Trabajo Fin de Grado
Grado en Ingeniería de Organización Industrial

Taller de flujo regular con recursos humanos y minimización del makespan: Modelos de programación lineal entera.

Autor: David Gómez Medina

Tutor: Víctor Fernández-Viagas Escudero

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

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de Organización Industrial

Taller de flujo regular con recursos humanos y minimización del makespan: Modelos de programación lineal entera.

Autor:

David Gómez Medina

Tutor:

Víctor Fernández-Viagas Escudero

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

Sevilla, 2020

Trabajo Fin de Grado: Taller de flujo regular con recursos humanos y minimización del makespan: Modelos de programación lineal entera.

Autor: David Gómez Medina

Tutor: Víctor Fernández-Viagas Escudero

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

Capítulo 1 Introducción

Desde la Primera Revolución Industrial, el sector industrial ha ido sufriendo una serie de cambios debido al descubrimiento y desarrollo tecnológico que han ido enfocados a que, empleando los mismos recursos, se pudiera realizar una mayor producción con menores costes, en menos tiempo y sin perder calidad en el producto. Actualmente, uno de los objetivos más importantes en toda industria es la búsqueda de la eficiencia. Para esta búsqueda de la eficiencia, las empresas han dejado de centrarse solamente en la tecnología que empleaban, comenzándose a preocupar por cómo organizaban el trabajo ya que se dieron cuenta de que, pequeños cambios en la cadena productiva podían afectar en gran medida en los recursos que empleaban. De esta necesidad surge la organización de la producción.

Framinan *et al.* (2014), definen la organización de la producción como "la coordinación y control necesario para obtener los productos al menor coste posible y en el menor plazo posible. Para ello será necesario un conjunto de decisiones operacionales que gestionen todos los recursos de los que dispone la empresa (máquinas, operarios, materias primas, etc.)". El objetivo es que las empresas sean capaces de tomar las decisiones que proporcionen resultados óptimos para su entorno productivo. De manera más específica, las empresas tienen que tomar de manera bastante frecuente (diaria o semanalmente) decisiones que competen específicamente a la cadena productiva de la empresa, planteándose preguntas como ¿qué producto fabrico primero?, ¿cuándo voy a tener estos productos listos?, etc.

La programación y planificación se convierte entonces en elementos de gran importancia dentro de la empresa. Por ello, surge el concepto de programación de la producción que se puede definir como la "Asignación de los recursos de la empresa a las diferentes tareas para un periodo de tiempo determinado con la intención de optimizar uno o más objetivos (Pinedo, 2016)". El resultado de esta planificación de la producción es una "programación donde se especifica qué trabajo debe entrar en qué recurso y cuándo, (Framinan *et al*, 2014)". En la programación de la producción se define trabajo como "los productos a fabricar o las unidades de trabajo en las que se puede dividir la actividad de fabricación, (Framinan *et al*, 2014)" y máquinas como "las operaciones / tareas requeridas por los trabajos son proporcionadas por diferentes recursos productivos disponibles en taller.

Estos pueden ir desde herramientas y accesorios baratos hasta recursos humanos y maquinaria, (Framinan *et al*, 2014)".

Durante las últimas décadas, la programación de la producción ha ido cobrando cada vez una mayor importancia al ponerle solución a un problema muy presente en el sector industrial que consiste en la asignación de los recursos disponibles de manera eficiente para la producción.

Este proyecto tiene como objetivo el diseño de modelos de programación de la producción que estudien un entorno productivo con el objetivo de minimizar el tiempo en el que se terminan de procesar todos los trabajos previstos. El entorno productivo del que parte este proyecto es el *flowshop,* un entorno bastante conocido en el que los trabajos pasan por las máquinas, una tras otra, siguiendo la misma ruta y con permutación, es decir, los trabajos se procesan en las máquinas siguiendo siempre la misma secuencia de trabajos. Adicionalmente, se considerará la presencia de recursos humanos que intervienen en el proceso productivo.

La programación de la producción para el entorno *flowshop* ha sido ya profundamente estudiada lo que permitirá tomar como base modelos ya existentes para adaptarlos al entorno productivo de este proyecto ya que éste apenas ha sido estudiado en la literatura.

1.1 Objetivos

Como objetivo principal de este proyecto es establecer diferentes filosofías en varios modelos que resuelvan el problema presentado al inicio de este capítulo. Para ello se partirá de diversos modelos ya existentes en la literatura para adaptarlos y analizar la eficiencia en cuanto al tiempo que tardan en resolver el problema.

Como objetivos específicos se encuentran los siguientes:

- Diseño de diferentes modelos de programación línea entera para la resolución de manera exacta del problema de estudio.
- Programación de los modelos adaptados a fin de una posterior resolución de diferentes instancias en el programa Gurobi.
- Validación de los modelos resolviendo pequeños problemas.

- Resolución de diversas instancias de diferentes dimensiones en Gurobi para su posterior análisis.
- Determinar la eficiencia de los modelos adaptados para determinar, mediante evidencias, el comportamiento de cada uno de ellos.

1.2 Sumario

Este documento se estructura en un total de cinco capítulos y un anexo con esta composición:

Capítulo 1: Incluye la presentación de este proyecto con los objetivos y la composición del documento.

Capítulo 2: Incluye una introducción del marco teórico y conceptual sobre el que se desarrolla el proyecto, así como la definición de la notación que se empleará y la descripción profunda del problema particular del proyecto.

Capítulo 3: Se presentan los cuatro modelos de la literaturas explicando cada uno de ellos, así como la adaptación para el problema particular de este proyecto. Además, para cada modelo adaptado, se presenta una resolución de un problema de pequeñas dimensiones.

Capítulo 4: Incluye el análisis computacional realizados a los diferentes modelos atendiendo a tres indicadores diferentes.

Capítulo 5: Incluye las conclusiones posteriores al análisis sobre los resultados obtenidos.

Anexo: Incluye las líneas de código de programación en C# de los modelos adaptados desarrollados en el capítulo 3.

Capítulo 2 Descripción del problema

A lo largo del capítulo 2 se expondrá, en primer lugar, la notación que se empleará a lo largo del TFG (Trabajo de Fin de Grado) mientras que en segundo lugar se realizará la descripción del problema particular que se estudiará.

2.1 Notación genérica de la descripción

Como se puede prever, a lo largo de la historia, las diferentes industrias han ido evolucionando hacia nuevos entornos productivos con el fin de optimizar su producción. Por ello, cada industria plantea un sistema productivo diferente adaptándose tanto a sus necesidades como a los objetivos que la dirección desea alcanzar.

Por todo esto, para definir el problema a tratar en este TFG, se empleará el formato que plantea Pinedo en su libro "Scheduling" (2016). En este libro, se define que un problema de programación se puede describir en base a tres elementos α | β | γ cuyos significados se detallan a continuación:

α: "Describe el entorno de las máquinas". Alpha hace referencia al entorno productivo, número de máquinas y cómo han de circular los diferentes trabajos por ellas. Los entornos más conocidos, definidos por Pinedo (2016) son:

- **(1)** ***Single Machine****:* Una única máquina por la que han de pasar uno o más trabajos.
- **(Pm)** ***Parallel machines****:* Hay *m* máquinas idénticas paralelas.
- **(Qm)** ***Uniform parallel machines****:* Un número m de máquinas que realizan la misma operación, pero con tiempos de procesos proporcionales según una constante de velocidad.
- **(Rm)** ***Unrelated parallel machines****:* Un número *m* de máquinas paralelas con tiempos de procesos sin relación entre ellas.
- **(Fm)** ***Flow shop****:* Hay *m* máquinas en serie en las que cada trabajo tiene que ser procesado una vez en cada máquina. Por ejemplo: F2 hace referencia a un *Flow shop*

de dos máquinas por las que tienen que pasar *n* trabajos siguiendo la misma ruta definida, máquina 1 y después máquina 2.

- **(Jm) *Job shop***: Hay *m* máquinas por las que deberán pasar diferentes trabajos, cada uno, con una ruta predeterminada.
- **(Om) *Open shop***: Un número *m* de máquinas por las que deben pasar todos los diferentes trabajos sin una ruta determinada en el que se hagan las operaciones sobre ellos.

β: Hace referencia a las " restricciones del sistema". En el apartado de restricciones se pueden incluir desde restricciones en cuanto a la fecha en la que el trabajo puede empezar a ser procesado hasta la obligación de que, para cada trabajo, la máquina deba incluir un tiempo de preparación o *setup.* Las más comunes comentadas por Pinedo (2016) son :

- **(r_j) *Release dates***: El trabajo *j* no se puede procesar antes de instante r_j.
- **(prmp) *Preemptions***: Permite la interrupción de la operación de un trabajo en una máquina para insertar un trabajo diferente en dicha máquina.
- **(prec) *Precedence constrains***: Requisito de que uno o más trabajos tienen que ser completados antes que otros.
- **(s_{jk}) *Sequence dependent setup times***: Representa la existencia de un tiempo de *setup* entre el trabajo *j* y *k*. Este tiempo de *setup* puede depender de cada máquina en caso de tener más de una, en este caso se denota como s_{ijk}.
- **(prmu) *Permutation***: Restricción común en los entornos Fm. Hace referencia a que el orden de procesamiento en cada máquina sigue la disciplina FIFO (*First In First Out).*
- **(nwt) *No-wait***: Restricción común en entornos Fm. Hace referencia a que los trabajos no pueden esperar entre dos máquinas sucesivas.

γ: "Describe el objetivo que se quiere minimizar". Para poder comentar los diferentes objetivos debemos definir dos conceptos: C_j (momento en el que el trabajo j termina de procesarse en la última máquina y d_j (en caso de existir, fecha en la que la empresa se ha comprometido a tener el trabajo finalizado).

Una vez definidos estos conceptos, Pinedo (2016) describe tres indicadores para cada trabajo que son útiles para los diferentes objetivos.

