E_{32}=23$	Bin_{121}	Bin_{221}	Bin_{323}
$E_{31}=10$	$E_{23}=19$	$E_{33}=30$	Bin_{131}	Bin_{232}	Bin_{333}
$E_{11}=19$	$E_{24}=30$	$E_{34}=39$	Bin_{142}	Bin_{243}	Bin_{344}
$E_{21}=20$	$E_{25}=32$	$E_{35}=40$	Bin_{152}	Bin_{254}	Bin_{354}
$E_{31}=28$	$E_{26}=34$	$E_{36}=42$	Bin_{163}	Bin_{264}	Bin_{365}

Tabla 3.3: Solución del ejemplo mediante el modelo WST adaptado

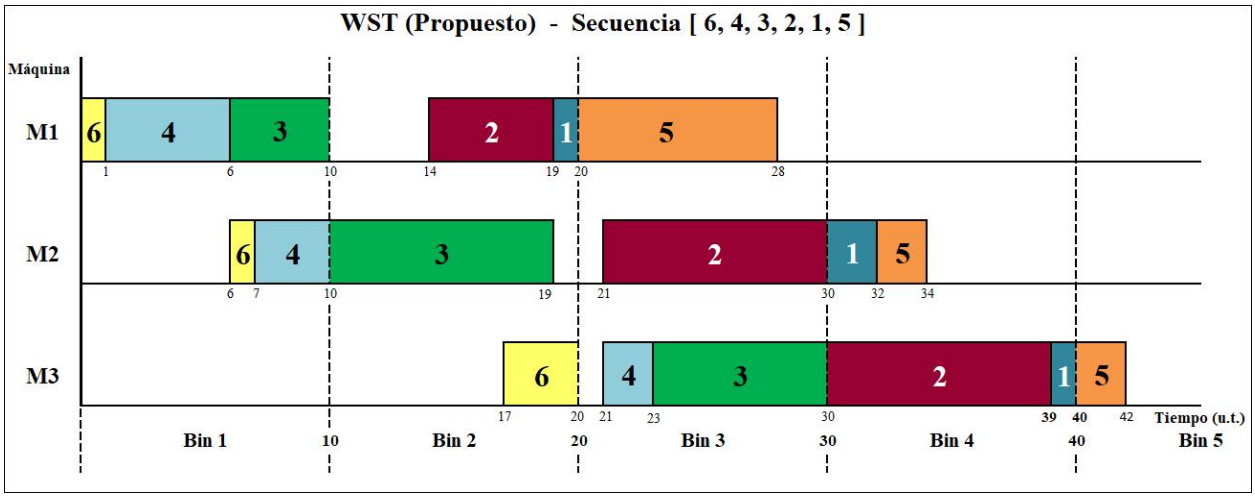

Figura 3.2: D. Gantt de la solución del ejemplo mediante el modelo WST para el problema propuesto

Modelo Wilson original

Las variables que entran en juego en este modelo son B_{rj} y Z_{ij}.

$$\text{Min } B_{MN} + \sum_{i=1}^{N} T_{Mi} Z_{iN} \tag{3.11}$$

$$\text{sa: } \sum_{j=1}^{N} Z_{ij} = 1 \quad (1 \leq i \leq N) \tag{3.12}$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \quad (1 \leq j \leq N) \tag{3.13}$$

$$B_{1j} + \sum_{i=1}^{N} T_{1i} Z_{ij} \leq B_{1,j+1} \quad (1 \leq j \leq N-1) \tag{3.14}$$

$$B_{11} = 0 \tag{3.15}$$

$$B_{r1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \leq B_{r+1,1} \quad (1 \leq r \leq M-1) \tag{3.16}$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \leq B_{r+1,j} \quad (1 \leq r \leq M-1, 2 \leq j \leq N) \tag{3.17}$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \leq B_{r,j+1} \quad (2 \leq r \leq N, 1 \leq j \leq N-1) \tag{3.18}$$

$$B_{rj} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq j \leq N$$

$$Z_{ij} \in \{0,1\}; \quad 1 \leq i \leq N,\ 1 \leq j \leq N$$

El makespan (función objetivo) se expresa como el tiempo de comienzo del último trabajo en la última máquina más su tiempo de proceso (3.11). Los dos primeros conjuntos (3.12 y 3.13) son los (3.2 y 3.3) del modelo WST y tienen la misma interpretación. El grupo (3.14) fuerza que todos los trabajos empiecen a ser procesados en la máquina 1 justo después de que haya terminado el trabajo predecesor. (3.15) restringe el comienzo del primer trabajo en la primera máquina al instante $t = 0$. Por (3.16), el primer trabajo de la secuencia no espera después de ser procesado. Al terminar en una máquina pasa inmediatamente a la siguiente. Por último, (3.17 y 3.18) aseguran, respectivamente, que ningún trabajo puede pasar a la siguiente máquina sin haber sido completamente procesado en la actual y que ningún trabajo puede comenzar a ser procesado en una máquina hasta que el trabajo que le precede haya finalizado en esa misma máquina.

Resolución del ejemplo: Puesto que todos los modelos son igualmente válidos, es esperable que el makespan que alcance el modelo Wilson sea el mismo que el modelo original anterior WST (36 u.t.), aunque no necesariamente con la misma secuencia. Los valores que toman las variables se observan en la Tabla 3.4 y la Gráfica 3.3 muestra la distribución temporal de la planificación. Los trabajos se procesan según la secuencia (6, 2, 3, 5, 1, 4) y el makespan se calcula como el tiempo de comienzo del último trabajo (4) en la última máquina ($E_{36} = 34$) más su tiempo de proceso ($T_{34} = 2$).

B_{rj}			$Z_{ij} = 1$
$B_{11} = 0$	$B_{21} = 1$	$B_{31} = 12$	Z_{15}
$B_{12} = 1$	$B_{22} = 6$	$B_{32} = 15$	Z_{22}
$B_{13} = 6$	$B_{23} = 15$	$B_{33} = 24$	Z_{33}
$B_{14} = 10$	$B_{24} = 25$	$B_{34} = 31$	Z_{46}
$B_{15} = 18$	$B_{25} = 29$	$B_{35} = 33$	Z_{54}
$B_{16} = 26$	$B_{26} = 31$	$B_{36} = 34$	Z_{61}

Tabla 3.4: Solución del ejemplo mediante el modelo Wilson original

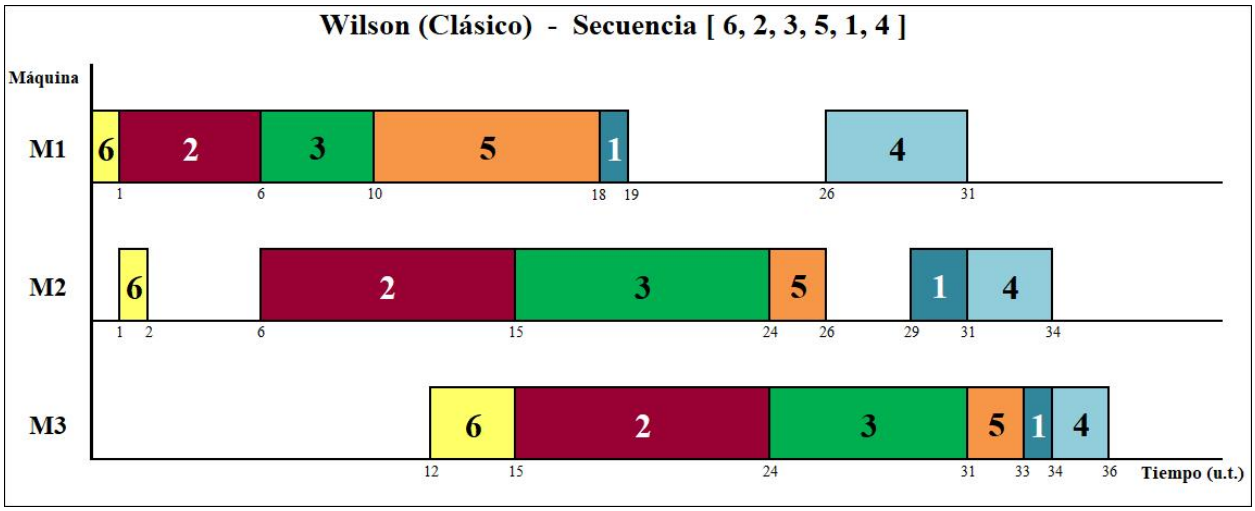

Figura 3.3: D. Gantt de la solución del ejemplo mediante el modelo Wilson para el problema clásico

Modelo Wilson adaptado

Para modificar el modelo original, se añaden las restricciones (3.19, 3.20 y 3.21). Estas requieren variables diferentes a las utilizadas en el WST, aunque tienen la misma interpretación.

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} - bT \leq P(1 - Bin_{rjb}) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines) \tag{3.19}$$

$$B_{rj} + P(1 - Bin_{rjb}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines) \tag{3.20}$$

$$\sum_{b=1}^{Bines} Bin_{rjb} = 1 \quad (1 \leq r \leq M, 1 \leq j \leq N) \tag{3.21}$$

El modelo Wilson modificado resulta ser:

$$\text{Min } B_{MN} + \sum_{i=1}^{N} T_{Mi} Z_{iN}$$

$$\text{sa: } \sum_{j=1}^{N} Z_{ij} = 1 \quad (1 \leq i \leq N)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \quad (1 \leq j \leq N)$$

$$B_{1j} + \sum_{i=1}^{N} T_{1i} Z_{ij} \leq B_{1,j+1} \quad (1 \leq j \leq N-1)$$

$$B_{11} = 0$$

$$B_{r1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \leq B_{r+1,1} \quad (1 \leq r \leq M-1)$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \leq B_{r+1,j} \quad (1 \leq r \leq M-1, 2 \leq j \leq N)$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \leq B_{r,j+1} \quad (2 \leq r \leq N, 1 \leq j \leq N-1)$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} - bT \leq P(1 - Bin_{rjb}) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines)$$

$$B_{rj} + P(1 - Bin_{rjb}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines)$$

$$\sum_{b=1}^{Bines} Bin_{rjb} = 1 \quad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$B_{rj} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq j \leq N$$

$$Z_{ij}, Bin_{rjb} \in \{0,1\}; \quad 1 \leq r \leq M,\ 1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq b \leq Bines$$

Resolución del ejemplo: Los valores obtenidos por el modelo para las variables se reflejan en la Tabla 3.5. De nuevo, el makespan es el tiempo de comienzo del último trabajo (5) en la última máquina ($B_{36} = 40$) más lo que tarda este en ser procesado ($T_{35} = 2$), es decir, 42 unidades de tiempo. Interpretando los valores obtenidos de las variables se obtiene una secuencia (6, 1, 2, 3, 4, 5) y se elabora el Gráfico 3.4.

B_{rj}			$Z_{ij} = 1$	$Bin_{rjb} = 1$		
$B_{11} = 0$	$B_{21} = 4$	$B_{31} = 10$	Z_{12}	Bin_{111}	Bin_{211}	Bin_{312}
$B_{12} = 1$	$B_{22} = 5$	$B_{32} = 14$	Z_{23}	Bin_{121}	Bin_{221}	Bin_{322}
$B_{13} = 2$	$B_{23} = 10$	$B_{33} = 20$	Z_{34}	Bin_{131}	Bin_{232}	Bin_{333}
$B_{14} = 10$	$B_{24} = 20$	$B_{34} = 30$	Z_{45}	Bin_{142}	Bin_{243}	Bin_{344}
$B_{15} = 14$	$B_{25} = 30$	$B_{35} = 37$	Z_{56}	Bin_{152}	Bin_{254}	Bin_{354}
$B_{16} = 20$	$B_{26} = 33$	$B_{36} = 40$	Z_{61}	Bin_{163}	Bin_{264}	Bin_{365}

Tabla 3.5: Solución del ejemplo mediante el modelo Wilson adaptado

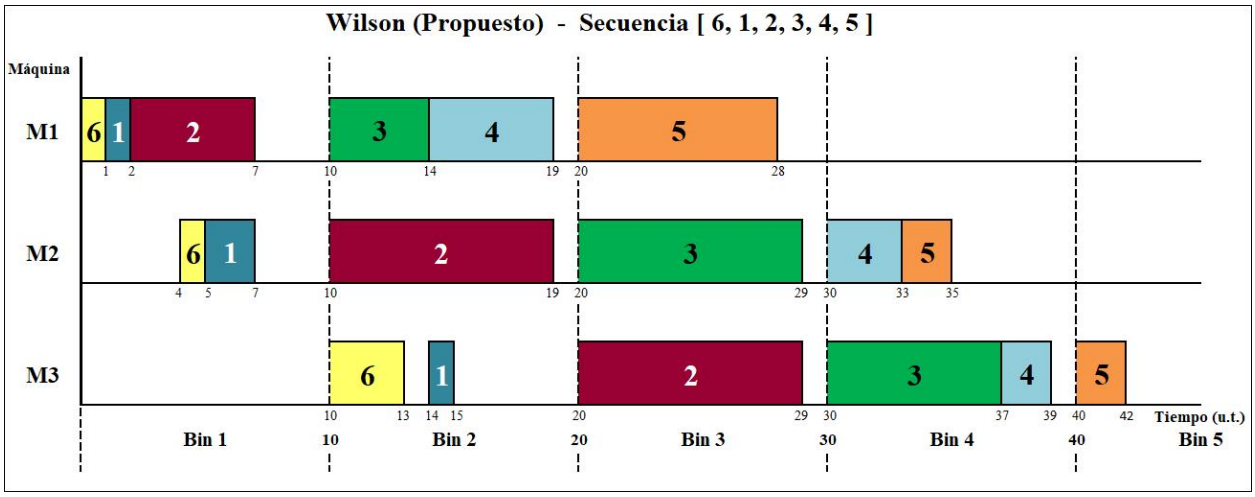

Figura 3.4: D. Gantt de la solución del ejemplo mediante el modelo Wilson para el problema propuesto

Modelo TS2 original

Las restricciones del modelo TS2 son similares a las del modelo anterior, pero en el modelo de Wilson la variable que se utilizaba era el tiempo de comienzo de los trabajos (B_{rj}) y aquí se emplea el de finalización, E_{rj}.

$$\text{Min } E_{MN} \tag{3.22}$$

$$\text{sa: } \sum_{j=1}^{N} Z_{ij} = 1 \quad (1 \leq i \leq N) \tag{3.23}$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \quad (1 \leq j \leq N) \tag{3.24}$$

$$E_{rj} + \sum_{i=1}^{N} T_{ri} Z_{i,j+1} \leq E_{r,j+1} \quad (1 \leq r \leq M, 1 \leq j \leq N-1) \tag{3.25}$$

$$E_{rj} + \sum_{i=1}^{N} T_{r+1,i} Z_{ij} \leq E_{r+1,j} \quad (1 \leq r \leq M-1, 1 \leq j \leq N) \tag{3.26}$$

$$E_{11} \geq \sum_{i=1}^{N} T_{1i} Z_{i1} \tag{3.27}$$

$$E_{rj} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq j \leq N$$

$$Z_{ij} \in \{0, 1\}; \quad 1 \leq i \leq N,\ 1 \leq j \leq N$$

Las restricciones (3.25) aseguran que el trabajo procesado en la posición $j+1$ de la secuencia no pueda terminar en ninguna máquina hasta que su predecesor en la secuencia y el propio trabajo en la posición $j+1$ se procesen completamente en esa máquina. El conjunto (3.26) garantiza que un trabajo dado no puede terminar en la máquina $r+1$ hasta que finalice en la máquina anterior y luego se procese completamente en la máquina $r+1$. Por último, la inecuación (3.27) fuerza que el primer trabajo de la secuencia no pueda terminar en la máquina 1 antes de su tiempo de proceso.