- L_j *Lateness:* $L_j = C_j - d_j$
- T_j *Tardiness:* $T_j = \max(L_j, 0)$
- U_j *Unit penalty:* $\begin{cases} 1 \; si \; C_j > d_j \\ 0 \; en \; otro \; caso \end{cases}$

Los objetivos (γ) más conocidos se definen en el libro de Pinedo, (2016) de la siguiente manera:

- $(\boldsymbol{C_{max}})$ ***Makespan****:* Momento en el que termina de procesarse el último trabajo en la última máquina.
- $(\sum \boldsymbol{C_j})$ ***Total completion time****:* Suma total de tiempos de finalización.
- $(\boldsymbol{L_{max}})$ ***Maximum lateness****:* Máxima violación de las fechas comprometidas.
- $(\sum \boldsymbol{T_j})$ ***Total tardiness****:* Suma de los tiempos de retraso de cada trabajo.
- $(\sum \boldsymbol{U_j})$ ***Number of tardy jobs***: Número de trabajos que van tarde.

Los objetivos planteados son los más comunes, aunque pueden aparecer más ajustándose al objetivo de la empresa. Del mismo modo, puede que se dé el caso en el que no todos los trabajos tengan la misma importancia para la empresa por diferentes motivos y por ello, se puede incluir un peso w_j en los objetivos para que se contemple la importancia relativa de cada trabajo.

2.2 Descripción del problema particular

Este TFG se centrará en el modelado y resolución de un problema con entorno *Flow shop* con *m* máquinas en el que habrá *n* trabajos que deberán ser procesados siguiendo la misma secuencia en las diferentes máquinas, es decir, con la restricción *prmu*. Adicionalmente, existe una restricción de tiempos de *setup* dependientes de la secuencia y de la máquina por lo que será de la forma s_{ijk}. Dicho *setup* será no anticipatorio, es decir, que no se puede comenzar a preparar la máquina hasta que se haya terminado de procesar el trabajo en la máquina anterior. Por último, el objetivo a minimizar es C_{max}.

La característica diferencial de este problema es que el tiempo de *setup,* que hace referencia a una puesta a punto de la máquina, lo realiza un server o un recurso humano (operario) siendo *s* (número de servers) menor al número de máquinas. La definición de server se puede encontrar en numerosos artículos de la programación de la producción como por ejemplo en el artículo de Fernandez-Viagas *et al.*, (2020) donde se define server como "un recurso reutilizable encargados de ejecutar los cambios entre trabajos o grupos de trabajos". La concepción de los servers supone un avance en la programación ya que permite el estudio de estos recursos humanos encargados del *setup* para optimizar el proceso productivo al tener un modelo que se acerque más a la realidad.

Tal y como se puede apreciar en las ilustraciones 2-1 y 2-2, la concepción de la limitación de los servers puede producir cambios en el problema que se presenta. Sea un entorno de F3 sin y con server respectivamente, con tres trabajos que han de procesarse en las 3 máquinas según la secuencia (2,3,1). Por simplificar, para este ejemplo se supone constante el tiempo de *setup* a 2 u.t.

Si se compara el *makespan* de las dos ilustraciones, se aprecia un aumento de 4 u.t. en el caso en el que hay un server.

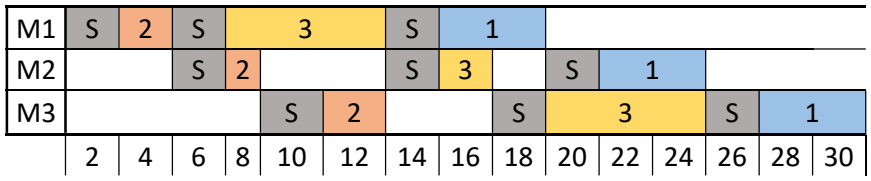

Ilustración 2-1 Diagrama de Gantt sin server ***[Fuente: Elaboración propia]***

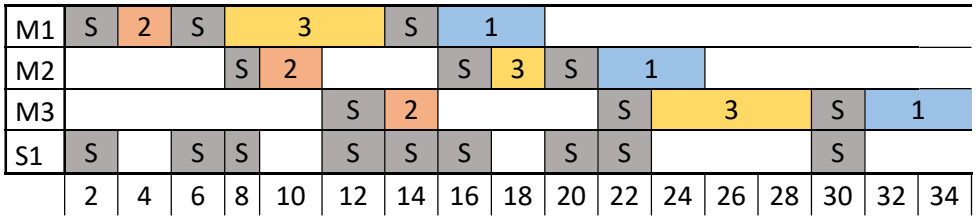

Ilustración 2-2 Diagrama de Gantt con un server ***[Fuente: Elaboración propia]***

Por todo ello, en este trabajo se tendrá en cuenta la presencia de varios servers que se formularán como máquinas paralelas cuyos trabajos son los diferentes *setups* por lo que los servers se incluirán en el entorno de las máquinas siguiendo el siguiente ejemplo: α = Pm, S2

(Werner y Kravchenko, 2010) que hace referencia a un problema con *m* máquinas paralelas y dos servers.

A modo de resumen, empleando la nomenclatura planteada en el apartado 2.1, el problema se nombrará de la siguiente manera:

$$\langle F_m, S_s \mid S_{ijk}, \text{prmu} \mid C_{max} \rangle$$

En cuanto a la complejidad de este problema, se puede afirmar que la complejidad es NP-Hard ya que, tal y como afirman en su artículo Tseng, F.T., Stafford Jr., E.F. (2008), los problemas del tipo *Flow Shop* con más de tres máquinas serán del tipo NP-Hard. Por otro lado, Gupta *et al.,* (1986) afirman que el problema $\langle F2 \mid SDST \mid C_{max} \rangle$ tiene una complejidad del tipo NP-Hard. Sin embargo, Brucker et al. (2005) afirmaron en su artículo, que la complejidad del problema específico $\langle F2, S1 \mid P_{ij} = \text{p}, S_{ij} = \text{s} \mid C_{max} \rangle$, versión simplificada del problema de este TFG, es de tipo polinomial O(n).

Capítulo 3 Modelización

3.1 Introducción

En este capítulo se presentarán diferentes modelos de programación lineal propuestos para resolver el problema de estudio. Cada uno de los modelos toma como referencia modelos previos de la literatura y, en la medida de lo posible, se realizarán sus adaptaciones respetando la misma tipología de variables. Así, para cada uno de los modelos, se planteará en primer lugar el modelo original del que parte y posteriormente, la adaptación de ese modelo para la resolución del problema $\langle F_m, S_s \mid S_{ijk}, \text{prmu} \mid C_{max} \rangle$.

En total se propondrán y analizarán cuatro modelos, tomando como referencia modelos en artículos previos de la literatura . Se comenzará con dos modelos, el Wilson y el TS2, pertenecientes a la familia Wagner del artículo de Stafford et al. (2005) planteados para el problema $\langle F_m \mid \text{prmu} \mid C_{max} \rangle$. Estos dos modelos emplean una variable de asignación de un trabajo a una posición de una secuencia.

A continuación, se adaptará un modelo desarrollado para el entorno *Flow Shop* híbrido $\langle HFS/HR \mid s_{ij} \mid C_{max} \rangle$ con restricciones de servers que ejecutan los *setups* en las diferentes etapas. El modelo por adaptar ha sido propuesto por Costa et al., (2020). Por último, se analizará y adaptará el "Modelo 1" propuesto en el artículo de Naderi *et al.* (2014), que fue planteado para un entorno de *Flow Shop* híbrido sin tiempos de *setup.*

3.2 Modelos MILP.

En primer lugar, se presentarán los datos de entrada para los modelos seguido de los índices que se emplearán. Tras esto, se expondrán las variables empleadas en los cuatro modelos adaptados por lo que variables que se emplean en los modelos originales, se explicarán en cada uno de ellos, cuando sea necesario. Por último, se presentará un ejemplo con sus datos asociados que se irá resolviendo con cada uno de los modelos adaptados con la finalidad de verificarlos y poder entender, en mayor medida, el funcionamiento de cada uno de ellos.

3.2.1 Datos

N: Número de trabajos.

M: Número de máquinas.

S: Número de servers.

$T_{i,j}$: Tiempo de proceso del trabajo j en la máquina i.

$S_{i,j,k}$: Tiempo de setup del trabajo k cuando va después del trabajo j en la máquina i

V: *"*Valor lo suficientemente grande para asegurar que solo una restricción de una relación dicotómica se cumpla*"* Sttaford, (2005).

3.2.2 Índices

i,h para máquinas. $(1 \leq i, h \leq M)$

j,k para trabajos. $(1 \leq j, k \leq N)$

p para la posición en la secuencia. $(1 \leq p \leq N)$

s para los servers. $(1 \leq s \leq S)$

3.2.3 Variables

Debido al estudio y adaptación de diversos modelos provenientes de diferentes artículos y autores se dispondrán de diversas maneras de modelar el problema con su tipología específica de variables. A continuación, se detallan todas ellas, aunque no todas se usen en todos los modelos.

$Z_{j,p} \in \{0,1\}$ 1 si el trabajo j ocupa la posición p de la secuencia, 0 en caso contrario. $(1 \leq j \leq N)$ $(1 \leq p \leq N)$.

$\alpha_{j,p,k} \in \{0,1\}$ 1 si el trabajo j está en la posición p y le sigue el trabajo k, 0 en caso contrario. $(1 \leq j,k \leq N)$ $(1 \leq p \leq N)$.

$E_{i,p}$ Tiempo de fin del trabajo en posición p en máquina i. $(1 \leq i \leq M)$ $(1 \leq p \leq N)$.

$B_{i,p}$ Tiempo de inicio del trabajo en posición p en la máquina i. $(1 \leq i \leq M)$ $(1 \leq p \leq N)$. Dadas las restricciones del problema abordado, también pueden ser consideradas con el tiempo de fin del setup del trabajo en posición p en la máquina i.

$X_{i,j,s} \in \{0,1\}$ 1 si el setup del trabajo j en la máquina i lo hace el server s, 0 en caso contrario. $(1 \leq i \leq M)$ $(1 \leq j \leq N)$ $(1 \leq s \leq S)$.

$\gamma_{s,i,p} \in \{0,1\}$ 1 si el server s realiza el setup del trabajo en posición p en máquina i, 0 en caso contrario. $(1 \leq s \leq S)$ $(1 \leq i \leq M)$ $(1 \leq p \leq N)$.

$D_{j,k} \in \{0,1\}$ 1 si el trabajo j inmediatamente antes que k, 0 en caso contrario. $(1 \leq j,k \leq N)$.

$U_{i,j}$ Tiempo de inicio del trabajo j en la máquina i (antes del setup). $(1 \leq i \leq M)(1 \leq j \leq N)$.

$C_{i,k}$ Tiempo de fin de operación del trabajo k en máquina i. $(1 \leq i \leq M)$ $(1 \leq k \leq N)$.

$F_{i,j,h,k} \in \{0,1\}$ Para los setups del trabajo j en máquina i y trabajo k en h, realizados por el mismo server. Vale 1 si el *setup* del trabajo j se procesa antes que el de k, 0 en caso contrario. $(1 \leq i,h \leq M)$ $(1 \leq j,k \leq N)$.

$Y_{s,i,j,h,k} \in \{0,1\}$ 1 si el server s hace el setup del trabajo j en la máquina i inmediatamente antes que el setup de k en la máquina h, 0 en caso contrario. $(1 \leq s \leq S)$ $(1 \leq i,h \leq M)$ $(1 \leq j,k \leq N)$.

3.2.4 Ejemplo

Para una mayor ilustración del comportamiento de los modelos, se procederá a la resolución de un problema de pequeñas dimensiones, de esta manera, se podrá comprender

mejor la filosofía que plantea cada uno de ellos. Del mismo modo, al ser modelos que buscan la solución óptima del problema, los resultados de los modelos adaptados deberán coincidir en el *makespan* y en la secuencia solución (en el caso de que únicamente hubiese un óptimo del problema, no teniendo por qué coincidir en caso contrario). El ejemplo se irá desarrollando para cada uno de los modelos analizando sus variables específicas junto al diagrama de Gantt asociado para una mejor comprensión.

Datos para el ejemplo:

Número de máquinas, M =3.

Número de trabajos, N = 4.

Número de servers, S=2.

Tiempos de proceso. Tabla 3-1.

$T_{i,j}$ Máquinas *(i)*	Trabajos *(j)*			
	1	2	3	4
1	18	12	9	10
2	10	18	15	8
3	14	8	13	17

Tabla 3-1 Tiempos de proceso para instancia de ejemplo. ***[Fuente: Elaboración propia]***

Tiempos de *setup* para la máquina 1. Tabla 3-2.

$S_{1,j,k}$	Trabajos *(k)*			
Trabajos *(j)*	1	2	3	4
0	4	6	3	4
1		6	5	7
2	2		3	3
3	2	9		5
4	5	3	9	

Tabla 3-2 Tiempos de *setup* para máquina 1 para instancia de ejemplo. ***[Fuente: Elaboración propia]***

Tiempos de *setup* para la máquina 2. Tabla 3-3.

$S_{2,j,k}$	Trabajos *(k)*			
Trabajos *(j)*	1	2	3	4
0	2	4	5	3
1		7	4	6
2	8		5	2
3	2	7		6
4	4	2	3	

Tabla 3-3 Tiempos de *setup* para máquina 2 para instancia de ejemplo. *[Fuente: Elaboración propia]*

Tiempos de *setup* para la máquina 3. Tabla 3-4.

$S_{3,j,k}$	Trabajos *(k)*			
Trabajos *(j)*	1	2	3	4
0	8	5	7	3
1		3	7	4
2	6		1	6
3	1	1		9
4	4	4	8	

Tabla 3-4 Tiempos de *setup* para máquina 3 para instancia de ejemplo. *[Fuente: Elaboración propia]*

3.3 Modelo Wilson

3.3.1 Introducción

Este modelo es un modelo de la familia Wagner del artículo de Stafford *et al.,* (2005) para el problema del tipo *Flow Shop* de permutación. Tanto este modelo como el TS2 presentado en el apartado 3.4, utilizan una variable que asigna un trabajo a una posición de una secuencia.

El modelo Wilson emplea también una variable para el tiempo en el que se inicia la operación de un trabajo en una determinada posición en una máquina ($B_{i,p}$).

Para la adaptación se usará una variable de precedencia inmediata de trabajos ($\alpha_{j,p,k}$) para el *setup* y una variable binaria de asignación de *setup* a server ($\gamma_{s,i,p}$).

3.3.2 Modelo Wilson original

$$MIN\ \ B_{M,N} + \sum_{j=1}^{N} T_{M,j} Z_{j,N}\ \ s.a: \tag{1}$$

$$\sum_{p=1}^{N} Z_{j,p} = 1\ \ \forall j \tag{2}$$

$$\sum_{j=1}^{N} Z_{j,p} = 1\ \ \forall p \tag{3}$$

$$B_{1,p} + \sum_{j=1}^{N} T_{1,j} Z_{j,p} = B_{1,p+1}\ \ \forall p\ (1, N-1) \tag{4}$$

$$B_{1,1} = 0 \tag{5}$$

$$B_{i,1} + \sum_{j=1}^{N} T_{i,j} Z_{j,1} = B_{i+1,1}\ \ \forall i\ (1, M-1) \tag{6}$$

$$B_{i,p} + \sum_{j=1}^{N} T_{i,j} Z_{j,p} \leq B_{i+1,p}\ \ \forall i\ (1, M-1)\ \ \forall p(2, N) \tag{7}$$

$$B_{i,p} + \sum_{j=1}^{N} T_{i,j} Z_{j,p} \leq B_{i,p+1}\ \ \forall i\ (2, M)\ \ \forall p(1, N-1) \tag{8}$$

$$B_{i,p} \geq 0\ (1 \leq i \leq M)(1 \leq p \leq N)$$

$$Z_{j,p}\ \in \{0,1\}\ (1 \leq j \leq N)(1 \leq p \leq N)$$

En la función objetivo (1) se encuentra el cálculo del *makespan*. Con las restricciones pertenecientes a los conjuntos (2) y (3), el modelo busca que cada trabajo se asigne solamente a una posición de la secuencia y que cada posición solo tenga un trabajo asignado. El conjunto de restricciones (4) marcan que el inicio de cada trabajo en la primera máquina se haga una vez finalizado el proceso del trabajo anterior de la secuencia. Para que el modelo sepa cuándo empezar, el conjunto (5) marca el momento de inicio del primer trabajo en la primera máquina en el instante 0. Con el conjunto de restricciones (6), el autor marca el inicio del primer trabajo en las máquinas posteriores en cuanto acabe de ser procesada en la anterior. Por último, con los conjuntos de restricciones (7) y (8) se asegura que un trabajo no comience en la siguiente máquina hasta que haya acabado en la anterior y que en cada máquina no se empiece a procesar un trabajo hasta que haya acabado el trabajo anterior.

3.3.3 Modelo Wilson adaptado

$$MIN\ \ B_{M,N} + \sum_{j=1}^{N} T_{M,j} Z_{j,N}\ \ s.a: \tag{9}$$

$$\sum_{p=1}^{N} Z_{j,p} = 1\ \ \forall j \tag{10}$$

$$\sum_{j=1}^{N} Z_{j,p} = 1\ \ \forall p \tag{11}$$

$$\sum_{k=1\ j\neq k}^{N} \alpha_{j,p,k} = Z_{j,p}\ \ \forall j, \forall p(1, N-1) \tag{12}$$

$$\sum_{k=1\ j\neq k}^{N} \alpha_{k,p-1,j} = Z_{j,p}\ \ \forall j, \forall p(2, N) \tag{13}$$

$$\alpha_{0,0,j} = Z_{j,1}\ \ \forall j \tag{13'}$$

$$B_{1,p} + \sum_{j=1}^{N}\ \sum_{k=1\ j\neq k}^{N} (S_{1,j,k} * \alpha_{j,p,k}) + \sum_{j=1}^{N} T_{1,j} Z_{j,p} \leq B_{1,p+1}\ \ \forall p\ (1, N-1) \tag{14}$$

$$B_{i,1} + \sum_{k=1}^{N} (S_{i+1,0,k} * \alpha_{0,0,k}) + \sum_{j=1}^{N} T_{i,j} Z_{j,1} \leq B_{i+1,1}\ \ \forall i\ (1, M-1) \tag{15}$$

$$B_{1,1} = \sum_{k=1}^{N} (S_{1,0,k} * \alpha_{0,0,k}) \tag{16}$$

$$B_{i,p} + \sum_{j=1}^{N}\ \sum_{k=1\ j\neq k}^{N} (S_{i+1,j,k} * \alpha_{j,p-1,k}) + \sum_{j=1}^{N} T_{i,j} Z_{i,p} \leq B_{i+1,p} \tag{17}$$

$$\forall i\ (1, M-1)\ \ \forall p(2, N)$$

$$B_{i,p} + \sum_{j=1}^{N}\ \sum_{k=1\ j\neq k}^{N} (S_{i,j,k} * \alpha_{j,p,k}) + \sum_{j=1}^{N} T_{i,j} Z_{j,p} \leq B_{i,p+1}\ \ \forall i\ (2, M)\ \ \forall p(1, N-1) \tag{18}$$

$$\sum_{s=1}^{S} \gamma_{s,i,p} = 1 \ \ \forall i, p \tag{19}$$

$$B_{i',p} - B_{i,p'} + V\left(2 - \gamma_{s,i',p} - \gamma_{s,i,p'}\right) \geq \sum_{j=0}^{N} \cdot \sum_{k=1\ j\neq k}^{N} (S_{i',j,k} * \alpha_{j,p-1,k}) \tag{20}$$