Resolución del ejemplo: Para el problema clásico, la solución se refleja en la Tabla 3.6. La secuencia de los trabajos es (1, 6, 2, 3, 4, 5) y el makespan, el tiempo de finalización del último trabajo en la tercera máquina: $E_{36} = 36$ u.t. La solución gráfica se muestra en la Figura 3.5.

B_{rj}			$Z_{ij} = 1$
$E_{11} = 1$	$E_{21} = 3$	$E_{31} = 13$	Z_{11}
$E_{12} = 2$	$E_{22} = 7$	$E_{32} = 16$	Z_{23}
$E_{13} = 7$	$E_{23} = 16$	$E_{33} = 25$	Z_{34}
$E_{14} = 11$	$E_{24} = 25$	$E_{34} = 32$	Z_{45}
$E_{15} = 16$	$E_{25} = 32$	$E_{35} = 34$	Z_{56}
$E_{16} = 24$	$E_{26} = 34$	$E_{36} = 36$	Z_{62}

Tabla 3.6: Solución del ejemplo mediante el modelo TS2 original

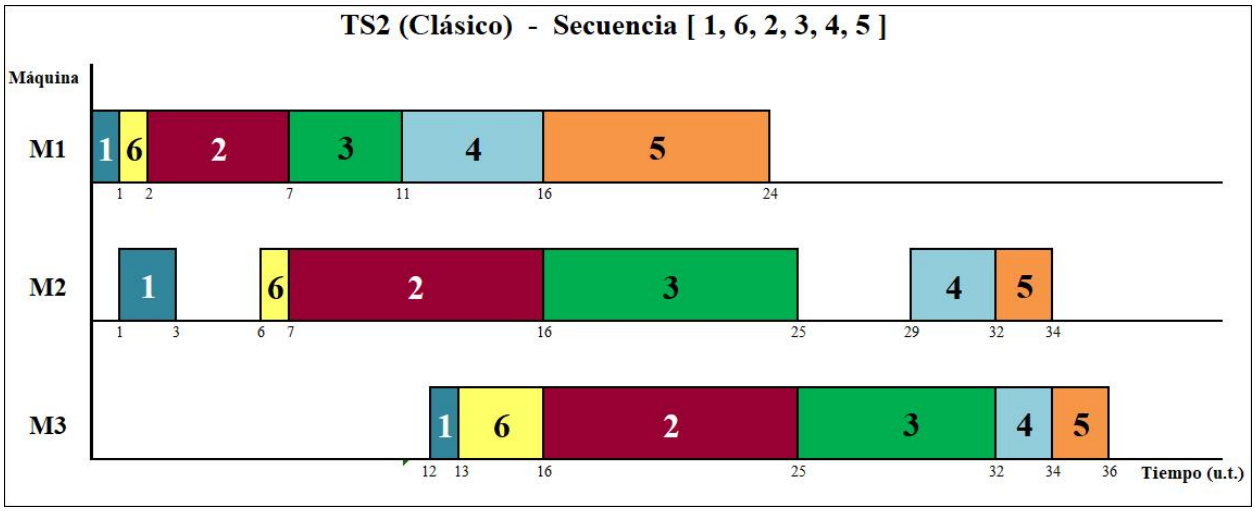

Figura 3.5: D. Gantt de la solución del ejemplo mediante el modelo TS2 para el problema clásico

Modelo TS2 adaptado

Para adaptar el modelo al problema propuesto, se añaden las expresiones que garantizan el cumplimiento de los límites de los bines. Dado que el modelo TS2 y el WST trabajan con la variable E_{rj}, estas expresiones son las mismas (3.8, 3.9 y 3.10).

La adaptación del modelo TS2 queda:

$$\text{Min } E_{MN}$$

$$\text{sa: } \sum_{j=1}^{N} Z_{ij} = 1 \quad (1 \leq i \leq N)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \quad (1 \leq j \leq N)$$

$$E_{rj} + \sum_{i=1}^{N} T_{ri} Z_{i,j+1} \leq E_{r,j+1} \quad (1 \leq r \leq M, 1 \leq j \leq N-1)$$

$$E_{rj} + \sum_{i=1}^{N} T_{r+1,i} Z_{ij} \leq E_{r+1,j} \quad (1 \leq r \leq M-1, 1 \leq j \leq N)$$

$$E_{11} \geq \sum_{i=1}^{N} T_{1i} Z_{i1}$$

$$E_{rj} - bT \leq P(1 - Bin_{rjb}) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines)$$

$$E_{rj} - \sum_{i=1}^{N} T_{ri} Z_{ij} + P(1 - Bin_{rjb}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq j \leq N, 1 \leq b \leq Bines)$$

$$\sum_{b=1}^{Bines} Bin_{rjb} = 1 \qquad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$E_{rj} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq j \leq N$$

$$Z_{ij}, Bin_{rjb} \in \{0,1\}; \quad 1 \leq r \leq M,\ 1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq b \leq Bines$$

Resolución del ejemplo: Al resolver el problema propuesto con el modelo TS2, las variables resultan según la Tabla 3.7. La secuencia obtenida es (6, 2, 3, 5, 1, 4) y el makespan, 42 unidades. Trasladando los resultados numéricos, se obtiene el Gráfico 3.6.

B_{rj}			$Z_{ij} = 1$	$Bin_{rjb} = 1$		
$E_{11} = 1$	$E_{21} = 2$	$E_{31} = 10$	Z_{15}	Bin_{111}	Bin_{211}	Bin_{311}
$E_{12} = 6$	$E_{22} = 20$	$E_{32} = 30$	Z_{22}	Bin_{121}	Bin_{222}	Bin_{323}
$E_{13} = 10$	$E_{23} = 30$	$E_{33} = 37$	Z_{33}	Bin_{131}	Bin_{233}	Bin_{334}
$E_{14} = 18$	$E_{24} = 32$	$E_{34} = 39$	Z_{46}	Bin_{142}	Bin_{244}	Bin_{344}
$E_{15} = 20$	$E_{25} = 34$	$E_{35} = 40$	Z_{54}	Bin_{152}	Bin_{254}	Bin_{354}
$E_{16} = 30$	$E_{26} = 40$	$E_{36} = 42$	Z_{61}	Bin_{163}	Bin_{264}	Bin_{365}

Tabla 3.7: Solución del ejemplo mediante el modelo TS2 adaptado

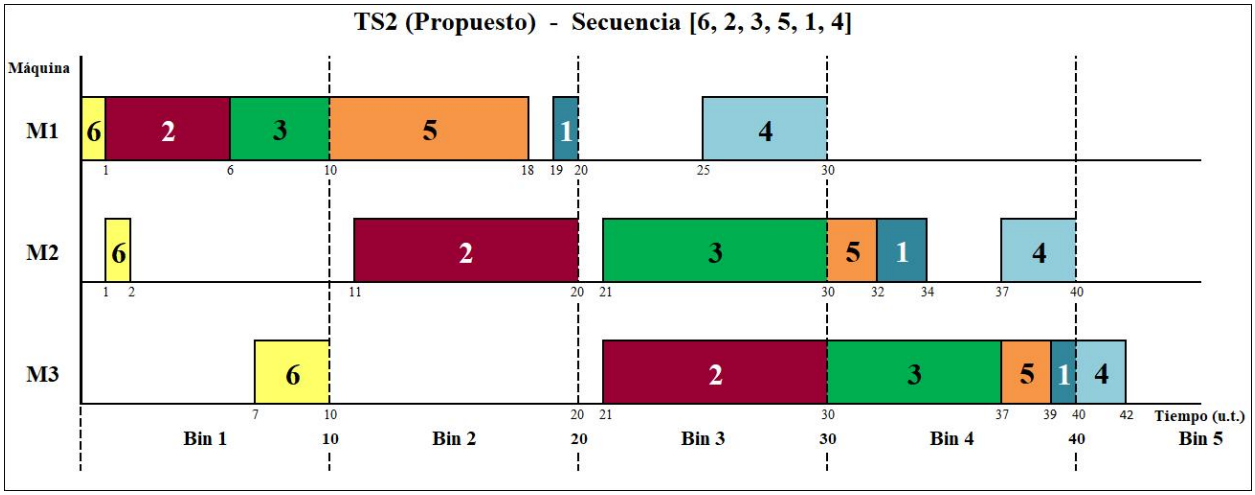

Figura 3.6: D. Gantt de la solución del ejemplo mediante el modelo TS2 para el problema propuesto

3.3.2. La familia de modelos Manne

Los dos modelos que van a ser expuestos a continuación (*SGST* y *LYeq*) pertenecen a la familia de modelos Manne. Estos utilizan, para la asignación de trabajos a las posiciones de la secuencia de procesamiento, pares de restricciones dicotómicas. Además, a diferencia de la familia Wagner, las variables no se refieren a los trabajos como su posición en la secuencia (subíndice j), sino que se hace referencia al trabajo en sí (subíndice i).

Modelo SGST original

El primero de los modelos de la familia Manne utiliza las variables C_{ri} y D_{ik}.

$$\text{Min } C_{max} \tag{3.28}$$

$$\text{sa: } C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \tag{3.29}$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1, 1 \leq i \leq N) \tag{3.30}$$

$$C_{ri} - C_{rk} + PD_{ik} \geq T_{ri} \quad (1 \leq r \leq M, 1 \leq i < k \leq N) \tag{3.31}$$

$$C_{rk} - C_{ri} + PD_{ik} \geq T_{rk} \quad (1 \leq r \leq M, 1 \leq i < k \leq N) \tag{3.32}$$

$$C_{max} \geq C_{Mi} \quad (1 \leq i \leq N) \tag{3.33}$$

$$C_{ri}, C_{max} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq i \leq N$$
$$D_{ij} \in \{0, 1\}; \quad 1 \leq i \leq N,\ 1 \leq j \leq N$$

En este modelo, el makespan se expresa directamente como C_{max} (3.28), ya que es una variable en el modelo. En cuanto a las restricciones, el primer conjunto (3.29) garantiza que ningún trabajo pueda terminar en la primera máquina antes de lo que tarda en ser procesado. Por las restricciones (3.30), ningún trabajo puede terminar en la máquina $r+1$ hasta que haya finalizado en la máquina anterior y, luego, haya sido completamente procesado en la máquina actual. Las expresiones (3.31 y 3.32) forman los pares de restricciones dicotómicas que relacionan cada par de trabajos i y k ($i \neq k$), de modo que cualquiera de los trabajos i o bien precede, o bien sigue, al trabajo k en la secuencia, pero no ambos. Por último, (3.33) fuerza a que el makespan (C_{max}) coincida con el tiempo de finalización del último de los trabajos procesados en la última máquina.

Resolución del ejemplo: Los valores que toman las variables al ejecutar la instancia ejemplo con el modelo SGST en su versión original se muestran en la Tabla 3.8.

C_{ri}			D_{ik}		
$C_{11} = 1$	$C_{21} = 3$	$C_{31} = 4$	$D_{12} = 1$	$D_{23} = 1$	$D_{35} = 1$
$C_{12} = 7$	$C_{22} = 16$	$C_{32} = 25$	$D_{13} = 1$	$D_{24} = 1$	$D_{36} = 0$
$C_{13} = 11$	$C_{23} = 25$	$C_{33} = 32$	$D_{14} = 1$	$D_{25} = 1$	$D_{45} = 1$
$C_{14} = 16$	$C_{24} = 28$	$C_{34} = 34$	$D_{15} = 1$	$D_{26} = 0$	$D_{46} = 0$
$C_{15} = 32$	$C_{25} = 34$	$C_{35} = 36$	$D_{16} = 1$	$D_{34} = 1$	$D_{56} = 0$
$C_{16} = 2$	$C_{26} = 4$	$C_{36} = 16$			

Tabla 3.8: Solución del ejemplo mediante el modelo SGST original

La diferencia más llamativa en estos modelos respecto a los de la familia Wagner es el tratamiento de los subíndices. Ahora, a los trabajos se les hace referencia por su número, no por el orden en el que son procesados. De esta manera, la variable $C_{16} = 2$ indica el tiempo de terminación del trabajo 6 (no el que es procesado en sexto lugar de la secuencia) en la máquina 1. Las variables binarias D_{ik} proporcionan información sobre el orden en que los trabajos son procesados. Por ejemplo, $D_{34} = 1$ indica que el trabajo 4 es procesado después del 3, mientras que $D_{36} = 0$ indica que el trabajo 6 se procesa antes que el 3.

Con ayuda de las variables D_{ik} se identifica al trabajo 5 como el último en ser procesado. Por ello, el makespan coincide con el tiempo de finalización de este trabajo: 36 u.t. La secuencia es (1, 6, 2, 3, 4, 5) y el programa el que aparece en la Figura 3.7.