$\forall s, i, i'(2, M), \forall p, p'(2, N)\ i' > i, p' > p$

$$B_{i,p} \geq 0\ (1 \leq i \leq M)(1 \leq p \leq N)$$

$$Z_{j,p}\ \in \{0,1\}\ (1 \leq j \leq N)(1 \leq p \leq N)$$

$$\alpha_{j,p,k} \in \{0,1\}\ (1 \leq j, k \leq N)\ j \neq k (1 \leq p \leq N)$$

$$\alpha_{0,0,k} \in \{0,1\}\ (1 \leq k \leq N)$$

$$\gamma_{s,i,p} \in \{0,1\}\ (1 \leq s \leq S)(1 \leq i \leq i' \leq M)(1 \leq p \leq p' \leq N)$$

En la adaptación del modelo se encuentra, en primer lugar, el cálculo del *makespan* (9). Con los conjuntos de restricciones (10) y (11) se busca que cada trabajo se asigne a una posición de la secuencia y que cada posición solo tenga un trabajo asignado. El modelo emplea los conjuntos (12) y (13) para ajustar la variable de qué trabajo sucede inmediatamente a otro en una determinada posición haciendo que cada trabajo tenga un sucesor y predecesor, ajustando con el conjunto de restricciones (13') el trabajo inicial para contemplar el *setup* inicial. El conjunto (14) asegura que un trabajo no comience a procesarse en la primera máquina hasta que no haya acabado el proceso del trabajo anterior y se haya completado el *setup* de dicho trabajo. Con las restricciones del conjunto (15), el sistema marca el inicio del procesamiento del trabajo en la primera posición de la secuencia cuando haya acabado en la máquina anterior y se haya hecho el *setup.* El inicio del proceso del primer trabajo en la primera máquina se establece en el conjunto (16) cuando se haya completado el tiempo de *setup.* Con los conjuntos de restricciones (17) y (18) el modelo asegura que un trabajo no pase a la siguiente máquina hasta que se haya completado el proceso en la máquina anterior y se haya realizado el *setup* teniendo en cuenta que un trabajo no puede empezar en una máquina hasta que no se haya completado el trabajo que estaba secuenciado en la posición anterior y se haya realizado el *setup*. El conjunto (19) asigna a un solo server el *setup* de cada trabajo en cada máquina y con los conjuntos de restricciones (20 y 21) se asegura que cada server no comenzará un *setup* hasta que no haya completado los anteriores. El concepto que se emplea

en el conjunto de restricciones (20) es similar a la que emplea Samarghandi (2015) en el modelo matemático que desarrolla en su artículo para una instancia de características similares a las que se abordan en este TFG.

Resolución de la instancia de ejemplo.

A continuación, se presenta la solución que aborda este modelo para la instancia planteada en el punto 3.2.4. En la tabla 3-5, se puede ver tanto las variables binarias que se activan (cuyo valor será igual a 1) y las variables que hacen referencia a tiempos con sus respectivos valores. La secuencia óptima es (4,2,3,1) y el valor de la función objetivo *makespan* es 98 u.t. (unidades temporales). En la figura 3-1 se puede apreciar el diagrama de Gantt para la solución óptima en la que se representa cada trabajo en la posición correspondiente de la secuencia solución.

Posición		Asignación a server		Tiempos fin *setup*	
$Z_{4,1}$	$\alpha_{0,0,4}$	$\gamma_{1,1,1}$	$\gamma_{1,3,4}$	$B_{1,1} = 4$	$B_{2,3} = 55$
$Z_{2,2}$	$\alpha_{4,1,2}$	$\gamma_{1,2,1}$	$\gamma_{2,1,2}$	$B_{1,2} = 17$	$B_{2,4} = 71$
$Z_{3,3}$	$\alpha_{2,2,3}$	$\gamma_{1,2,2}$	$\gamma_{2,1,3}$	$B_{1,3} = 32$	$B_{3,1} = 34$
$Z_{1,4}$	$\alpha_{3,3,1}$	$\gamma_{1,3,1}$	$\gamma_{2,1,4}$	$B_{1,4} = 43$	$B_{3,2} = 55$
		$\gamma_{1,3,2}$	$\gamma_{2,2,3}$	$B_{2,1} = 17$	$B_{3,3} = 70$
		$\gamma_{1,3,3}$	$\gamma_{2,2,4}$	$B_{2,2} = 31$	$B_{3,4} = 84$

Tabla 3-5 Resultados de la instancia de ejemplo con modelo Wilson Adaptado. ***[Fuente: Elaboración propia]***

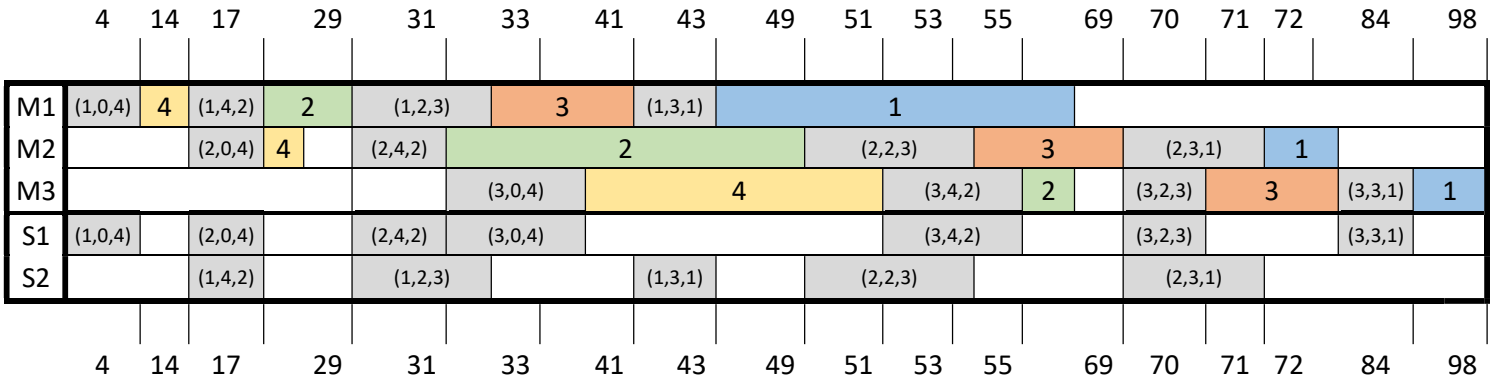

Ilustración 3-1 Diagrama de Gantt para instancia ejemplo resuelto con modelo Wilson adaptado. Para una mejor visualización de la gráfica, los tiempos de proceso no presentan escalas uniformes ***[Fuente: Elaboración propia]***

3.4 Modelo TS2

3.4.1 Introducción

El modelo TS2, a diferencia que el modelo Wilson, emplea como variable el tiempo de terminación de un trabajo en una determinada posición en cada máquina ($E_{i,p}$) en lugar del tiempo de inicio ($B_{i,p}$). Del mismo modo, se mantiene la variable de asignación de un trabajo a una posición de la secuencia ($Z_{j,p}$).

Para la adaptación, se introducirán restricciones similares a las planteadas en el apartado 3.3.3 para el modelo Wilson adaptado debido a la gran similitud entre las variables utilizadas en ambos modelos.

3.4.2 Modelo TS2 original

MIN $E_{M,N}$ $s.a:$ (1)

$\sum_{p=1}^{N} Z_{j,p} = 1 \ \forall j$ (2)

$\sum_{j=1}^{N} Z_{j,p} = 1 \ \forall p$ (3)

$E_{i,p} + \sum_{j=1}^{N} T_{i,j} Z_{j,p+1} \leq E_{i,p+1} \ \forall i, \forall p (1, N-1)$ (4)

$E_{i,p} + \sum_{j=1}^{N} T_{i+1,j} Z_{j,p} \leq E_{i+1,p} \ \forall i \ (1 \dots M-1) \ \forall p$ (5)

$\sum_{j=1}^{N} T_{1,j} Z_{j,1} \leq E_{1,1}$ (6)

$$E_{i,p} \geq 0 \ (1 \leq i \leq M)(1 \leq p \leq N)$$

$$Z_{j,p} \in \{0,1\} \ (1 \leq j \leq N)(1 \leq p \leq N)$$

La función objetivo de este modelo se calcula como el tiempo de terminación del trabajo en la última posición en la última máquina (1). Con los conjuntos de restricciones (2) y (3) se busca que cada trabajo se asigne a una posición de la secuencia y que cada posición solo tenga un trabajo asignado. Los conjuntos (4) y (5) aseguran que no se procese un trabajo

en una máquina hasta que no se haya acabo de procesar en la anterior y que no pueda empezar a procesarse un trabajo en una máquina hasta que no haya acabado el anterior en esa máquina. Con las restricciones que del conjunto (6), se marca el momento de inicio del trabajo en primera posición en la primera máquina.