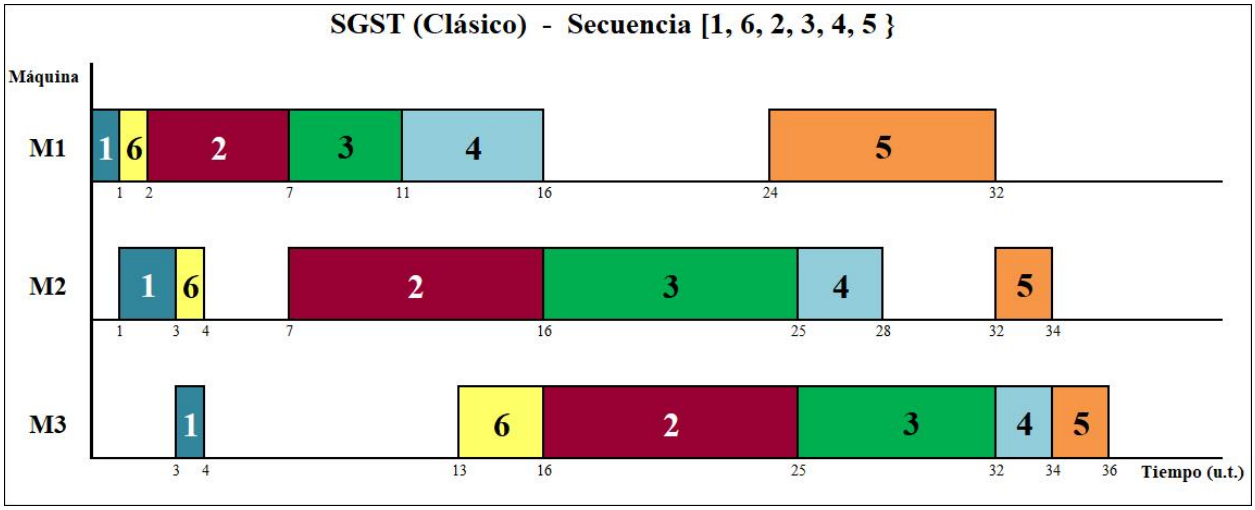

Figura 3.7: D. Gantt de la solución del ejemplo mediante el modelo SGST para el problema clásico

Modelo SGST adaptado

Para adaptar este modelo a la versión restrictiva, se sigue la misma pauta que en los modelos de la familia Wagner. Como se ha dicho anteriormente, en los modelos de la familia Manne las expresiones hacen referencia a los trabajos en sí, no a la posición que ocupan en la secuencia de procesamiento. Por eso, el subíndice i sustituye al j en la variable Bin_{rib} (3.34, 3.35 y 3.36).

$$C_{ri} - bT \leq P(1 - Bin_{rib}) \quad (1 \leq r \leq M, 1 \leq i \leq N, 1 \leq b \leq Bines) \tag{3.34}$$

$$C_{ri} - T_{ri} + P(1 - Bin_{rib}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq i \leq N, 1 \leq b \leq Bines) \tag{3.35}$$

$$\sum_{b=1}^{Bines} Bin_{rib} = 1 \quad (1 \leq r \leq M, 1 \leq i \leq N) \tag{3.36}$$

Con la adición de estas tres restricciones, el modelo SGST adaptado al problema propuesto resulta:

$$\text{Min } C_{max}$$

$$\text{sa: } C_{1i} \geq T_{1i} \quad (1 \leq i \leq N)$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1, 1 \leq i \leq N)$$

$$C_{ri} - C_{rk} + PD_{ik} \geq T_{ri} \quad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

$$C_{rk} - C_{ri} + PD_{ik} \geq T_{rk} \quad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

$$C_{max} \geq C_{Mi} \quad (1 \leq i \leq N)$$

$$C_{ri} - bT \leq P(1 - Bin_{rib})$$

$$(1 \leq r \leq M, 1 \leq i \leq N, 1 \leq b \leq Bines)$$

$$C_{ri} - T_{ri} + P(1 - Bin_{rib}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq i \leq N, 1 \leq b \leq Bines)$$

$$\sum_{b=1}^{Bines} Bin_{rib} = 1 \quad (1 \leq r \leq M, 1 \leq i \leq N)$$

$$C_{ri}, C_{max} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq i \leq N$$

$$D_{ij}, Bin_{rib} \in \{0,1\}; \quad 1 \leq r \leq M,\ 1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq b \leq Bines$$

Resolución del ejemplo: La Tabla 3.9 y la Figura 3.8 muestran la resolución del modelo SGST adaptado. La secuencia obtenida es (1, 6, 2, 3, 5, 4) y el makespan = $C_{34} = 42$ u.t.

C_{ri}			D_{ik}		
$C_{11} = 1$	$C_{21} = 3$	$C_{31} = 6$	$D_{12} = 1$	$D_{23} = 1$	$D_{35} = 1$
$C_{12} = 10$	$C_{22} = 20$	$C_{32} = 30$	$D_{13} = 1$	$D_{24} = 1$	$D_{36} = 0$
$C_{13} = 14$	$C_{23} = 29$	$C_{33} = 37$	$D_{14} = 1$	$D_{25} = 1$	$D_{45} = 0$
$C_{14} = 35$	$C_{24} = 40$	$C_{34} = 42$	$D_{15} = 1$	$D_{26} = 0$	$D_{46} = 0$
$C_{15} = 30$	$C_{25} = 37$	$C_{35} = 40$	$D_{16} = 1$	$D_{34} = 1$	$D_{56} = 0$
$C_{16} = 5$	$C_{26} = 6$	$C_{36} = 10$			

$Bin_{rib} = 1$		
Bin_{111}	Bin_{211}	Bin_{311}
Bin_{121}	Bin_{222}	Bin_{323}
Bin_{132}	Bin_{233}	Bin_{334}
Bin_{144}	Bin_{244}	Bin_{345}
Bin_{153}	Bin_{254}	Bin_{354}
Bin_{161}	Bin_{261}	Bin_{361}

Tabla 3.9: Solución del ejemplo mediante el modelo SGST adaptado

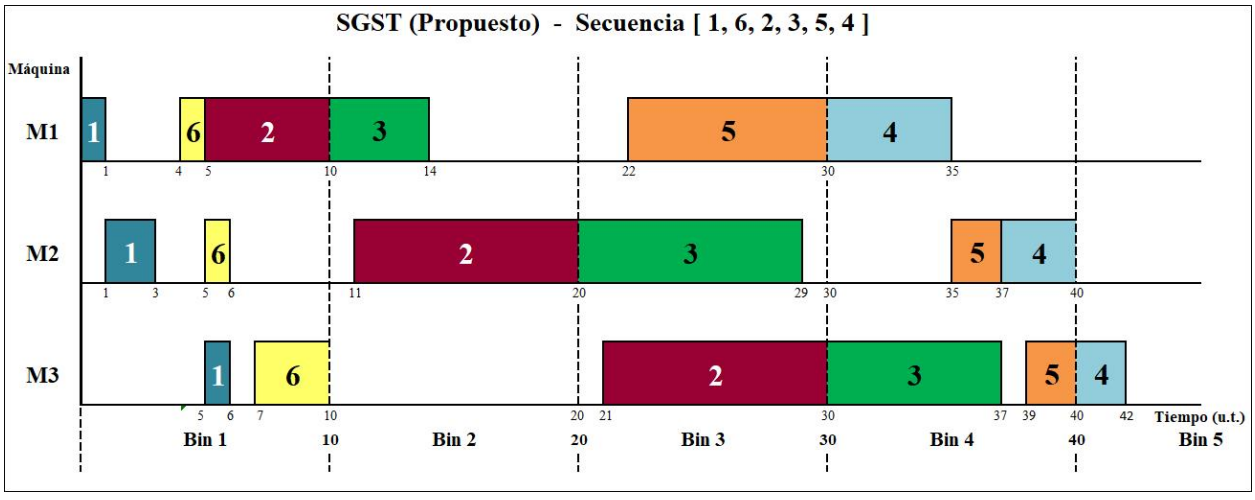

Figura 3.8: D. Gantt de la solución del ejemplo mediante el modelo SGST para el problema propuesto

Modelo LYeq original

En este modelo LYeq, al igual que en el caso anterior, están presentes las variables C_{ri}y D_{ik}, además de la variable auxiliar Q_{rik}.

$$\text{Min } C_{max} \tag{3.37}$$

$$\text{sa: } C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \tag{3.38}$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1, 1 \leq i \leq N) \tag{3.39}$$

$$C_{max} \geq C_{Mi} \quad (1 \leq i \leq N) \tag{3.40}$$

$$PD_{ik} + C_{ri} - C_{rk} - T_{ri} = Q_{rik} \quad (1 \leq r \leq M, 1 \leq i < k \leq N) \tag{3.41}$$

$$Q_{rik} \leq P - T_{ri} - T_{rk} \quad (1 \leq r \leq M, 1 \leq i < k \leq N) \tag{3.42}$$

$$C_{ri}, C_{max}, Q_{rik} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq i < k \leq N$$
$$D_{ij} \in \{0,1\}; \quad 1 \leq i \leq N,\ 1 \leq j \leq N$$

La función objetivo (3.37) y las restricciones (3.38, 3.39 y 3.40) son las expresiones (3.28, 3.29, 3.30 y 3.33), respectivamente, del modelo SGST. Las expresiones (3.41 y 3.42) establecen el par de restricciones dicotómicas.

Resolución del ejemplo: En la Tabla 3.10 se muestra la solución que devuelve el modelo LYeq original a la instancia del ejemplo. A partir de los valores de estas variables, se obtiene la secuencia (6, 2, 3, 1, 5, 4) y un makespan de 36 u.t. (C_{34}). El gráfico con el plan de producción se muestra en la Figura 3.9.

C_{ri}			D_{ik}		
$C_{11} = 11$	$C_{21} = 26$	$C_{31} = 32$	$D_{12} = 0$	$D_{23} = 1$	$D_{35} = 1$
$C_{12} = 6$	$C_{22} = 15$	$C_{32} = 24$	$D_{13} = 0$	$D_{24} = 1$	$D_{36} = 0$
$C_{13} = 10$	$C_{23} = 24$	$C_{33} = 31$	$D_{14} = 1$	$D_{25} = 1$	$D_{45} = 0$
$C_{14} = 24$	$C_{24} = 31$	$C_{34} = 36$	$D_{15} = 1$	$D_{26} = 0$	$D_{46} = 0$
$C_{15} = 19$	$C_{25} = 28$	$C_{35} = 34$	$D_{16} = 0$	$D_{34} = 1$	$D_{56} = 0$
$C_{16} = 1$	$C_{26} = 6$	$C_{36} = 15$			

Tabla 3.10: Solución del ejemplo mediante el modelo LYeq original

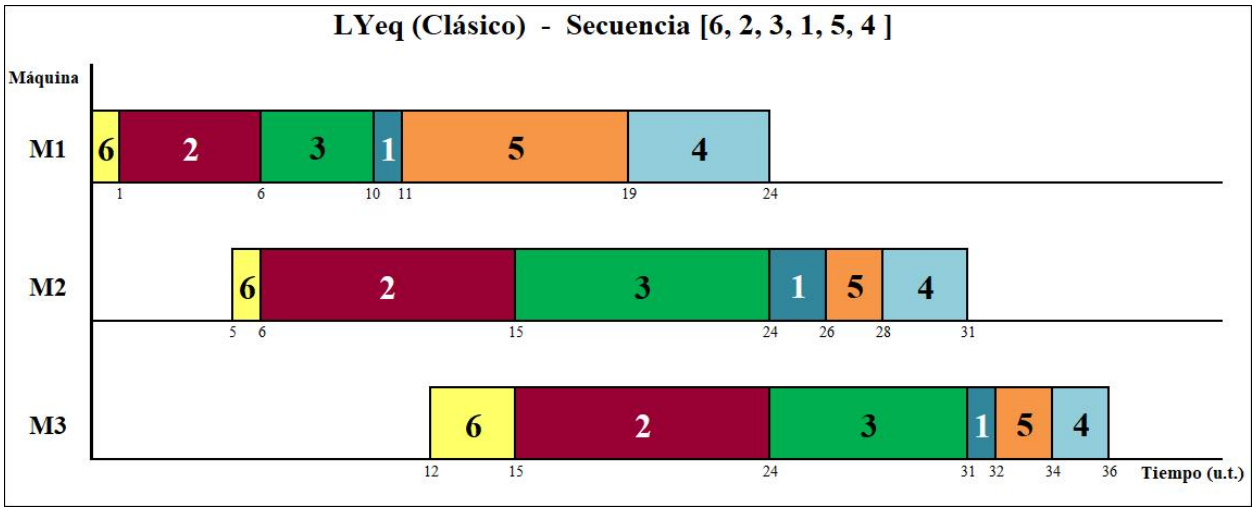

Figura 3.9: D. Gantt de la solución del ejemplo mediante el modelo LYeq para el problema clásico

Modelo LYeq adaptado

Dado que los modelos SGST y LYeq trabajan con las mismas variables, las restricciones que se agregan en ambos modelos para limitar la disponibilidad en las máquinas son las mismas (3.34, 3.35 y 3.36).

Así, el modelo LYeq modificado, queda:

$$\text{Min } C_{max}$$

$$\text{sa: } C_{1i} \geq T_{1i} \quad (1 \leq i \leq N)$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1, 1 \leq i \leq N)$$

$$C_{max} \geq C_{Mi} \quad (1 \leq i \leq N)$$

$$PD_{ik} + C_{ri} - C_{rk} - T_{ri} = Q_{rik} \quad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

$$Q_{rik} \leq P - T_{ri} - T_{rk} \quad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

$$C_{ri} - bT \leq P(1 - Bin_{rib}) \quad (1 \leq r \leq M, 1 \leq i \leq N, 1 \leq b \leq Bines)$$

$$C_{ri} - T_{ri} + P(1 - Bin_{rib}) \geq T(b-1) \quad (1 \leq r \leq M, 1 \leq i \leq N, 1 \leq b \leq Bines)$$

$$\sum_{b=1}^{Bines} Bin_{rib} = 1 \quad (1 \leq r \leq M, 1 \leq i \leq N)$$

$$C_{ri}, C_{max}, Q_{rik} \geq 0; \quad 1 \leq r \leq M,\ 1 \leq i < k \leq N$$

$$D_{ij}, Bin_{rib} \in \{0,1\}; \quad 1 \leq r \leq M,\ 1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq b \leq Bines$$

Resolución del ejemplo: El modelo LYeq propone la solución que aparece en la Tabla 3.11 al problema propuesto. Asignando los trabajos en las máquinas y en el tiempo se construye la secuencia (6, 2, 3, 4, 1, 5) y el Diagrama 3.10. El trabajo 5 es el último en la secuencia, por lo que el makespan coincide con C_{35} y vale 42 unidades.