3.4.3 Modelo TS2 adaptado

MIN $E_{M,N}$ $s.a:$ (7)

$\sum_{p=1}^{N} Z_{j,p} = 1 \ \forall j$ (8)

$\sum_{j=1}^{N} Z_{j,p} = 1 \ \forall p$ (9)

$\sum_{k=1 j\neq k}^{N} \alpha_{j,p,k} = Z_{j,p} \ \forall j, \forall p(1, N-1)$ (10)

$\sum_{k=1 j\neq k}^{N} \alpha_{k,p-1,j} = Z_{j,p} \ \forall j, \forall p(2, N)$ (11)

$\alpha_{0,0,k} = Z_{k,1} \ \forall k$ (11')

$E_{i,p} + \sum_{j=1}^{N} \sum_{k=1 j\neq k}^{N} (S_{i,j,k} * \alpha_{j,p,k}) + \sum_{j=1}^{N} T_{i,j} Z_{j,p+1} \leq E_{i,p+1} \ \forall i, \forall p(1, N-1)$ (12)

$E_{i,p} + \sum_{j=1}^{N} T_{i+1,j} Z_{j,p} + \sum_{j=1}^{N} \sum_{k=1 j\neq k}^{N} (S_{i+1,j,k} * \alpha_{j,p-1,k}) \leq E_{i+1,p}$ (13)

$\forall i \ (1, M-1) \ \forall p(2, N)$

$E_{i,1} + \sum_{k=1}^{N} (S_{i+1,0,k} * \alpha_{0,0,k}) + \sum_{j=1}^{N} T_{i+1,j} Z_{j,1} \leq E_{i+1,1} \ \forall i \ (1, M-1)$ (14)

$\sum_{k=1}^{N} (S_{1,0,k} * \alpha_{0,0,k}) + \sum_{j=1}^{N} T_{1,j} Z_{j,1} = E_{1,1}$ (15)

$\sum_{s=1}^{S} \gamma_{s,i,p} = 1 \ \forall i, p$ (16)

$E_{i',p} - \sum_{j=1}^{N} T_{i',j} Z_{j,p} - (E_{i,p'} - \sum_{j=1}^{N} T_{i,j} Z_{j,p'}) + V(2 - \gamma_{s,i',p} - \gamma_{s,i,p'}) \geq$
$\sum_{j=0.}^{N} \sum_{k=1 \ j\neq k}^{N} (S_{i',j,k} * \alpha_{j,p-1,k}) \quad \forall s, i, i'(2, M), \forall p, p'(2, N). i' > i, p' > p$ (17)

$$E_{i,p} \geq 0 \ (1 \leq i \leq M)(1 \leq p \leq N)$$

$$Z_{j,p} \in \{0,1\}\ (1 \leq j \leq N)(1 \leq p \leq N)$$

$$\alpha_{j,p,k} \in \{0,1\}\ (1 \leq j,k \leq N)\ j \neq k (1 \leq p \leq N)$$

$$\alpha_{0,0,k} \in \{0,1\}\ (1 \leq k \leq N)$$

$$\gamma_{s,i,p} \in \{0,1\}\ (1 \leq s \leq S)(1 \leq i \leq i' \leq M)(1 \leq p \leq p' \leq N)$$

En el modelo TS2 adaptado se mantiene la función objetivo como el tiempo de finalización del proceso del trabajo secuenciado en última posición en la última máquina (7). Con los conjuntos de restricciones (8-11) se establece la asignación de trabajos a posiciones de la secuencia y se establece la precedencia y sucesión de cada trabajo en cada posición, teniendo en cuenta con el conjunto (11') el trabajo en la primera posición para el *setup* inicial. Los conjuntos de restricciones (12) y (13) regulan que un trabajo no comience en la siguiente máquina hasta que haya acabado en la anterior y se haya realizado el *setup* y que un trabajo no empiece a procesarse en una máquina hasta que no haya acabado el secuenciado antes y se haya realizado el *setup.* Los conjuntos (14 y 15) marcan el momento de inicio del primer trabajo secuenciado en la primera máquina y que el paso de ese trabajo por las máquinas posteriores. Con las restricciones del conjunto (16), se asigna cada *setup* a un único server. Por último, el conjunto de restricciones (17) regulan la restricción de los servers siguiendo la filosofía empleada en el apartado 3.3.3 para el modelo adaptado de Wilson.

Resolución de la instancia de ejemplo.

A continuación, se presenta la solución que plantea este modelo para la instancia planteada en el punto 3.2.4. La secuencia óptima es (4,2,3,1). En la tabla 3-6 se presentan las variables binarias que se activan (con valor igual a 1) y las variables que hacen referencia a tiempos con los valores que toman. El valor del *makespan* coincide con el presentado en el modelo de Wilson de 98 u.t. En la figura 3-2 se puede apreciar el diagrama de Gantt para la solución óptima en la que se representa cada trabajo en la posición correspondiente de la secuencia solución.

Posición		Asignación a server		Tiempos fin proceso	
$Z_{4,1}$	$\alpha_{0,0,4}$	$\gamma_{1,1,1}$	$\gamma_{1,3,4}$	$E_{1,1} = 14$	$E_{2,3} = 69$
$Z_{2,2}$	$\alpha_{4,1,2}$	$\gamma_{1,1,2}$	$\gamma_{2,2,1}$	$E_{1,2} = 29$	$E_{2,4} = 81$
$Z_{3,3}$	$\alpha_{2,2,3}$	$\gamma_{1,1,3}$	$\gamma_{2,2,2}$	$E_{1,3} = 41$	$E_{3,1} = 51$
$Z_{1,4}$	$\alpha_{3,3,1}$	$\gamma_{1,1,4}$	$\gamma_{2,3,1}$	$E_{1,4} = 61$	$E_{3,2} = 63$
		$\gamma_{1,2,3}$	$\gamma_{2,3,2}$	$E_{2,1} = 25$	$E_{3,3} = 83$
		$\gamma_{1,2,4}$	$\gamma_{2,3,3}$	$E_{2,2} = 49$	$E_{3,4} = 98$

Tabla 3-6 Resultados de la instancia de ejemplo con modelo TS2 Adaptado. ***[Fuente: Elaboración propia]***

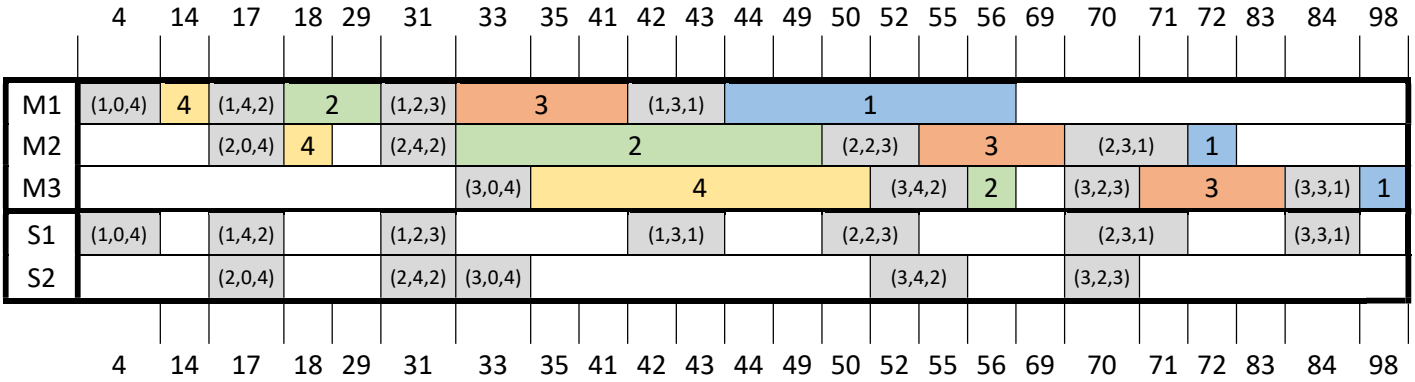

Ilustración 3-2 Diagrama de Gantt para instancia ejemplo resuelto con modelo TS2 adaptado. Para una mejor visualización de la gráfica, los tiempos de proceso no presentan escalas uniformes. ***[Fuente: Elaboración propia]***

3.5 Modelo HFS/HR

3.5.1 Introducción.

Este modelo presentado en el artículo de Costa *et al.*, (2020), está diseñado para un entorno híbrido con diferentes etapas que se denotará con el índice *w*. Dentro de cada etapa hay un número definido de máquinas paralelas. El modelo está diseñado de forma que cada etapa dispone de un número determinado de servers que atenderán los *setups* (independientes de la secuencia y de la máquina) de las diferentes máquinas dentro de su etapa.

De cara a la adaptación, se considerará que cada etapa consta de una única máquina lo que significa que hay una equivalencia entre etapa y máquina. Del mismo modo, se considerará que los servers puedan operar en cualquier máquina del sistema.

3.5.2 Modelo HFS/HR original

Además de las variables de la Sección 3.2.3, este modelo emplea las siguientes variables, ajustadas a la notación empleada en este TFG:

$X_{w,i,j,s} \in \{0,1\}$ 1 si el server s realiza el *setup* del trabajo j en la máquina i en la etapa w.

$F_{w,k,j} \in \{0,1\}$ Para los trabajos j y k cuyos *setup* los realice el mismo server en la etapa w, 1 si k precede a j.

$G_{w,k,j} \in \{0,1\}$ En caso en el que los trabajos j y k se procesen en la misma máquina en la etapa w, 1 si k precede a j.

$U_{w,j}$ Tiempo de inicio del trabajo j en la etapa w.

$C_{w,j}$ Tiempo finalización del proceso del trabajo j en la etapa w.

C_{max} *Makespan.*

MIN $C_{max}\ s.a$:

$$\sum_{i=1}^{m_w}\sum_{s=1}^{s_w} X_{w,i,j,s} = 1\ \ \forall w,j \tag{1}$$

$$F_{w,k,j} + F_{w,j,k} \geq \sum_{i=1}^{m_w}(X_{w,i,j,s} + X_{w,i,k,s}) - 1 \quad \forall w,j,k\ (1 \leq s \leq s_w)\ j \neq k \tag{2}$$

$$G_{w,k,j} + G_{w,j,k} \geq \sum_{s=1}^{s_w}(X_{w,i,j,s} + X_{w,i,k,o}) - 1\ \ \forall w,j,k\ (1 \leq s \leq s_w)\ (1 \leq i \leq m_w) \tag{3}$$

$$U_{w,j} \geq C_{w,k} - M\left(1 - G_{w,k,j}\right)\ \forall w,j,k \tag{4}$$

$$U_{w,j} \geq U_{w,k} + S_{w,k} - M\left(1 - F_{w,k,j}\right)\ \forall w,j,k \tag{5}$$

$$C_{w,j} \geq C_{(w-1),j} + T_{w,j}\ \forall w(2,W)\ \forall j \tag{6}$$

$$C_{w,j} \geq U_{w,j} + S_{w,j} + T_{w,j}\ \forall w,j \tag{7}$$

$$C_{max} \geq C_{W,j}\ \forall j \tag{8}$$

$$X_{w,i,j,s} \in \{0,1\}\ (1 \leq w \leq W)(1 \leq j \leq N)(1 \leq k \leq N)\ (1 \leq i \leq m_w)$$

$$F_{w,j,k}\ , G_{w,j,k} \in \{0,1\}\ (1 \leq w \leq W)(1 \leq j \leq N)(1 \leq k \leq N)$$

$$U_{w,j}, C_{w,j} \geq 0\ (1 \leq w \leq W)(1 \leq j \leq N)$$

Con el conjunto de restricciones (1), el sistema asegura que cada trabajo está asignado a una máquina en cada etapa y a un server de los disponibles en esa etapa para que realice el setup. El conjunto (2) impone que para aquellos trabajos cuyos *setup* los realice el mismo server, uno vaya antes que el otro. Del mismo modo, el conjunto (3) emplea la misma filosofía, pero para dos trabajos que se procesen en la misma máquina. El conjunto de restricciones (4) asegura que un trabajo no empiece a ser procesado en una máquina hasta que los anteriores trabajos asignados a la misma máquina se hayan completado. Del mismo modo, el conjunto (5) aplica la misma filosofía, pero para los *setups* de dos trabajos asignados al mismo server en el que uno no puede comenzar hasta que se haya completado el anterior. Con las restricciones del conjunto (6), el sistema asegura que un trabajo no se procesa en la siguiente etapa hasta que no se haya terminado de procesar en la anterior. La relación entre el tiempo

de inicio del procesamiento de un trabajo en una etapa con el tiempo de finalización se modela con el conjunto (7). Por último, con el conjunto de restricciones (8) se calcula el *makespan.*

3.5.3 Modelo HFS/HR adaptado

$MIN\ C_{max}\ \ s.a$:

$$\sum_{j=0\ j\neq k}^{N} D_{j,k} = 1\ \ \forall k \tag{9}$$

$$\sum_{k=1\ j\neq k}^{N} D_{j,k} \leq 1\ \ \forall j\ (0,N) \tag{10}$$

$$\sum_{h=0\ h\neq j}^{N} D_{h,j} \geq\ D_{j,k}\ \ \forall j,k,j\neq k \tag{11}$$

$$\sum_{s=1}^{S} X_{i,j,s} = 1\ \forall i,j \tag{12}$$

$$F_{i,j,h,k} + F_{h,k,i,j} \geq \left(X_{i,j,s} + X_{h,k,s}\right) - 1\ \forall h,i,j,k,s\ (i,j\neq h,k) \tag{13}$$

$$U_{i,k} \geq C_{i,j} - V\left(1 - D_{j,k}\right)\ \forall i,j(0,N),k,j\neq k \tag{14}$$

$$U_{h,k} \geq\ U_{i,j} + \sum_{l=0,l\neq j}^{N} S_{i,l,j} * D_{l,j} - V\left(1 - F_{i,j,h,k}\right)\forall i,h,j,k\ \ (i,j\neq h,k) \tag{15}$$

$$C_{i,j} \geq U_{i,j} + \sum_{k=0\ j\neq k}^{N} S_{i,k,j} * D_{k,j} + T_{i,j}\ \forall i,j \tag{16}$$

$$C_{i-1,j} \leq U_{i,j}\ \forall i(2,M)\forall j \tag{17}$$

$$C_{i,0} = U_{i,0} = 0\ \forall i \tag{18 y 19}$$

$$C_{max} \geq C_{M,j}\ \forall j \tag{20}$$

$$X_{i,j,s} \in \{0,1\}\ (1\leq i\leq M)(1\leq j\leq N)(1\leq s\leq S)$$

$$F_{i,j,h,k} \in \{0,1\}\ (1\leq i,h\leq M)(1\leq j,k\leq N)\ (i,j\neq h,k)$$

$$D_{j,k} \in \{0,1\}\ (0\leq j\leq N)(1\leq k\leq N)(j\neq k)$$

$$U_{i,j}, C_{i,j} \geq 0\ (1\leq i\leq M)(0\leq j\leq N)$$

Los tres primeros conjuntos de restricciones del modelo (9, 10 y 11) ajustan la precedencia y sucesión de un trabajo respecto a otro. Con el conjunto (12), el modelo asegura

que el *setup* de cada trabajo en cada máquina se asigna a un único server. El conjunto (13) impone que para aquellos trabajos cuyos *setup* los realice el mismo server, uno vaya antes que el otro. El conjunto de restricciones (14) impone que un trabajo no puede empezar hasta que el que iba inmediatamente antes no haya acabado. Las restricciones que imponen los servers de que solo pueden procesar un *setup* a la vez se ve reflejada en el conjunto (15). El modelo impone con el conjunto de restricciones (16) la relación entre el tiempo de inicio de procesamiento de un trabajo en una máquina con su tiempo de finalización. EL conjunto (17) asegura que un trabajo no comienza a procesarse en la siguiente máquina hasta que no haya acabado en la anterior. Por último, los conjuntos de restricciones (18 y 19) establecen los tiempos de inicio y acabado del trabajo *dummy* inicial mientras que el conjunto (20) calcula el *makespan.*

Resolución de la instancia de ejemplo.

A continuación, se presenta la solución que plantea este modelo para la instancia planteada en el punto 3.2.4. En la tabla 3-7 se presentan las variables binarias que se activan en el modelo HFSHR adaptado, así como los valores de las variables que hacen referencia a tiempos. Como se puede comprobar, tanto la secuencia óptima como el valor del *makespan* coinciden con los presentados en los modelos de Wilson y TS2 en los apartados 3.3.3 y 3.4.3 siendo (4,2,3,1) y *makespan* igual a 98 u.t. La figura 3-3 se corresponde con el diagrama de Gantt para la solución óptima en la que se representa cada trabajo en la posición correspondiente de la secuencia solución.

Posición	Asignación a server		Tiempos inicio *setup*		Tiempos fin proceso	
$D_{0,4}$	$X_{1,1,2}$	$X_{2,3,1}$	$U_{1,1}=51$	$U_{2,3}=49$	$C_{1,1}=71$	$C_{2,3}=69$
$D_{4,2}$	$X_{1,2,2}$	$X_{2,4,2}$	$U_{1,2}=14$	$U_{2,4}=18$	$C_{1,2}=29$	$C_{2,4}=29$
$D_{2,3}$	$X_{1,3,2}$	$X_{3,1,1}$	$U_{1,3}=34$	$U_{3,1}=83$	$C_{1,3}=46$	$C_{3,1}=98$
$D_{3,1}$	$X_{1,4,1}$	$X_{3,2,2}$	$U_{1,4}=0$	$U_{3,2}=57$	$C_{1,4}=14$	$C_{3,2}=69$
	$X_{2,1,2}$	$X_{3,3,1}$	$U_{2,1}=71$	$U_{3,3}=69$	$C_{2,1}=83$	$C_{3,3}=83$
	$X_{2,2,2}$	$X_{3,4,2}$	$U_{2,2}=29$	$U_{3,4}=37$	$C_{2,2}=49$	$C_{3,4}=57$

Precedencia en ejecución de los *setups*

$F_{1,1,2,1}$	$F_{1,2,2,3}$	$F_{1,3,3,1}$	$F_{1,4,2,4}$	$F_{2,2,3,1}$	$F_{2,4,1,1}$	$F_{3,2,2,1}$
$F_{1,1,3,1}$	$F_{1,2,2,4}$	$F_{1,3,3,2}$	$F_{1,4,3,1}$	$F_{2,2,3,2}$	$F_{2,4,1,3}$	$F_{3,2,3,3}$

$F_{1,1,3,2}$	$F_{1,2,3,1}$	$F_{1,3,3,3}$	$F_{1,4,3,2}$	$F_{2,2,3,3}$	$F_{2,4,2,1}$	$F_{3,3,3,1}$
$F_{1,1,3,3}$	$F_{1,2,3,2}$	$F_{1,3,3,4}$	$F_{1,4,3,3}$	$F_{2,2,3,4}$	$F_{2,4,2,2}$	$F_{3,4,1,1}$
$F_{1,2,1,1}$	$F_{1,2,3,4}$	$F_{1,4,1,1}$	$F_{2,1,3,1}$	$F_{2,3,2,1}$	$F_{2,4,2,3}$	$F_{3,4,2,1}$
$F_{1,2,1,3}$	$F_{1,3,1,1}$	$F_{1,4,1,3}$	$F_{2,2,1,1}$	$F_{2,3,3,1}$	$F_{2,4,3,2}$	$F_{3,4,2,3}$
$F_{1,2,2,1}$	$F_{1,3,2,1}$	$F_{1,4,2,2}$	$F_{2,2,1,3}$	$F_{2,3,3,2}$	$F_{2,4,3,3}$	$F_{3,4,3,1}$
$F_{1,2,2,2}$	$F_{1,3,2,3}$	$F_{1,4,2,3}$	$F_{2,2,2,1}$	$F_{2,3,3,3}$	$F_{2,4,3,4}$	$F_{3,4,3,2}$

Tabla 3-7 Resultados de la instancia de ejemplo con modelo HFSHR Adaptado. ***[Fuente: Elaboración propia]***