C_{ri}			D_{ik}		
$C_{11} = 20$	$C_{21} = 35$	$C_{31} = 40$	$D_{12} = 0$	$D_{23} = 1$	$D_{35} = 1$
$C_{12} = 6$	$C_{22} = 19$	$C_{32} = 29$	$D_{13} = 0$	$D_{24} = 1$	$D_{36} = 0$
$C_{13} = 14$	$C_{23} = 29$	$C_{33} = 37$	$D_{14} = 0$	$D_{25} = 1$	$D_{45} = 1$
$C_{14} = 19$	$C_{24} = 33$	$C_{34} = 39$	$D_{15} = 1$	$D_{26} = 0$	$D_{46} = 0$
$C_{15} = 28$	$C_{25} = 37$	$C_{35} = 42$	$D_{16} = 0$	$D_{34} = 1$	$D_{56} = 0$
$C_{16} = 1$	$C_{26} = 10$	$C_{36} = 14$			

$Bin_{rib} = 1$		
Bin_{112}	Bin_{214}	Bin_{314}
Bin_{121}	Bin_{222}	Bin_{323}
Bin_{132}	Bin_{233}	Bin_{334}
Bin_{142}	Bin_{244}	Bin_{344}
Bin_{153}	Bin_{254}	Bin_{355}
Bin_{161}	Bin_{261}	Bin_{362}

Tabla 3.11: Solución del ejemplo mediante el modelo LYeq adaptado

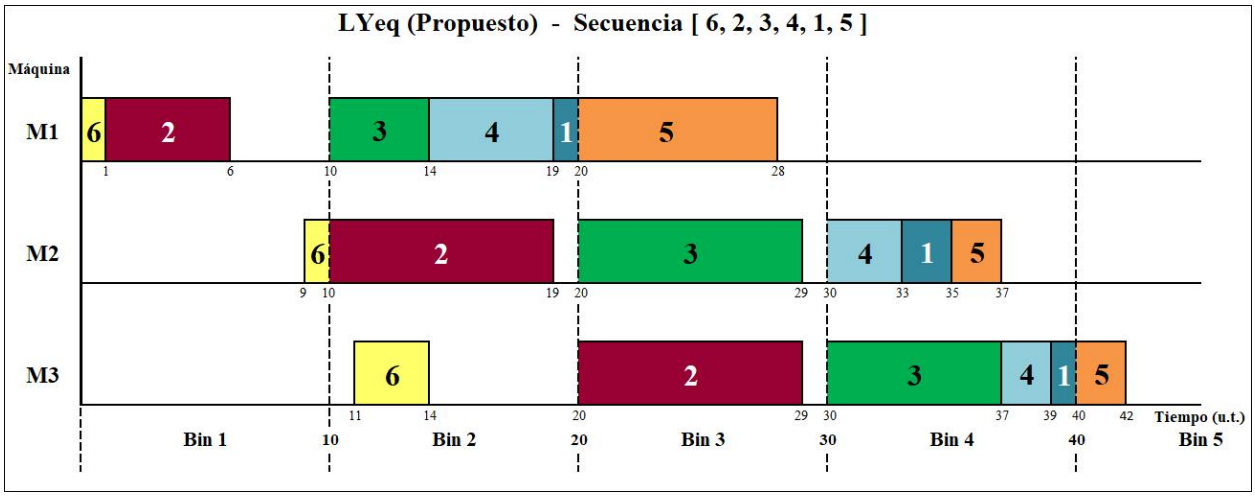

Figura 3.10: D. Gantt de la solución del ejemplo mediante el modelo LYeq para el problema propuesto

Una vez han sido adaptados los cinco modelos para que cumplan correctamente las restricciones de disponibilidad en las máquinas y se ha comprobado que resuelven el problema de manera óptima, interesa determinar cuál de todos ellos es más eficiente en términos de tiempo de resolución.

Para ello se llevará a cabo un análisis computacional, en el que cada modelo resolverá la misma batería de problemas y se compararán los tiempos de cómputo. Las conclusiones de este estudio se expone en el siguiente capítulo.

Capítulo 4

Resultados y análisis

4.1. Resolución de los modelos

Se dispone de cinco modelos para resolver el problema que es objeto final de este proyecto. Una instancia del problema puede ser resuelta por un modelo en cuestión de segundos, mientras que otro modelo puede no llegar a encontrar la solución óptima para ese mismo caso en varios minutos de cómputo.

En función del tamaño y la complejidad del problema que se enfrente, será más interesante emplear un modelo u otro. Decidir qué modelo es capaz de resolver un determinado problema más rápidamente es el objeto del análisis que se va a realizar.

Cada modelo resolverá una batería de 480 problemas, que difieren en el número de máquinas (M), trabajos (N) y periodo de disponibilidad (T) que los caracterizan. Se harán combinaciones de problemas que tengan 2, 3, 4 y 5 máquinas, que procesen, cada uno de ellos, 5, 10, 15 y 20 trabajos y con un tiempo de disponibilidad que varíe entre 100, 200 y 300 u.t. Además, para cada combinación concreta se resolverán 10 instancias distintas, cada una de ellas con tiempos de proceso generados aleatoriamente según una distribución uniforme discreta entre 1 y 99 u.t.

$$T_{ri} \sim U[1, 99]$$

Es esperable que, cuando crezca la complejidad de los problemas al aumentar el número de máquinas y/o trabajos, el tiempo de cómputo requerido para hallar la solución óptima al problema aumente también. Para evitar tiempos de ejecución extremadamente largo, se ha limitado el tiempo de ejecución para resolver cada instancia a un máximo de 15 minutos ($900s$). Si, para un problema, el modelo no ha alcanzado la solución óptima en ese tiempo, este devolverá la mejor solución factible que haya encontrado hasta entonces.

En previsión de que algunos modelos no alcancen si quiera una solución factible en esos 15 minutos, se proporciona una solución admisible de partida (por lejos que se halle esta del óptimo).Esta solución inicial es la correspondiente a la secuencia $[1, ..., N]$.

Ejecutar una batería de 480 instancias mediante los 5 modelos que se han desarrollado se traduce en la resolución de un total de 2400 problemas diferentes. Su ejecución se ha llevado a cabo en las mismas condiciones con las que se ha trabajado en el ejemplo durante el capítulo 3: lenguaje de programación *Python v. 3.5.2 (64-bit)* y optimizador *Gurobi-7.0.2-win64*. Plataforma *PC* con procesador *Intel(R) Core(TM) i5 de 1.80 GHz* y sistema operativo de *64 bits Windows 10 Home* de *Microsoft Corporation*.

Los resultados del experimento se han recogido en un archivo "*.txt*". De ahí se han exportado a un archivo *excel* y se han clasificado según el número de máquinas M, trabajos N, periodo de disponibilidad T y tiempo de cómputo (*CPU Time*). En la Figura 4.1 se muestra un ejemplo del formato del archivo de salida.

```
Wilson   M 3  N 15  T 200  Cmax 918.0  Runtime 402.29244804382324
Wilson   M 3  N 15  T 200  Cmax 909.0  Runtime 5.524028778076172
Wilson   M 3  N 15  T 200  Cmax 934.0  Runtime 151.83449745178223
Wilson   M 3  N 15  T 200  Cmax 813.0  Runtime 59.617177963256836
Wilson   M 3  N 15  T 200  Cmax 772.0  Runtime 190.47104835510254
Wilson   M 3  N 15  T 200  Cmax 916.0  Runtime 605.886173248291
Wilson   M 3  N 15  T 200  Cmax 718.0  Runtime 3.900053024291992
Wilson   M 3  N 15  T 200  Cmax 861.0  Runtime 9.479927062988281
Wilson   M 3  N 15  T 200  Cmax 817.0  Runtime 28.824195861816406
Wilson   M 3  N 15  T 200  Cmax 954.0  Runtime 7.249412536621094
```

Figura 4.1: Formato de salida de los resultados.

4.2. Análisis estadístico de resultados

En este apartado se va a realizar un análisis a partir de los resultados obtenidos del experimento, en total 2400 observaciones, con el objeto de determinar qué modelo es el más rápido en media a la hora de resolver las instancias del problema propuesto.

El tiempo de cómputo (*CPU Time*), que es la variable objeto de estudio, guarda relación de dependencia respecto a cada una de las variables independientes o factores utilizados en el experimento: Modelo, Número de máquinas (M), Número de trabajos (N) y Periodo de disponibilidad (T). Es interesante estudiar cómo influyen los modelos en el tiempo de cómputo y cómo responde cada uno de ellos en función de los distintos niveles o valores que puedan tomar los factores.

Para la realización de cálculos se ha utilizado el software de análisis estadístico y gráfico de datos *SPSS*, desarrollado por *IBM Corporation*. Si se desea observar con más detalle la salida de resultados del experimento y los datos facilitados por el software *SPSS*, se puede consultar el archivo *Salida y análisis estadístico de los resultados.xlsx* adjunto en la versión digital de esta memoria.

CPU Time vs Modelos

En primer lugar se analiza la influencia de los distintos modelos en el tiempo de cómputo. Se realiza un estudio por parejas, comparando el valor medio de *CPU Time* y sus respectivos niveles de incertidumbre, correspondientes al nivel de confianza del 95 %, para los distintos valores que puede tomar el factor, es decir, los modelos *WST*, *Wilson*, *TS2*, *SGST* y *LYeq*. Se emplea el Test LSD (*Least Significant Difference*) de Fisher, que determina la menor diferencia que puede haber entre cada pareja de medias para que pueda decirse que existe una diferencia estadísticamente significativa entre ellas [Montgomery and Runger, 2002].

La Tabla 4.1 muestra, para cada modelo, el valor medio de *CPU Time* total (considerando todos los valores de los factores M, N, y T) y su respectiva desviación estándar. En el Gráfico 4.2 se representan estos valores medios junto con sus respectivos intervalos de confianza al 95 %.

CPU Time (s)									
WST		**Wilson**		**TS2**		**SGST**		**LYeq**	
Media	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**
324,82	401,03	268,27	388,56	256,17	384,89	412,18	432,82	428,56	438,73

Tabla 4.1: *CPU Time* medio para cada modelo

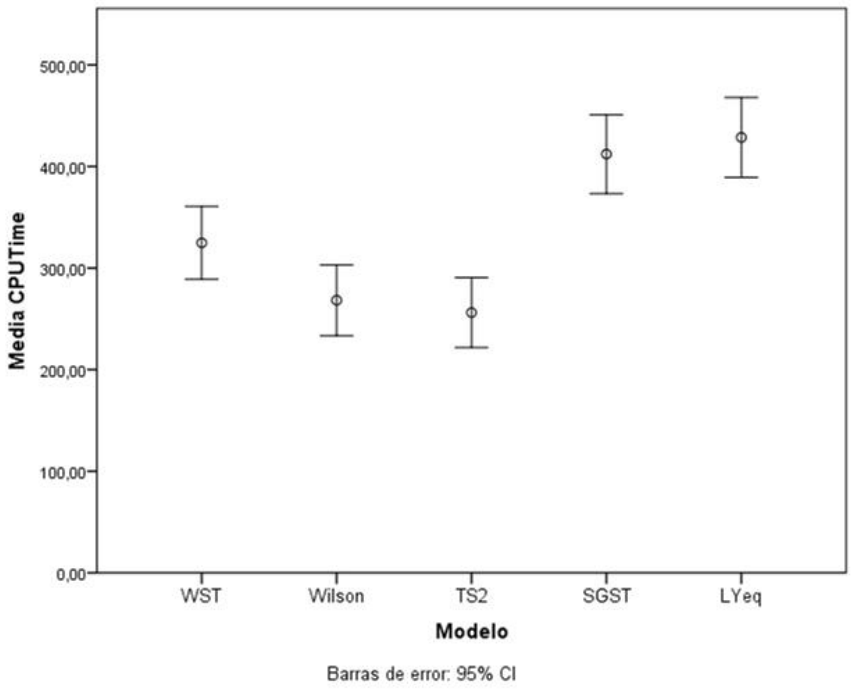

Figura 4.2: Intervalo de confianza (95 %) del *CPU Time* medio para cada modelo

De su análisis se obtienen algunas conclusiones:

- Al nivel de confianza establecido (95 %), las medias de los modelos LYeq y SGST se superponen, luego se puede decir que no existe una diferencia estadísticamente significativa entre las medias de ambos. Lo mismo se puede afirmar del conjunto de modelos WST, Wilson y TS2. Sin embargo, los dos grupos mencionados no se superponen entre sí, lo que indica que la pareja LYeq-SGST y el conjunto WST-Wilson-TS2 son estadísticamente diferentes.

- Atendiendo al valor medio de *CPU Time*, se concluye que los modelos de la familia Wagner (WST, Wilson y TS2) son más eficientes que los de la familia Manne (SGST y LYeq). Dentro de la familia Wagner, el modelo TS2 ha resuelto más rápido, en media, todas las instancias del experimento, seguido de los modelos Wilson y WST, respectivamente. El que más tiempo de cómputo ha requerido ha sido el modelo LYeq.

- Respecto a la dispersión de los datos, los modelos TS2 y Wilson son los que tienen menor valor de la desviación estándar. No solo son los modelos que resuelven las instancias en menor tiempo medio de cómputo, sino que además son los que presentan una menor incertidumbre en los resultados.

Una vez se ha concluído qué modelo es el más rápido en resolver todas las instancias, resulta interesante profundizar en este análisis y desglosar los valores de los tiempos de cómputo medios para cada modelo y cada factor. Conocer cómo influye en el tiempo de cómputo, y para cada modelo, el número de trabajos (N), el número de máquinas (M) y el periodo de disponibilidad (T), sirve para determinar qué modelo es más conveniente emplear en cada caso.

CPU Time vs Modelo y Número de Máquinas

La Tabla 4.2 muestra la media y la variabilidad, medida en términos de la desviación típica, de los tiempos de cómputo empleados por cada modelo para resolver las instancias con 2, 3, 4 y 5 máquinas, respectivamente. Trasladando esos valores a un gráfico resulta la Figura 4.3.