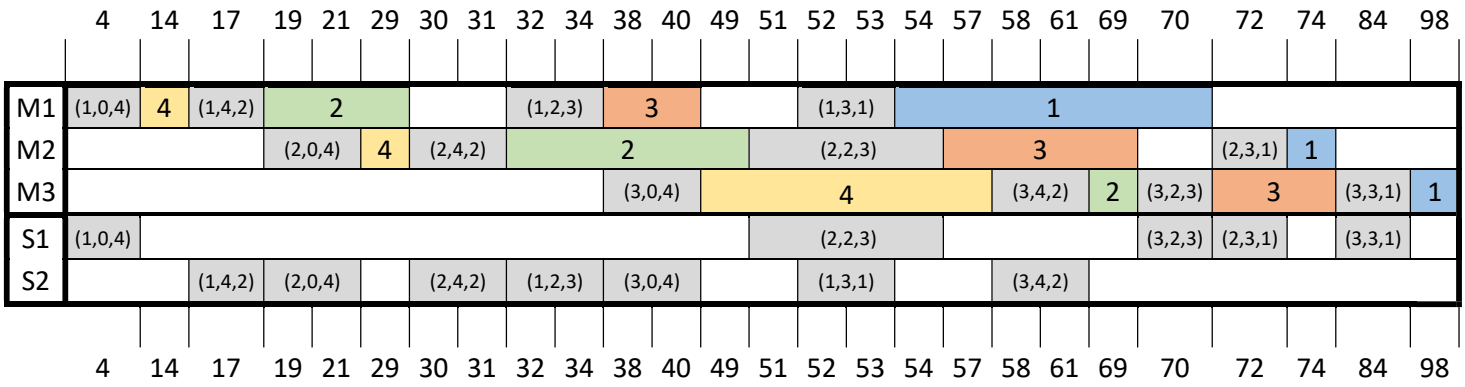

Ilustración 3-3 Diagrama de Gantt para instancia ejemplo resuelto con modelo HFSHR adaptado. Para una mejor visualización de la gráfica, los tiempos de proceso no presentan escalas uniformes. ***[Fuente: Elaboración propia]***

3.6 Modelo Naderi (2014) Modelo 1

3.6.1 Introducción

Este modelo del artículo de Naderi *et al.,* (2014) está planteado para la resolución de un entorno *Flow Shop* híbrido flexible, es decir, que todos los trabajos tienen que ser procesados en todas las etapas, pero únicamente en una de las máquinas disponibles en cada etapa. El índice que se empleará para las etapas será *w*. Para la adaptación se empleará una variable binaria de precedencia inmediata de dos *setup* asignados al mismo server.

3.6.2 Naderi (2014) Modelo 1 original.

El modelo emplea las siguientes variables:

$Y_{w,i,j,k}$ 1 si el trabajo j de procesa inmediatamente antes que k en la máquina i en la etapa w

MIN C_{max} $s.a$:

$$\sum_{k\epsilon\{0\cup E_w\},j\neq k}\sum_{i=1}^{m_w} Y_{w,i,j,k} = 1\ \forall w, j\epsilon E_w \quad (1)$$

$$\sum_{j\epsilon E_w,j\neq k}\sum_{i=1}^{m_w} Y_{w,i,j,k} \leq 1\ \forall w, k\epsilon E_w \quad (2)$$

$$\sum_{j\epsilon E_w} Y_{w,i,0,j} = 1\ \forall w, i \quad (3)$$

$$\sum_{j\epsilon E_w,j\neq k} Y_{w,i,j,k} \leq \sum_{j\epsilon\{0\cup E_w\},j\neq k} Y_{w,i,k,j}\ \ \forall w, i, k\ \epsilon E_w \quad (4)$$

$$\sum_{i=1}^{m_w}(Y_{w,i,j,k} + Y_{w,i,k,j}) \leq 1\ \forall w, (j,k)\epsilon E_w \quad (5)$$

$$C_{1,j} \geq T_{1,j}\ \ \forall j\epsilon E_w \quad (6)$$

$$C_{w,j} \geq T_{w-1,j}\ \ \forall w(2,W)\forall j\epsilon E_w \quad (7)$$

$$C_{w,j} \geq C_{w,k} + T_{w,j} - V(1-\sum_{i=1}^{m_w} Y_{w,i,j,k})\ \ \forall w, \forall(j,k)\epsilon E_w, j\neq k \quad (8)$$

$$C_{max} \geq C_{W,j}\ \forall j \quad (9)$$

$$C_{w,j} \geq 0\ \forall w, j$$

$$Y_{w,i,j,k} \in \{0,1\}\ \forall w, i, j \epsilon E_w, j \epsilon \{0 \cup E_w\}, j \neq k$$

El conjunto de restricciones (1) asegura que, en cada etapa, cada trabajo tiene un solo predecesor y con el conjunto (2), que como mucho, cada trabajo tiene un sucesor. Con las restricciones del conjunto (3), el modelo impone que hay un único trabajo inicial. Estas tres restricciones expuestas anteriormente, se modelan para todas las etapas asegurando que solo hay una máquina en cada etapa que lo cumple con el conjunto (4). Las restricciones pertenecientes al conjunto (5) imponen la precedencia inmediata para los trabajos que se procesen en la misma máquina. El conjunto (6) marca el tiempo de finalización del proceso de los trabajos en la primera etapa para que, con el conjunto de restricciones (7), se asegure de que un trabajo no pase a la siguiente etapa hasta que no se haya completado en la anterior. El conjunto (8) asegura que se cumple la relación de dos trabajos consecutivos cumple la secuencia. Por último, el conjunto de restricciones (9) calcula el *makespan.*

3.6.3 Naderi (2014) Modelo 1 adaptado

$Y_{s,i,j,h,k} = 1$ Si el server s hace el setup del trabajo j en la máquina i inmediatamente antes que el setup de k en la máquina h

MIN C_{max} $s.a$:

$$\sum_{j=0\ j\neq k}^{N} D_{j,k} = 1\ \forall k \quad (10)$$

$$\sum_{k=1\ j\neq k}^{N} D_{j,k} \leq 1\ \forall j\ (0,N) \quad (11)$$

$$\sum_{h=0\ h\neq j}^{N} D_{h,j} \geq D_{j,k}\ \forall j,k, j \neq k \quad (12)$$

$$C_{1,k} \geq T_{1,k} + \sum_{j=0\ j\neq k}^{N} (S_{1,j,k} * D_{j,k})\ \forall k \quad (13)$$

$$C_{i,k} - C_{i-1,k} \geq T_{i,k} + \sum_{j=0\ j\neq k}^{N} (S_{i,j,k} * D_{j,k})\ \forall i(2,M)\ \forall k \quad (14)$$

$$C_{i,k} - C_{i,j} + V(1 - D_{j,k}) \geq T_{i,k} + S_{i,j,k}\ \forall j(0,N) \forall k, j \neq k, \forall i \quad (15)$$

$$\sum_{s=1}^{S} (Y_{s,0,0,h,k} + \sum_{i=1}^{M} \sum_{j=1}^{N} Y_{s,i,j,h,k}) = 1\ \forall h,k\ (i,j \neq h,k) \quad (16)$$

$$\sum_{s=1}^{S} \sum_{h=1}^{M} \sum_{k=1}^{N} Y_{s,i,j,h,k} \leq 1\ \forall i,j\ (i,j \neq h,k) \quad (17)$$

$$\sum_{h=1}^{M}\sum_{k=1}^{N} Y_{s,0,0,h,k} = 1\ \forall s \quad (18)$$

$$\sum_{o=1}^{M}\sum_{l=1}^{N} Y_{s,o,l,i,j} + Y_{s,0,0,i,j} \geq \sum_{h=1}^{M}\sum_{k=1}^{N} Y_{s,i,j,h,k}\ \ \forall s,i,j\ (i,j \neq h,k)\ (i,j \neq o,l) \quad (19)$$

$$C_{h,k} - T_{h,k} + V\left(1 - Y_{s,i,j,h,k}\right) \geq C_{i,j} - T_{i,j} + \sum_{l=0\ l\neq k}^{N}(S_{h,l,k} * D_{l,k}) \quad (20)$$

$\forall s,h,i,j,k\ (i,j \neq h,k)$

$$C_{h,k} - T_{h,k} + V\left(1 - Y_{s,0,0,h,k}\right) \geq \sum_{l=0\ l\neq k}^{N}(S_{h,l,k} * D_{l,k})\ \ \forall s,h,k \quad (21)$$

$$C_{max} \geq C_{M,j}\ \forall j \quad (22)$$

$$D_{j,k} \in \{0,1\}\ (0 \leq j \leq N)(1 \leq k \leq N)(j \neq k)$$

$$C_{i,j} \geq 0\ \forall i,j(0,N)$$

$$Y_{s,i,j,h,k}\ \in \{0,1\}\ \forall s,h,i,j,k(i,j \neq h,k)$$

$$Y_{s,0,0,h,k}\ \in \{0,1\}\ \forall s,h,k$$

En el modelo adaptado, los conjuntos de restricciones (10) y (11) aseguran que cada trabajo tenga un predecesor y un sucesor mientras que el conjunto (12), restringe a que cada trabajo tenga como máximo un sucesor. El conjunto de restricciones (13) asegura que todos los trabajos se procesen en la primera máquina. Con las restricciones pertenecientes al conjunto (14) y (15), se impone que un trabajo no se procese en la siguiente máquina hasta que no haya finalizado en la anterior y se haya realizado el *setup* y que no se empiece a procesar un trabajo hasta que no haya acabado de procesarse en esa máquina el trabajo anterior. Las restricciones de los conjuntos (16-19) modelan la variable de asignación de *setup* a servers sabiendo qué *setup* precede inmediatamente a otro. Finalmente, con los conjuntos (20 y 21) se impone que cada server solo puede procesar un *setup* a la vez mientras que el conjunto de restricciones (22) calcula el *makespan.*

Resolución de la instancia de ejemplo.