	CPU Time (s)									
	WST		**Wilson**		**TS2**		**SGST**		**LYeq**	
	Media	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**
M=2	191,27	346,12	180,77	344,88	179,34	346,84	405,60	436,10	411,09	439,89
M=3	273,32	384,14	231,95	372,21	215,51	365,32	414,04	437,81	425,52	440,65
M=4	367,50	403,70	290,55	392,61	283,33	394,07	415,73	434,02	438,11	440,63
M=5	467,21	417,24	369,80	419,72	346,49	413,12	413,35	428,69	439,55	438,69

Tabla 4.2: *CPU Time* medio en función del número de máquinas, para cada modelo

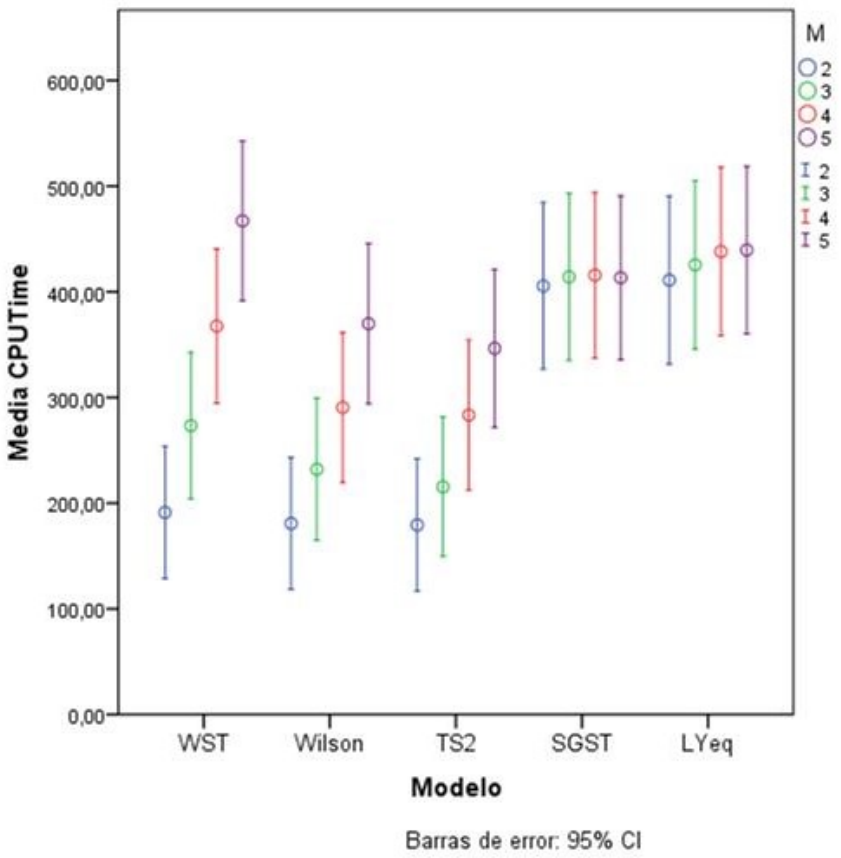

Figura 4.3: Intervalo de confianza (95 %) del *CPU Time* medio en función de M, para cada modelo

De su observación se extraen las siguientes conclusiones:

- *CPU Time* depende del factor M aproximadamente de forma lineal. Según crece el número de máquinas del problema, aumenta también el tiempo requerido para resolverlo. En la familia de modelos Wagner, la razón de proporcionalidad es mucho mayor que en la familia Manne.

- Para valores pequeños de M ($M = 2$ y $M = 3$) los modelos de la familia Wagner tardan la mitad de tiempo aproximadamente que lo que tardan los de la familia Manne. Dentro de la familia Wagner, el modelo más rápido es el TS2, seguido de los modelos Wilson y WST.

- A medida que aumenta M, la diferencia de *CPU Time* entre las familias se reduce. Al elevar el número de máquinas ($M = 4$ y $M = 5$), el tiempo requerido por los modelos WST, Wilson y TS2 (Wagner) aumenta considerablemente, mientras que los modelos SGST y LYeq (Manne) no tardan mucho más que para valores de M pequeños.

- Para $M = 5$, el modelo WST es el que requiere más *CPU Time* de media. Podría ser interesante ampliar el estudio aumentando el número de máquinas para verificar si los modelos de la familia Manne acaban siendo más rápidos que los de la Wagner y para qué valor de M comenzaría a ser más conveniente emplear el modelo SGST (el más rápido de la familia Manne).

- Para los modelos de la familia Wagner, la dispersión de los datos crece conforme aumenta el número de máquinas. Un entorno productivo con un número de máquinas elevado no solo requiere más tiempo de computación, sino que supone una mayor desviación de los valores de *CPU Time* respecto a la media. En cambio, para los modelos de la familia Manne, y para los valores de M analizados, los intervalos de confianza permanecen prácticamente invariables.

- Por último, los valores de la desviación estándar para los modelos de la familia Manne, aunque constantes, siempre son superiores a los de la familia Wagner. Esto supone que los tiempos de cómputo medios requeridos por los modelos SGST y LYeq son más inciertos que los que requieren los modelos WST, Wilson y TS2.

CPU Time vs Modelo y Número de Trabajos

La Tabla 4.3 muestra la media y la desviación estándar de los tiempos de cómputo empleados por cada modelo para resolver las instancias con 5, 10, 15 y 20 trabajos, respectivamente. La Figura 4.4 muestra estos valores medios junto con sus respectivos intervalos de confianza al 95 %.

	CPU Time (s)									
	WST		**Wilson**		**TS2**		**SGST**		**LYeq**	
	Media	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**
N=5	1,38	1,88	0,28	0,31	0,25	0,35	0,19	0,19	0,31	0,26
N=10	80,76	174,08	20,56	73,51	15,69	44,17	11,03	15,57	14,93	20,48
N=15	502,99	396,26	385,52	401,08	354,12	397,28	737,48	271,45	798,96	233,23
N=20	714,16	331,06	666,72	364,97	654,60	377,37	900,03	0,03	900,05	0,14

Tabla 4.3: *CPU Time* medio en función del número de trabajos, para cada modelo

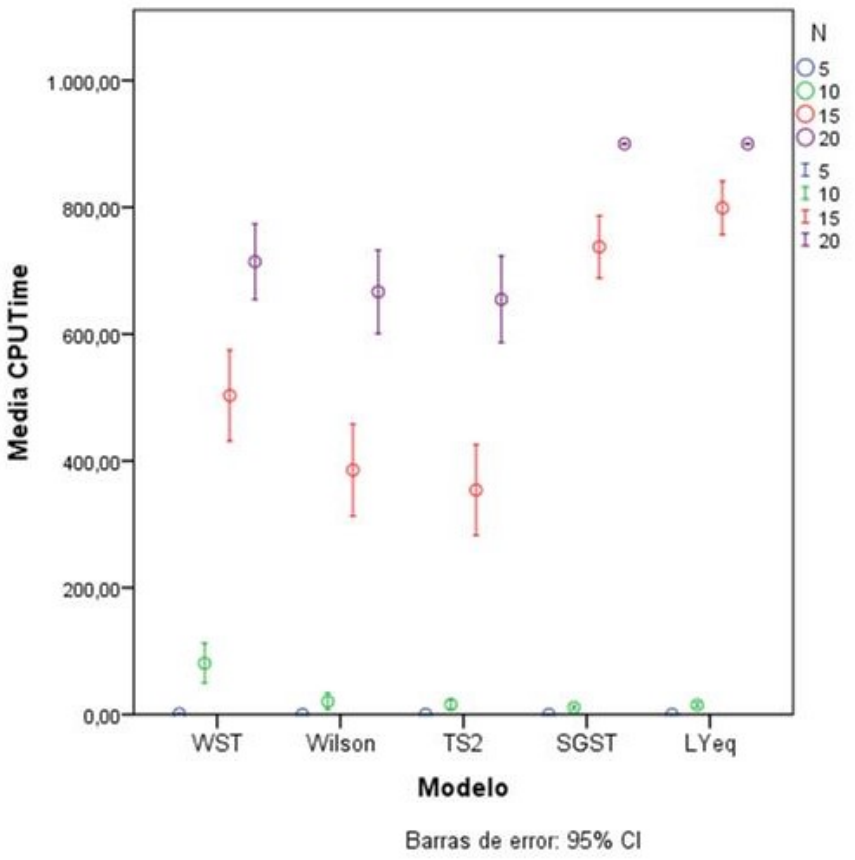

Figura 4.4: Intervalo de confianza (95 %) del *CPU Time* medio en función de N, para cada modelo

Conclusiones:

- El número de trabajos es el factor al que la variable *CPU Time* presenta una mayor sensibilidad. Esta crece exponencialmente con N, debido al carácter *NP-hard* del problema.

- Cuando el problema es simple, $N \leq 10$, el tiempo de resolución requerido por los modelos es muy pequeño: los modelos Wilson, TS2, SGST y LYeq emplean entre $0,2s$ y $0,3s$ para $N = 5$ y entre $10s$ y $20s$ para $N = 10$. El modelo SGST (Manne) es ligeramente más rápido. Sin embargo, el modelo WST puede requerir hasta cinco veces más tiempo que los demás, por lo que se concluye que no es conveniente emplear este modelo para valores bajos de N.

- Con un número más elevado de trabajos, $N \geq 15$, surge una diferencia considerable entre los *CPU Time* de los modelos de la familia Wagner y los de la Manne. Los modelos SGST y LYeq no encuentran la solución óptima para muchos problemas con 15 trabajos y para ningún

problema cuando $N = 20$. Por esta razón, el *CPU Time* medio para los modelos de la familia Manne resulta $900s$, el límite de tiempo de cómputo máximo impuesto. El mejor modelo de la familia Wagner vuelve a ser el TS2.

- Por lo general, la incertidumbre de los tiempos de cómputo crece para todos los modelos a medida que aumenta el número de trabajos que se procesan. Para valores intermedios de N ($N = 10$ y $N = 15$), los tiempos de cómputo requeridos por los modelos SGST y LYeq tienen una menor dispersión respecto a su media que los modelos de la familia Wagner. El modelo WST es el que presenta mayor incertidumbre en los valores de *CPU Time*.

- Para $N = 20$, la desviación estándar de los tiempos de cómputo se reduce. Esto se debe al límite impuesto al tiempo de computación: los modelos no son capaces de resolver muchas de las instancias más complejas (elevado número de trabajos) en un tiempo admisible y el *CPU Time* que devuelven es el límite impuesto de $900s$. Llama la atención que la desviación estándar de los *CPU Time* de los dos modelos de la familia Manne, SGST y LYeq, es prácticamente nula al no encontrar en el tiempo establecido la solución óptima para ninguna de las instancias con 20 trabajos.

CPU Time vs Modelo y Periodo de Disponibilidad

La Tabla 4.4 muestra la media y la dispersión de los tiempos de cómputo empleados por cada modelo para resolver las instancias con un periodo de disponibilidad de 100, 200 y 300 u.t, respectivamente. Los mismos datos se pueden observar gráficamente en la Figura 4.5.

	CPU Time (s)									
	WST		**Wilson**		**TS2**		**SGST**		**LYeq**	
	Media	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**	**Media**	**Desv. Est.**
T=100	481,94	422,34	435,88	436,47	432,41	435,28	420,32	435,57	431,80	440,68
T=200	330,31	407,70	261,77	383,12	244,90	377,83	381,96	420,79	414,98	435,27
T=300	162,23	297,39	107,16	251,24	91,19	235,15	434,26	442,84	438,91	442,64

Tabla 4.4: *CPU Time* medio en función del periodo de disponibilidad, para cada modelo

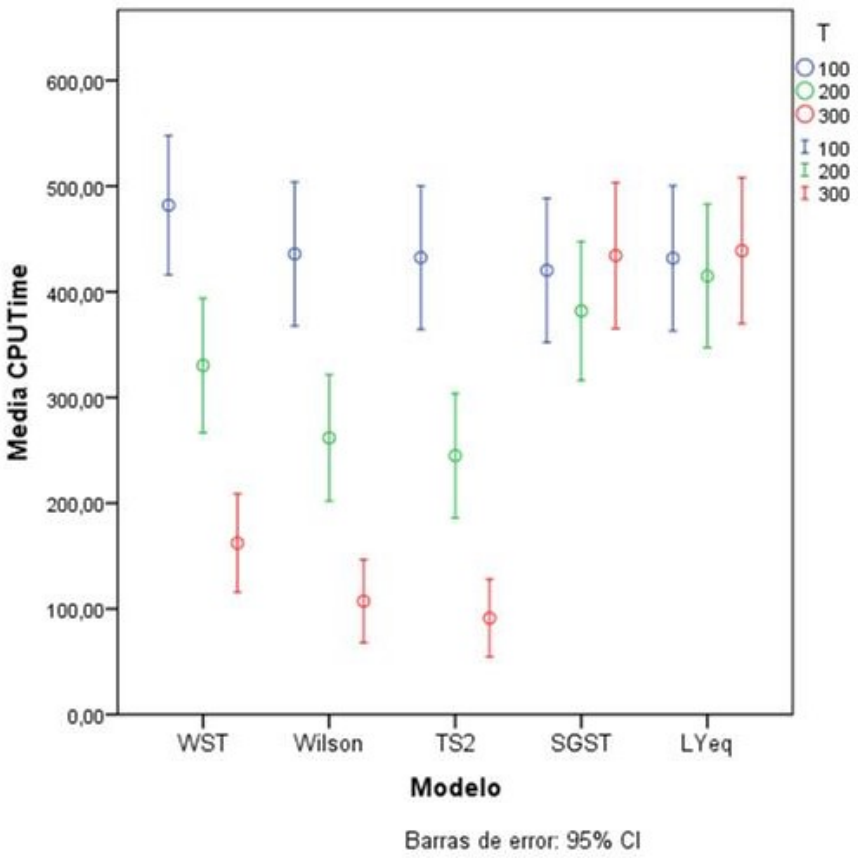

Figura 4.5: Intervalo de confianza (95 %) del *CPU Time* medio en función de T, para cada modelo

Se concluye:

- La relación entre *CPU Time* y T es inversa y aproximadamente lineal. El tiempo de cómputo se reduce con el aumento del periodo de disponibilidad. Este resultado es predecible, y nos hace ver que, mientras mayor es T (menos bines se necesitan), más fácil es para el *solver* hallar la solución óptima. El problema clásico se podría considerar como un tipo especial del problema propuesto en el que el periodo de disponibilidad T es infinito. Así, una conclusión directa es que encontrar una solución óptima del problema propuesto puede ser más difícil que para el problema clásico.
- Para $T = 100$, todos los modelos tardan aproximadamente lo mismo, en media, en resolver las instancias del problema propuesto. El modelo SGST es ligeramente más rápido que los demás.

- A medida que crece el periodo de disponibilidad, el tiempo de cómputo se reduce hasta una cuarta parte para los modelos de la familia Wagner, mientras que para los de la familia Manne permanece prácticamente invariable. Para valores altos del periodo de disponibilidad ($T = 200$ y $T = 300$), el modelo TS2 es el más rápido.
- Para los modelos WST, Wilson y TS2, la variabilidad se reduce a medida que crece el periodo de disponibilidad. La incertidumbre sobre los tiempos de cómputo para los modelos SGST y LYeq, aunque permanece prácticamente constante para los valores de T, es superior, por lo general, a la de los modelos de la familia Wagner.

Con las observaciones realizadas, se concluye que el modelo TS2 de la familia Wagner es el más rápido en la mayoría de los casos, especialmente para problemas complejos (número de trabajos y máquinas altos) y con periodos de disponibilidad también elevados. Solo ante problemas más simples (valores bajos de M y N) y un periodo de disponibilidad reducido sería más conveniente utilizar el modelo SGST, que es el más rápido de la familia Manne.

También los valores de la dispersión analizados invitan a utilizar los modelos de la familia Wagner. Aunque la incertidumbre de los valores de *CPU Time* de los modelos SGST y LYeq respecto a su media no varía considerablemente con los distintos niveles de los factores, esta es, en la mayoría de los casos, superior a la de los modelos WST, Wilson y TS2.

Capítulo 5

Conclusiones

Este proyecto tiene como objeto final el diseño de un modelo de programación lineal entera capaz de hallar, en el menor tiempo posible, la solución óptima a un problema de programación de la producción con máquinas en serie (flowshop), periodos de indisponibilidad en las mismas y objetivo el minimizar el tiempo total de producción (makespan).

Para ello, se analizan cinco modelos de la literatura para un problema similar, aunque sin la disponibilidad restringida en las máquinas, que se denomina problema "clásico" y se adaptan al problema objeto del proyecto: problema "propuesto". Los nombres de los modelos son los mismos que dan los autores de [Stafford et al., 2005]: WST, Wilson y TS2 (pertenecientes a la familia de modelos Wagner) y los modelos SGST y LYeq (familia Manne).

Los modelos obtenidos son implementados en lenguaje de programación *Python* y se validan resolviendo problemas pequeños mediante el solver *Gurobi*. Los cinco modelos se ejecutan para una batería de 2400 instancias del problema propuesto para obtener los tiempos de cómputo y determinar, tras un análisis estadístico, qué modelo es más eficiente para el problema propuesto.

Del estudio se concluye que el modelo TS2 de la familia Wagner es el más rápido en la mayoría de los casos, especialmente para problemas complejos (número de trabajos y máquinas altos) y con periodos de disponibilidad también elevados. Solo ante problemas más simples (valores bajos de M y N) y un periodo de disponibilidad reducido sería más conveniente utilizar el modelo SGST, que es el más rápido de la familia Manne.

Desde el punto de vista de la dispersión de los tiempos de cómputo, también es más adecuado, en general, emplear los modelos de la familia Wagner.

En [Stafford et al., 2005] también se realiza un análisis comparativo de los tiempos de cómputo de los modelos originales que resuelven el problema clásico. De él se obtiene que, en general, los modelos de la familia Wagner son mejores que los de la familia Manne. El modelo Wilson es más rápido que los modelos WST y TS2 para problemas complejos (más de diez máquinas). Para problemas con más de diez trabajos, no hay diferencia significativa entre los tres modelos.

Se prueba que, tanto para el problema clásico como para el propuesto, los modelos de la familia Wagner requieren menos tiempo computacional que cualquiera de los de la familia Manne y esta diferencia crece considerablemente a medida que aumenta el tamaño del problema (aumenta el número de máquinas y de trabajos). Por esta razón, los modelos Wagner son preferibles para encontrar soluciones óptimas al problema de flowshop con permutación y objetivo el minimizar el tiempo de fabricación.

Bibliografía

[Framinan et al., 2014] Framinan, J. M., Leisten, R., and Ruiz García, R. (2014). *Manufacturing Scheduling Systems*, volume 9781447162.

[Graham et al., 1979] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Annals of Discrete Mathematics*.

[Montgomery and Runger, 2002] Montgomery, D. C. and Runger, G. G. (2002). *Applied Statistics and Probability for Engineers*. Wiley.

[Pinedo, 2005] Pinedo, M. L. (2005). *Planning and Scheduling in Manufacturing and Services*.

[Rossum and Drake, 2010] Rossum, G. V. and Drake, F. L. (2010). Python Tutorial. *History*, 42(4):1–122.

[Stafford et al., 2005] Stafford, E. F., Tseng, F. T., and Gupta, J. N. D. (2005). Comparative evaluation of MILP flowshop models. *Journal of the Operational Research Society*, 56(1):88–101.

Anexo

A continuación se adjunta el código fuente en lenguaje *Python* que implementa cada uno de los cinco modelos desarrollados que resuelven de forma exacta el problema propuesto.

1. Modelo WST

```
from gurobipy import *
import sys

f=open(sys.argv[1],'r')
M=f.readline()
N=f.readline()
ciclo=f.readline()
M=int(M)
N=int(N)
ciclo=int(ciclo)
print ('Número de máquinas:',M)
print ('Número de trabajos:',N)
print ('Tiempo de ciclo:',ciclo)
Tiempos=[]
T={}
maq=1
trab=1
P=0
```

```
while True:
    Tiempos=f.readline()
    Tiempos=Tiempos.split()
    for i in Tiempos:
        T[maq,trab]=int(i)
        P=P+T[maq,trab]
        trab+=1
    maq=maq+1
    trab=1
    if len(Tiempos) == 0:
        break
f.close()

#Número de bines

bines=int(P/ciclo)+1
print("Número de bines:",bines)



try:
    # Create a new model
    WST=Model("WST")

    # Create variables
    X={}
    Y={}
    Z={}
    E={}
    Bin={}

    for r in range (1,M+1):
        for j in range (1,N+1):
```

```
            X[r,j]=WST.addVar(vtype=GRB.INTEGER, name='X[%d,%d]'%(r,j))

for r in range (1,M):
    for j in range (1,N+1):
        Y[r,j]=WST.addVar(vtype=GRB.INTEGER, name='Y[%d,%d]'%(r,j))

for i in range (1,N+1):
    for j in range (1,N+1):
        Z[i,j]=WST.addVar(vtype=GRB.BINARY, name='Z[%d,%d]'%(i,j))

for r in range (1,M+1):
    for j in range (1,N+1):
        for b in range (1,bines+1):
            Bin[r,j,b]=WST.addVar(vtype=GRB.BINARY, name='Bin[%d,%d
                ,%d]'%(r,j,b))

for r in range (0,M+1):
    for j in range (0,N+1):
        E[r,j]=WST.addVar(vtype=GRB.INTEGER, name='E[%d,%d]'%(r,j))


# Set objective
sum_Tmj=0
for j in range (1,N+1):
    sum_Tmj=sum_Tmj+T[M,j]
sum_Xmp=0
for p in range (1,N+1):
    sum_Xmp=sum_Xmp+X[M,p]
WST.setObjective(sum_Tmj+sum_Xmp, GRB.MINIMIZE)


# Add constraints
```

```
# C2
for i in range (1,N+1):
    sum_Zij=0
    for j in range (1,N+1):
        sum_Zij=sum_Zij+Z[i,j]
    WST.addConstr(sum_Zij==1,'C2(%d)\n'%(i))

# C3
for j in range (1,N+1):
    sum_Zij=0
    for i in range (1,N+1):
        sum_Zij=sum_Zij+Z[i,j]
    WST.addConstr(sum_Zij==1,'C3(%d)\n'%(j))

# C4
for r in range (1,M):
    for j in range (1,N):
        sum_Tri_Zijmas1=0
        sum_Trmas1i_Zij=0
        for i in range (1,N+1):
            sum_Tri_Zijmas1=sum_Tri_Zijmas1+T[r,i]*Z[i,j+1]
            sum_Trmas1i_Zij=sum_Trmas1i_Zij+T[r+1,i]*Z[i,j]
        WST.addConstr(sum_Tri_Zijmas1+X[r,j+1]+Y[r,j+1]-
            sum_Trmas1i_Zij-X[r+1,j+1]-Y[r,j]==0, 'C4(%d,%d)\n'%(r,j
            ))

# C5
for r in range (1,M):
    sum_Tri_Zi1=0
    for i in range (1,N+1):
        sum_Tri_Zi1=sum_Tri_Zi1+T[r,i]*Z[i,1]
```

```
        WST.addConstr(X[r+1,1]-X[r,1]-Y[r,1]-sum_Tri_Zi1==0,'C5(%d)\n
            '%(r))


# Sacando Erj
for r in range (1,M+1):
    sum_Erj=0
    for j in range (1,N+1):
        sum_Erj=sum_Erj+X[r,j]
        for i in range (1,N+1):
            sum_Erj=sum_Erj+T[r,i]*Z[i,j]
        WST.addConstr(E[r,j]-sum_Erj==0,'SacandoErj(%d,%d)\n'%(r,j)
            )


# Bin1
for r in range (1,M+1):
    for j in range (1,N+1):
        for b in range (1,bines+1):
            WST.addConstr(E[r,j]+P*Bin[r,j,b]<=P+b*ciclo,'Bin1(%d,%
                d,%d)\n'%(r,j,b))

# Bin2
for r in range (1,M+1):
    for j in range (1,N+1):
        Tri_Zij=0
        for i in range (1,N+1):
            Tri_Zij=Tri_Zij+T[r,i]*Z[i,j]
        for b in range (1,bines+1):
            WST.addConstr(E[r,j]-Tri_Zij-P*Bin[r,j,b]>=(b-1)*ciclo-
                P,'Bin2(%d,%d,%d)\n'%(r,j,b))
```

```
# Bin3
for r in range (1,M+1):
    for j in range (1,N+1):
        sum_Bin=0
        for b in range (1,bines+1):
            sum_Bin=sum_Bin+Bin[r,j,b]
        WST.addConstr(sum_Bin==1,'Bin3(%d,%d)\n'%(r,j))
print ("\n")

WST.params.timelimit=900

# Pasamos solución inicial [1,...,N]
for i in range (1,N+1):
    for j in range (1,N+1):
        if i==j:
            Z[i,j].start=1
        else:
            Z[i,j].start=0
WST.update()

for r in range (0,M+1):
    E[r,0].start=0

for j in range (0,N+1):
    E[0,j].start=0
WST.update()

nbins=[]
for r in range (1,M+1):
    nbins.append(1)
WST.update()
for j in range (1,N+1):
```

```
    for r in range (1,M+1):
        if (max(E[r-1,j].start,E[r,j-1].start)+T[r,j])>(nbins[r-1]*
            ciclo):
            E[r,j].start=nbins[r-1]*ciclo+T[r,j]
            nbins[r-1]+=1
            for l in range (r,M):
                if nbins[l]<nbins[r-1]:
                    nbins[l]+=1
        else:
            E[r,j].start=max(E[r,j-1].start,E[r-1,j].start)+T[r,j]
        WST.update()

WST.update()
for r in range (1,M+1):
    for j in range (1,N+1):
        if j==1:
            X[r,j].start=E[r,j].start-T[r,j]
        else:
            X[r,j].start=E[r,j].start-T[r,j]-E[r,j-1].start
        WST.update()

for r in range (1,M):
    for j in range (1,N+1):
        Y[r,j]=E[r+1,j]-T[r+1,j]-E[r,j]
        WST.update()

for r in range (1,M+1):
    for j in range (1,N+1):
        for b in range (1,bines+1):
            if E[r,j].start>(b-1)*ciclo and E[r,j].start<=b*ciclo:
                Bin[r,j,b].start=1
            else:
```

```
                    Bin[r,j,b].start=0
                WST.update()

    print("\nCmax.start",E[M,N].start,"\n")

    WST.optimize()
    tiempo_WST = WST.Runtime
    if WST.status == GRB.Status.TIME_LIMIT and WST.Objval==1e+100:
        f=open("Salida.txt",'a')
        f.write("WST ")
        f.write("      ")
        f.write("M "+str(M))
        f.write("   ")
        f.write("N "+str(N))
        f.write("   ")
        f.write("T "+str(ciclo))
        f.write("   ")
        f.write("Cmax *")
        f.write("         ")
        f.write("Runtime "+str(tiempo_WST))
        f.write("\n")
        f.close()

#    for v in WST.getVars():
#        print(v.varName, v.x)
    print('\n\nModelo WST')
    print('Función objetivo:', WST.objVal)
    print ("Tiempo en optimizar:", tiempo_WST)
#    WST.write("WST.lp")
#    WST.write("WST.sol")
except GurobiError:
    print('Error reported')
```

```
sec_WST=[]

for j in range(1,N+1):
    for i in range (1,N+1):
        if Z[i,j].X==1:
            sec_WST.append(i)

print ("Secuencia: ",sec_WST)

f=open("Salida.txt",'a')
f.write("WST ")
f.write("      ")
f.write("M "+str(M))
f.write("   ")
f.write("N "+str(N))
f.write("   ")
f.write("T "+str(ciclo))
f.write("   ")
f.write("Cmax "+str(WST.objVal))
f.write("   ")
f.write("Runtime "+str(tiempo_WST))
f.write("\n")
f.close()
```

2. Modelo Wilson

```
from gurobipy import *
import sys

f=open(sys.argv[1],'r')
M=f.readline()
N=f.readline()
ciclo=f.readline()
N=int(N)
M=int(M)
ciclo=int(ciclo)
print ('Número de máquinas:',M)
print ('Número de trabajos:',N)
print ('Tiempo de ciclo:',ciclo)
Tiempos=[]
T={}
maq=1
trab=1
P=0
while True:
    Tiempos=f.readline()
    Tiempos=Tiempos.split()
    for i in Tiempos:
        T[maq,trab]=int(i)
        P=P+T[maq,trab]
        trab+=1
    maq=maq+1
    trab=1
    if len(Tiempos) == 0:
        break
f.close()
```

```
#Número de bines

bines=int(P/ciclo)+1
print("Número de bines:",bines)



try:
    # Create a new model
    Wilson=Model("Wilson")

    # Create variables
    B={}
    Z={}
    Bin={}

    for r in range (0,M+1):
        for j in range (0,N+1):
            B[r,j]=Wilson.addVar(vtype=GRB.INTEGER, name='B[%d,%d]'%(r,
                j))

    for i in range (1,N+1):
        for j in range (1,N+1):
            Z[i,j]=Wilson.addVar(vtype=GRB.BINARY, name='Z[%d,%d]'%(i,j
                ))

    for r in range (1,M+1):
        for j in range (1,N+1):
            for b in range (1,bines+1):
                Bin[r,j,b]=Wilson.addVar(vtype=GRB.BINARY, name='Bin[%d
                    ,%d,%d]'%(r,j,b))
```

```
# Set objective
sum_Tmi_Zin=0
for i in range (1,N+1):
    sum_Tmi_Zin=sum_Tmi_Zin+T[M,i]*Z[i,N]
Wilson.setObjective(B[M,N]+sum_Tmi_Zin, GRB.MINIMIZE)


# Add constraints

# C2
for i in range (1,N+1):
    sum_Zij=0
    for j in range (1,N+1):
        sum_Zij=sum_Zij+Z[i,j]
    Wilson.addConstr(sum_Zij==1,'C2(%d)\n'%(i))

# C3
for j in range (1,N+1):
    sum_Zij=0
    for i in range (1,N+1):
        sum_Zij=sum_Zij+Z[i,j]
    Wilson.addConstr(sum_Zij==1,'C3(%d)\n'%(j))

# C7
for j in range (1,N):
    sum_T1i_Zij=0
    for i in range (1,N+1):
        sum_T1i_Zij=sum_T1i_Zij+T[1,i]*Z[i,j]
    Wilson.addConstr(B[1,j]+sum_T1i_Zij-B[1,j+1]<=0, 'C7(%d)\n'%(j)
        )
```

```
# C8
Wilson.addConstr(B[1,1]==0, 'C8')

# C9
for r in range (1,M):
    sum_Tri_Zi1=0
    for i in range (1,N+1):
        sum_Tri_Zi1=sum_Tri_Zi1+T[r,i]*Z[i,1]
    Wilson.addConstr(B[r,1]+sum_Tri_Zi1-B[r+1,1]<=0, 'C9(%d)\n'%(r)
        )

# C10
for r in range (1,M):
    for j in range (2,N+1):
        sum_Tri_Zij=0
        for i in range (1,N+1):
            sum_Tri_Zij=sum_Tri_Zij+T[r,i]*Z[i,j]
        Wilson.addConstr(B[r,j]+sum_Tri_Zij-B[r+1,j]<=0, 'C10(%d,%d
            )\n'%(r,j))

# C11
for r in range (2,M+1):
    for j in range (1,N):
        sum_Tri_Zij=0
        for i in range (1,N+1):
            sum_Tri_Zij=sum_Tri_Zij+T[r,i]*Z[i,j]
        Wilson.addConstr(B[r,j]+sum_Tri_Zij-B[r,j+1]<=0, 'C11(%d,%d
            )\n'%(r,j))

# Bin1
for r in range (1,M+1):
    for j in range (1,N+1):
```

```
        Tri_Zij=0
        for i in range (1,N+1):
            Tri_Zij=Tri_Zij+T[r,i]*Z[i,j]
        for b in range (1,bines+1):
            Wilson.addConstr(B[r,j]+Tri_Zij+P*Bin[r,j,b]<=P+b*ciclo
                ,'Bin1(%d,%d,%d,%d)\n'%(r,j,i,b))

# Bin2
for r in range (1,M+1):
    for j in range (1,N+1):
        for b in range (1,bines+1):
            Wilson.addConstr(B[r,j]-P*Bin[r,j,b]>=(b-1)*ciclo-P,'
                Bin2(%d,%d,%d)\n'%(r,j,b))

# Bin3
for r in range (1,M+1):
    for j in range (1,N+1):
        sum_Bin=0
        for b in range (1,bines+1):
            sum_Bin=sum_Bin+Bin[r,j,b]
        Wilson.addConstr(sum_Bin==1,'Bin3(%d,%d)\n'%(r,j))
print("\n")

Wilson.params.timelimit=900

# Pasamos la solución inicial [1,...,N]
for i in range (1,N+1):
    for j in range (1,N+1):
        if i==j:
            Z[i,j].start=1
        else:
            Z[i,j].start=0
```

```
Wilson.update()

for r in range (0,M+1):
    B[r,0].start=0
    T[r,0]=0
Wilson.update()

for j in range (0,N+1):
    B[0,j].start=0
    T[0,j]=0
Wilson.update()

nbins=[]
for r in range (1,M+1):
    nbins.append(1)

B[1,1].start=0
Wilson.update()
for j in range (1,N+1):
    for r in range (1,M+1):
        aux1=max(B[r,j-1].start+T[r,j-1],B[r-1,j].start+T[r-1,j])+T
            [r,j]
        aux2=nbins[r-1]*ciclo
        if aux1>aux2:
            B[r,j].start=nbins[r-1]*ciclo
            nbins[r-1]+=1
            for l in range (r,M):
                if nbins[l]<nbins[r-1]:
                    nbins[l]+=1
        else:
            B[r,j].start=max(B[r,j-1].start+T[r,j-1],B[r-1,j].start
```

```
                    +T[r-1,j])
            Wilson.update()

    for r in range (1,M+1):
        for j in range (1,N+1):
            for b in range (1,bines+1):
                if B[r,j].start>=(b-1)*ciclo and B[r,j].start<b*ciclo:
                    Bin[r,j,b].start=1
                else:
                    Bin[r,j,b].start=0
                Wilson.update()

    Wilson.optimize()
    tiempo_Wilson = Wilson.Runtime
    if Wilson.status == GRB.Status.INFEASIBLE:
        f=open("Salida.txt",'a')
        f.write("Wilson ")
        f.write("  ")
        f.write("M "+str(M))
        f.write("  ")
        f.write("N "+str(N))
        f.write("  ")
        f.write("T "+str(ciclo))
        f.write("  ")
        f.write("Cmax ")
        f.write("-1")
        f.write("   ")
        f.write("Solución infactible\n")
        f.close()

#    for v in Wilson.getVars():
#        print(v.varName, v.x)
```

```
    print("\n\nModelo Wilson")
    print('Función objetivo:', Wilson.objVal)
    print ("Tiempo en optimizar:", tiempo_Wilson)
#     Wilson.write("Wilson.lp")
#     Wilson.write("Wilson.sol")
except GurobiError:
    print('Error reported')

sec_Wilson=[]
for j in range(1,N+1):
    for i in range (1,N+1):
        if Z[i,j].X ==1:
            sec_Wilson.append(i)
print ("Secuencia: ",sec_Wilson)

f=open("Salida.txt",'a')
f.write("Wilson ")
f.write("  ")
f.write("M "+str(M))
f.write("  ")
f.write("N "+str(N))
f.write("  ")
f.write("T "+str(ciclo))
f.write("  ")
f.write("Cmax "+str(Wilson.objVal))
f.write("  ")
f.write("Runtime "+str(tiempo_Wilson))
f.write("\n")
f.close()
```

3. Modelo TS2

```
from gurobipy import *
import sys

f=open(sys.argv[1],'r')
M=f.readline()
N=f.readline()
ciclo=f.readline()
N=int(N)
M=int(M)
ciclo=int(ciclo)
print ('Número de máquinas:',M)
print ('Número de trabajos:',N)
print ('Tiempo de ciclo:',ciclo)
Tiempos=[]
T={}
maq=1
trab=1
P=0
while True:
    Tiempos=f.readline()
    Tiempos=Tiempos.split()
    for i in Tiempos:
        T[maq,trab]=int(i)
        P=P+T[maq,trab]
        trab+=1
    maq=maq+1
    trab=1
    if len(Tiempos) == 0:
        break
f.close()
```

```
#Número de bines

bines=int(P/ciclo)+1
print("Número de bines:",bines)


try:
    # Create a new model
    TS2=Model("TS2")

    # Create variables
    E={}
    Z={}
    Bin={}

    for r in range (0,M+1):
        for j in range (0,N+1):
            E[r,j]=TS2.addVar(vtype=GRB.INTEGER, name='E[%d,%d]'%(r,j))

    for i in range (1,N+1):
        for j in range (1,N+1):
            Z[i,j]=TS2.addVar(vtype=GRB.BINARY, name='Z[%d,%d]'%(i,j))

    for r in range (1,M+1):
        for j in range (1,N+1):
            for b in range (1,bines+1):
                Bin[r,j,b]=TS2.addVar(vtype=GRB.BINARY, name='Bin[%d,%d
                    ,%d]'%(r,j,b))


                # Set objective
```

```
TS2.setObjective(E[M,N], GRB.MINIMIZE)


# Add constraints

# C2
for i in range (1,N+1):
    sum_Zij=0
    for j in range (1,N+1):
        sum_Zij=sum_Zij+Z[i,j]
    TS2.addConstr(sum_Zij==1,'C2(%d)\n'%(i))

# C3
for j in range (1,N+1):
    sum_Zij=0
    for i in range (1,N+1):
        sum_Zij=sum_Zij+Z[i,j]
    TS2.addConstr(sum_Zij==1,'C3(%d)\n'%(j))
# C12
for r in range (1,M+1):
    for j in range (1,N):
        sum_Tri_Zijmas1=0
        for i in range (1,N+1):
            sum_Tri_Zijmas1=sum_Tri_Zijmas1+T[r,i]*Z[i,j+1]
        TS2.addConstr(E[r,j]+sum_Tri_Zijmas1-E[r,j+1]<=0, 'C12(%d,%
            d)\n'%(r,j))

# C13
for r in range (1,M):
    for j in range (1,N+1):
        sum_Trmas1i_Zij=0
```

```
        for i in range (1,N+1):
            sum_Trmas1i_Zij=sum_Trmas1i_Zij+T[r+1,i]*Z[i,j]
        TS2.addConstr(E[r,j]+sum_Trmas1i_Zij-E[r+1,j]<=0, 'C13(%d,%
            d)\n'%(r,j))

# C14
sum_T1i_Zi1=0
for i in range (1,N+1):
    sum_T1i_Zi1=sum_T1i_Zi1+T[1,i]*Z[i,1]
TS2.addConstr(E[1,1]-sum_T1i_Zi1>=0,'C14')

# Bin1
for r in range (1,M+1):
    for j in range (1,N+1):
        for b in range (1,bines+1):
            TS2.addConstr(E[r,j]+P*Bin[r,j,b]<=P+b*ciclo,'Bin1(%d,%
                d,%d)\n'%(r,j,b))

# Bin2
for r in range (1,M+1):
    for j in range (1,N+1):
        Tri_Zij=0
        for i in range (1,N+1):
            Tri_Zij=Tri_Zij+T[r,i]*Z[i,j]
        for b in range (1,bines+1):
            TS2.addConstr(E[r,j]-Tri_Zij-P*Bin[r,j,b]>=(b-1)*ciclo-
                P,'Bin2(%d,%d,%d)\n'%(r,j,b))

# Bin3
for r in range (1,M+1):
    for j in range (1,N+1):
        sum_Bin=0
```

```
            for b in range (1,bines+1):
                sum_Bin=sum_Bin+Bin[r,j,b]
            TS2.addConstr(sum_Bin==1,'Bin3(%d,%d)\n'%(r,j))
print("\n")

TS2.params.timelimit=900

# Pasamos solución inicial [1,...,N]
for i in range (1,N+1):
    for j in range (1,N+1):
        if i==j:
            Z[i,j].start=1
        else:
            Z[i,j].start=0
TS2.update()

for r in range (0,M+1):
    E[r,0].start=0
    TS2.update()

for j in range (0,N+1):
    E[0,j].start=0
    TS2.update()

nbins=[]
for r in range (1,M+1):
    nbins.append(1)

TS2.update()
for j in range (1,N+1):
    for r in range (1,M+1):
        if (max(E[r-1,j].start,E[r,j-1].start)+T[r,j])>(nbins[r-1]*
```

```
            ciclo):
             E[r,j].start=nbins[r-1]*ciclo+T[r,j]
             nbins[r-1]+=1
             for l in range (r,M):
                  if nbins[l]<nbins[r-1]:
                       nbins[l]+=1
        else:
             E[r,j].start=max(E[r,j-1].start,E[r-1,j].start)+T[r,j]
        TS2.update()

TS2.update()

for r in range (1,M+1):
    for j in range (1,N+1):
        for b in range (1,bines+1):
             if E[r,j].start>(b-1)*ciclo and E[r,j].start<=b*ciclo:
                  Bin[r,j,b].start=1
             else:
                  Bin[r,j,b].start=0
             TS2.update()

print("\nCmax.start",E[M,N].start,"\n")
TS2.optimize()
tiempo_TS2 = TS2.Runtime

             if TS2.status == GRB.Status.INFEASIBLE:
    f=open("Salida.txt",'a')
    f.write("TS2 ")
    f.write("  ")
    f.write("M "+str(M))
    f.write("  ")
    f.write("N "+str(N))
```

```
        f.write("   ")
        f.write("T "+str(ciclo))
        f.write("   ")
        f.write("Cmax ")
        f.write("-1")
        f.write("    ")
        f.write("Solución infactible\n")
        f.close()

#    for v in TS2.getVars():
#        print(v.varName, v.x)
    print ("\n\nModelo TS2")
    print('Función objetivo:', TS2.objVal)
    print ("Tiempo en optimizar:", tiempo_TS2)
#    TS2.write("TS2.lp")
#    TS2.write("TS2.sol")
except GurobiError:
    print('Error reported')

sec_TS2=[]

for j in range(1,N+1):
    for i in range (1,N+1):
        if Z[i,j].X==1:
            sec_TS2.append(i)

print ("Secuencia:",sec_TS2)

f=open("Salida.txt",'a')
f.write("TS2 ")
f.write("      ")
f.write("M "+str(M))
```

```
f.write("    ")
f.write("N "+str(N))
f.write("    ")
f.write("T "+str(ciclo))
f.write("    ")
f.write("Cmax "+str(TS2.objVal))
f.write("    ")
f.write("Runtime "+str(tiempo_TS2))
f.write("\n")
f.close()
```

4. Modelo SGST

```
from gurobipy import *
import sys

f=open(sys.argv[1],'r')
M=f.readline()
N=f.readline()
ciclo=f.readline()
N=int(N)
M=int(M)
ciclo=int(ciclo)
print ('Número de máquinas:',M)
print ('Número de trabajos:',N)
print ('Tiempo de ciclo:',ciclo)
Tiempos=[]
T={}
maq=1
trab=1
P=0
while True:
    Tiempos=f.readline()
    Tiempos=Tiempos.split()
    for i in Tiempos:
        T[maq,trab]=int(i)
        P=P+T[maq,trab]
        trab+=1
    maq=maq+1
    trab=1
    if len(Tiempos) == 0:
        break
f.close()
```

```
#Número de bines
bines=int(P/ciclo)+1
print("Número de bines:",bines)
try:
    # Create a new model
    SGST=Model("SGST")

    # Create variables
    C={}
    D={}
    Bin={}

    for r in range (0,M+1):
        for i in range (0,N+1):
            C[r,i]=SGST.addVar(vtype=GRB.INTEGER, name='C[%d,%d]'%(r,i)
                )

    for i in range (1,N+1):
        for k in range (i+1,N+1):
            D[i,k]=SGST.addVar(vtype=GRB.BINARY, name='D[%d,%d]'%(i,k))

    for r in range (1,M+1):
        for i in range (1,N+1):
            for b in range (1,bines+1):
                Bin[r,i,b]=SGST.addVar(vtype=GRB.BINARY, name='Bin[%d,%
                    d,%d]'%(r,i,b))

    Cmax=SGST.addVar(vtype=GRB.INTEGER, name='Cmax')


    # Set objective
```

```
SGST.setObjective(Cmax, GRB.MINIMIZE)


# Add constraints

# C15
for i in range (1,N+1):
    SGST.addConstr(C[1,i]>=T[1,i],'C15(%d)\n'%(i))

# C16
for r in range (1,M):
    for i in range (1,N+1):
        SGST.addConstr(C[r+1,i]-C[r,i]>=T[r+1,i], 'C16(%d,%d)\n'%(r
            ,i))

# C17
for r in range (1,M+1):
    for i in range (1,N+1):
        for k in range (i+1,N+1):
            SGST.addConstr(C[r,i]-C[r,k]+P*D[i,k]>=T[r,i],'C17(%d,%
                d,%d)\n'%(r,i,k))

# C18
for r in range (1,M+1):
    for i in range (1,N+1):
        for k in range (i+1,N+1):
            SGST.addConstr(C[r,k]-C[r,i]-P*D[i,k]>=T[r,k]-P,'C18(%d
                ,%d,%d)\n'%(r,i,k))

# C19
for i in range (1,N+1):
```

```
    SGST.addConstr(Cmax>=C[M,i],'C19(%d)\n'%(i))

# Bin1
for r in range (1,M+1):
    for i in range (1,N+1):
        for b in range (1,bines+1):
            SGST.addConstr(C[r,i]-b*ciclo+P*Bin[r,i,b]<=P,'Bin1(%d
                ,%d,%d)\n'%(r,i,b))
# Bin2
for r in range (1,M+1):
    for i in range (1,N+1):
        for b in range (1,bines+1):
            SGST.addConstr(C[r,i]-T[r,i]-P*Bin[r,i,b]>=(b-1)*ciclo-
                P,'Bin2(%d,%d,%d)\n'%(r,i,b))

# Bin3
for r in range (1,M+1):
    for j in range (1,N+1):
        sum_Bin=0
        for b in range (1,bines+1):
            sum_Bin=sum_Bin+Bin[r,j,b]
        SGST.addConstr(sum_Bin==1,'Bin3(%d,%d)\n'%(r,j))
print("\n")

SGST.params.timelimit=900

# Pasando solución inicial [1,...,N]
for i in range (1,N+1):
    for k in range (i+1,N+1):
        if i<k:
            D[i,k].start=1
SGST.update()
```

```
for r in range (0,M+1):
    C[r,0].start=0
SGST.update()

for i in range (0,N+1):
    C[0,i].start=0
SGST.update()

nbins=[]
for r in range (1,M+1):
    nbins.append(1)

SGST.update()
for i in range (1,N+1):
    for r in range (1,M+1):
        aux1=max(C[r-1,i].start,C[r,i-1].start)+T[r,i]
        aux2=nbins[r-1]*ciclo
        if aux1>aux2:
            C[r,i].start=nbins[r-1]*ciclo+T[r,i]
            nbins[r-1]+=1
            for l in range (r,M):
                if nbins[l]<nbins[r-1]:
                    nbins[l]+=1
        else:
            C[r,i].start=max(C[r,i-1].start,C[r-1,i].start)+T[r,i]
        SGST.update()

for r in range (1,M+1):
    for i in range (1,N+1):
        for b in range (1,bines+1):
            if C[r,i].start>(b-1)*ciclo and C[r,i].start<=b*ciclo:
```

```
                    Bin[r,i,b].start=1
                else:
                    Bin[r,i,b].start=0
                SGST.update()
    SGST.optimize()
    tiempo_SGST=SGST.Runtime

                if SGST.status == GRB.Status.INFEASIBLE:
        f=open("Salida.txt",'a')
        f.write("SGST ")
        f.write("  ")
        f.write("M "+str(M))
        f.write("  ")
        f.write("N "+str(N))
        f.write("  ")
        f.write("T "+str(ciclo))
        f.write("  ")
        f.write("Cmax ")
        f.write("-1")
        f.write("   ")
        f.write("Solución infactible\n")
        f.close()

#    for v in SGST.getVars():
#        print(v.varName, v.x)
    print('\n\nModelo SGST')
    print('Función Objetivo:', SGST.objVal)
    print ("Tiempo en optimizar:", tiempo_SGST)
#    SGST.write("SGST.lp")
#    SGST.write("SGST.sol")
except GurobiError:
    print('Error reported')
```

```
sec_SGST=[]
for i in range (1,N+1):
    sec_SGST.append(i)
for i in range (1,N+1):
    for k in range (i+1,N+1):
        if D[i,k].X==0:
            if sec_SGST.index(i)<sec_SGST.index(k):
                aux=sec_SGST.pop(sec_SGST.index(k))
                sec_SGST.insert(sec_SGST.index(i),aux)
print("Secuencia:",sec_SGST)

f=open("Salida.txt",'a')
f.write("SGST ")
f.write("      ")
f.write("M "+str(M))
f.write("   ")
f.write("N "+str(N))
f.write("   ")
f.write("T "+str(ciclo))
f.write("   ")
f.write("Cmax "+str(SGST.objVal))
f.write("   ")
f.write("Runtime "+str(tiempo_SGST))
f.write("\n")
f.close()
```

5. Modelo LYeq

```
from gurobipy import *
import sys

f=open(sys.argv[1],'r')
M=f.readline()
N=f.readline()
ciclo=f.readline()
N=int(N)
M=int(M)
ciclo=int(ciclo)
print ('Número de máquinas:',M)
print ('Número de trabajos:',N)
print ('Tiempo de ciclo:',ciclo)
Tiempos=[]
T={}
maq=1
trab=1
P=0
while True:
    Tiempos=f.readline()
    Tiempos=Tiempos.split()
    for i in Tiempos:
        T[maq,trab]=int(i)
        P=P+T[maq,trab]
        trab+=1
    maq=maq+1
    trab=1
    if len(Tiempos) == 0:
        break
f.close()
```

```
#Número de bines
bines=int(P/ciclo)+1
print("Número de bines:",bines)

try:
    # Create a new model
    LYeq=Model("LYeq")

    # Create variables
    C={}
    D={}
    Q={}
    Bin={}

    for r in range (0,M+1):
        for i in range (0,N+1):
            C[r,i]=LYeq.addVar(vtype=GRB.INTEGER, name='C[%d,%d]'%(r,i)
                )

    for i in range (1,N+1):
        for k in range (i+1,N+1):
            D[i,k]=LYeq.addVar(vtype=GRB.BINARY, name='D[%d,%d]'%(i,k))

    for r in range (1,M+1):
        for i in range (1,N+1):
            for k in range (i+1,N+1):
                Q[r,i,k]=LYeq.addVar(vtype=GRB.INTEGER, name='Q[%d,%d,%
                    d]'%(r,i,k))

    for r in range (1,M+1):
        for i in range (1,N+1):
```

```
        for b in range (1,bines+1):
            Bin[r,i,b]=LYeq.addVar(vtype=GRB.BINARY, name='Bin[%d,%
                d,%d]'%(r,i,b))

Cmax=LYeq.addVar(vtype=GRB.INTEGER, name='Cmax')


# Set objective

LYeq.setObjective(Cmax, GRB.MINIMIZE)


# Add constraints

# C15
for i in range (1,N+1):
    LYeq.addConstr(C[1,i]>=T[1,i],'C15(%d)\n'%(i))

# C16
for r in range (1,M):
    for i in range (1,N+1):
        LYeq.addConstr(C[r+1,i]-C[r,i]>=T[r+1,i], 'C16(%d,%d)\n'%(r
            ,i))

# C19
for i in range (1,N+1):
    LYeq.addConstr(Cmax>=C[M,i],'C19(%d)\n'%(i))

# C21
for r in range (1,M+1):
    for i in range (1,N+1):
        for k in range (i+1,N+1):
```

```
                LYeq.addConstr(P*D[i,k]+C[r,i]-C[r,k]-Q[r,i,k]==T[r,i
                    ],'C21(%d,%d,%d)\n'%(r,i,k))

# C22
for r in range (1,M+1):
    for i in range (1,N+1):
        for k in range (i+1,N+1):
            LYeq.addConstr(Q[r,i,k]<=P-T[r,i]-T[r,k],'C22(%d,%d,%d)
                \n'%(r,i,k))

# Bin1
for r in range (1,M+1):
    for i in range (1,N+1):
        for b in range (1,bines+1):
            LYeq.addConstr(C[r,i]-b*ciclo+P*Bin[r,i,b]<=P,'Bin1(%d
                ,%d,%d)\n'%(r,i,b))

# Bin2
for r in range (1,M+1):
    for i in range (1,N+1):
        for b in range (1,bines+1):
            LYeq.addConstr(C[r,i]-T[r,i]-P*Bin[r,i,b]>=(b-1)*ciclo-
                P,'Bin2(%d,%d,%d)\n'%(r,i,b))

# Bin3
for r in range (1,M+1):
    for j in range (1,N+1):
        sum_Bin=0
        for b in range (1,bines+1):
            sum_Bin=sum_Bin+Bin[r,j,b]
        LYeq.addConstr(sum_Bin==1,'Bin3(%d,%d)\n'%(r,j))
print("\n")
```

```
LYeq.params.timelimit=900

# Pasando solución inicial [1,...,N]
for i in range (1,N+1):
    for k in range (i+1,N+1):
        if i<k:
            D[i,k].start=1
LYeq.update()

for r in range (0,M+1):
    C[r,0].start=0
LYeq.update()

for i in range (0,N+1):
    C[0,i].start=0
LYeq.update()

nbins=[]
for r in range (1,M+1):
    nbins.append(1)

LYeq.update()
for i in range (1,N+1):
    for r in range (1,M+1):
        aux1=max(C[r-1,i].start,C[r,i-1].start)+T[r,i]
        aux2=nbins[r-1]*ciclo
        if aux1>aux2:
            C[r,i].start=nbins[r-1]*ciclo+T[r,i]
            nbins[r-1]+=1
            for l in range (r,M):
                if nbins[l]<nbins[r-1]:
```

```
                    nbins[l]+=1
        else:
            C[r,i].start=max(C[r,i-1].start,C[r-1,i].start)+T[r,i]
        LYeq.update()

for r in range (1,M+1):
    for i in range (1,N+1):
        for b in range (1,bines+1):
            if C[r,i].start>(b-1)*ciclo and C[r,i].start<=b*ciclo:
                Bin[r,i,b].start=1
            else:
                Bin[r,i,b].start=0
            LYeq.update()

for r in range (1,M+1):
    for i in range (1,N+1):
        for k in range (i+1,N+1):
            Q[r,i,k].start=P*D[i,k].start+C[r,i].start-C[r,k].start
                -T[r,i]
            LYeq.update()
LYeq.optimize()
tiempo_LYeq = LYeq.Runtime

            if LYeq.status == GRB.Status.INFEASIBLE:
    f=open("Salida.txt",'a')
    f.write("LYeq ")
    f.write("  ")
    f.write("M "+str(M))
    f.write("  ")
    f.write("N "+str(N))
    f.write("  ")
    f.write("T "+str(ciclo))
```

```
            f.write("   ")
            f.write("Cmax ")
            f.write("-1")
            f.write("    ")
            f.write("Solución infactible\n")
            f.close()

#     for v in LYeq.getVars():
#         print(v.varName, v.x)
    print('\n\nModelo LYeq')
    print('Función Objetivo:', LYeq.objVal)
    print ("Tiempo en optimizar:", tiempo_LYeq)
#     LYeq.write("LYeq.lp")
#     LYeq.write("LYeq.sol")
except GurobiError:
    print('Error reported')

sec_LYeq=[]
for i in range (1,N+1):
    sec_LYeq.append(i)
for i in range (1,N+1):
    for k in range (i+1,N+1):
        if D[i,k].X==0:
            if sec_LYeq.index(i)<sec_LYeq.index(k):
                aux=sec_LYeq.pop(sec_LYeq.index(k))
                sec_LYeq.insert(sec_LYeq.index(i),aux)

print("Secuencia:",sec_LYeq)

f=open("Salida.txt",'a')
f.write("LYeq ")
f.write("     ")
```

```
f.write("M "+str(M))
f.write("    ")
f.write("N "+str(N))
f.write("    ")
f.write("T "+str(ciclo))
f.write("    ")
f.write("Cmax "+str(LYeq.objVal))
f.write("    ")
f.write("Runtime "+str(tiempo_LYeq))
f.write("\n")
f.close()
```