A continuación, se presenta la solución que plantea este modelo para la instancia planteada en el punto 3.2.4. En la tabla 3-8 se presentan las variables binarias que se activan

en el modelo Naderi (2014) “Modelo 1” adaptado, así como los valores de las variables que hacen referencia a tiempos. La secuencia óptima es (4,2,3,1) y el valor de la función objetivo es 98 u.t. En la figura 3-4 corresponde con el diagrama de Gantt para la solución óptima en la que se representa cada trabajo en la posición correspondiente de la secuencia solución.

Posición	Tiempos fin proceso		Precedencia *setups* en servers	
$D_{0,4}$	$C_{1,1} = 61$	$C_{2,3} = 69$	$Y_{1,0,0,1,4}$	$Y_{1,2,4,1,3}$
$D_{4,2}$	$C_{1,2} = 29$	$C_{2,4} = 25$	$Y_{2,0,0,1,2}$	$Y_{1,3,3,2,1}$
$D_{2,3}$	$C_{1,3} = 41$	$C_{3,1} = 98$	$Y_{1,1,3,2,3}$	$Y_{2,1,1,3,2}$
$D_{3,1}$	$C_{1,4} = 14$	$C_{3,2} = 63$	$Y_{1,1,4,2,4}$	$Y_{2,1,2,2,2}$
	$C_{2,1} = 83$	$C_{3,3} = 83$	$Y_{1,2,1,3,1}$	$Y_{2,2,2,3,4}$
	$C_{2,2} = 49$	$C_{3,4} = 51$	$Y_{1,2,3,3,3}$	$Y_{2,3,4,1,1}$

Tabla 3-8 Resultados de la instancia de ejemplo con modelo Naderi (2014) "Modelo 1" adaptado. ***[Fuente: Elaboración propia]***

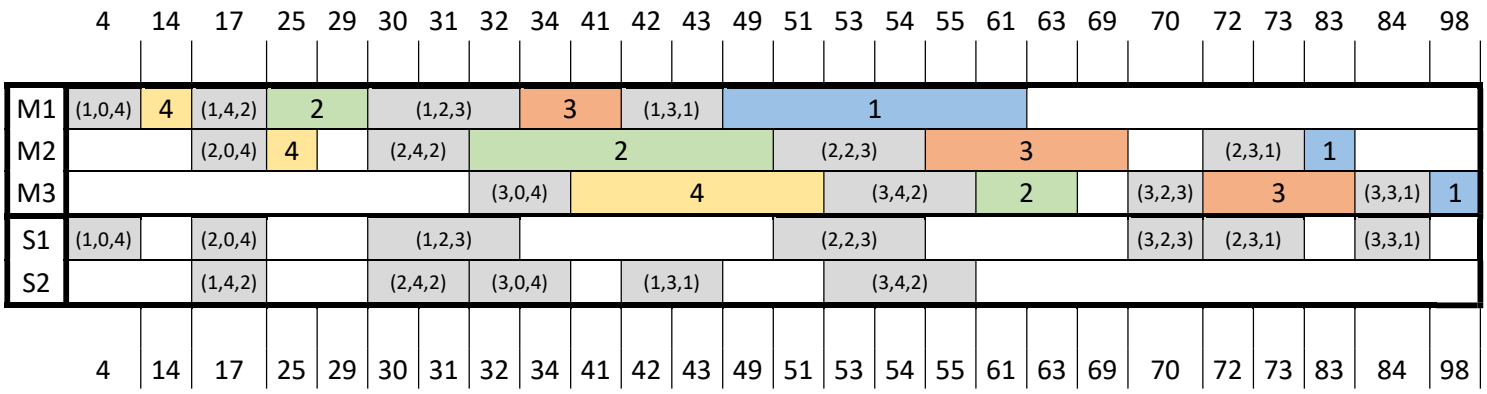

Ilustración 3-4 Diagrama de Gantt para instancia ejemplo resuelto con modelo Naderi (2014) adaptado. Para una mejor visualización de la gráfica, los tiempos de proceso no presentan escalas uniformes. ***[Fuente: Elaboración propia]***

Capítulo 4 Resultados y análisis

4.1 Resolución de los modelos

Una vez diseñado los diferentes modelos, se procederá a su resolución computacional en instancias de diferentes datos y tamaños con el fin de poder realizar un análisis sobre los resultados arrojados para determinar el comportamiento de cada modelo, observando la solución que alcanza y el tiempo que tarda en llegar a ella.

En total, se dispondrá de 720 problemas diferentes que deberá resolver cada modelo en los que varían el número de trabajos, de máquinas, de servers y los rangos de los tiempos de *setup.*

Los datos de las instancias se obtendrán de las siguientes combinaciones:

- Trabajos: 4, 5, 7, 10, 15, 20.
- Máquinas: 4, 5, 6.
- Servers: 2, 3.
- Tiempo máximo de setup (u.t.): 10, 50, 100, 125.

Para cada combinación de estos datos, se resolverán cinco instancias diferentes.

Por otro lado, en cuanto a los datos de entrada (tiempos de proceso y tiempos de *setup*), el sistema los generará aleatoriamente a través de la función de C# *int Random. Next (range)*. La variable local *int Random* irá asociada a una semilla que se irá actualizando para cada instancia, entendiéndose como semilla un número que inicializa la función de generación pseudoaleatoria. El rango para los tiempos de proceso será (1, 100) u.t.

Del mismo modo, los tiempos de *setup* se generarán de manera aleatoria empleando el mismo procedimiento que para el tiempo de proceso con la salvedad de que el rango variará en las instancias, generándose en base las distribuciones uniformes U[1,10], U[1,50], U[1,100] y U[1,125] en función del tiempo máximo de setup establecido.

Por último, se establece como tiempo límite de computación para cada problema, 500 segundos, de modo que en caso de alcanzar el tiempo límite, el sistema devuelve el mejor de la función objetivo encontrado hasta el momento. Se establecerá el tiempo límite debido a la

complejidad y extensión en número de variables y restricciones en problemas con un alto número de trabajos y máquinas.

La resolución de las instancias se hará empleando el programa de optimización Gurobi en su versión 9.0.2-win64 del 2019 en un ordenador portátil con procesador Intel® Core™ i7-4510U de hasta 3,10GHz, con una memoria RAM de 16 GB y con sistema operativo Windows 10 Home de 64 bits.

Los resultados de cada instancia se generarán de manera automática en un archivo *.csv* que, una vez completado será exportado a *Excel* para el posterior análisis de los resultados. Para cada problema resuelto por cada modelo se obtendrán los siguientes parámetros:

- Valor de la función objetivo (*makespan).*
- Tiempo de computación empleado.
- Indicador de haber alcanzado solución óptima.
- Indicador de haber encontrado una solución factible sin ser la óptima.
- Indicador de infactibilidad.
- Indicador de error de otro tipo.

4.2 Análisis de resultados

Para el análisis de los resultados, se tendrán en cuenta tres factores con los que se podría sacar una visión del funcionamiento global de los modelos. Los factores son:

- Número de óptimos, soluciones factibles e infactibles. Para ellos se contabilizará, para el total de las 720 instancias, cuántas veces ha llegado el sistema a la solución óptima, cuántas no ha llegado a la óptima, pero ha llegado a una solución factible y cuántas no ha llegado a una solución factible en los 500 segundos que se han establecido como tiempo límite.
- Tiempo medio de computación (*CPU time)*. Empleando este criterio, se puede analizar tanto el tiempo medio que ha empleado cada modelo como para, empleando los

resultados del número de soluciones óptimas, analizar la eficiencia de los diferentes modelos.

- ARPD (*Average Relative Percentage Deviation)*. Este indicador se emplea para calcular en porcentaje, cuánto se ha desviado la solución propuesta por un modelo respecto a la mejor solución encontrada. Típicamente, la mejor solución encontrada será el óptimo del problema. No obstante, para los casos en el que los cuatro modelos no hayan encontrado la solución óptima se tomará como valor la mejor solución de las cuatro propuestas.

4.2.1 Análisis de las soluciones según tipología

En la tabla 4.1 se puede ver el número de soluciones óptimas, factibles e infactibles de cada uno de los modelos respecto al tamaño de la instancia (número de trabajos y máquinas). En el apartado 4.1 se mencionó la captura en resultados de un indicador de otro tipo error para los casos en los que, por ejemplo, el modelo cuya región admisible no estuviera acotada pero dicho indicador no se ha activado en ninguna instancia y modelo por lo que no se comentará.

		Wilson			TS2			HFSHR			Naderi (2014)		
N	M	Ópt.	Fac.	Inf.	Ópt.	Fac.	Inf.	Ópt.	Fac.	Inf.	Ópt.	Fac.	Inf.
4	4	40	0	0	40	0	0	40	0	0	40	0	0
	5	40	0	0	40	0	0	40	0	0	40	0	0
	6	40	0	0	40	0	0	40	0	0	40	0	0
5	4	40	0	0	40	0	0	40	0	0	40	0	0
	5	40	0	0	40	0	0	40	0	0	40	0	0
	6	40	0	0	40	0	0	40	0	0	40	0	0
7	4	40	0	0	40	0	0	40	0	0	40	0	0
	5	40	0	0	40	0	0	40	0	0	22	18	0
	6	40	0	0	40	0	0	40	0	0	25	15	0
10	4	40	0	0	40	0	0	0	40	0	0	7	33
	5	40	0	0	40	0	0	0	40	0	0	0	40
	6	40	0	0	40	0	0	0	40	0	0	0	40
15	4	40	0	0	40	0	0	0	40	0	0	0	40
	5	40	0	0	40	0	0	0	40	0	0	0	40
	6	40	0	0	40	0	0	0	40	0	0	0	40
20	4	34	6	0	34	6	0	0	40	0	0	0	40
	5	1	39	0	1	39	0	0	40	0	0	0	40
	6	0	40	0	0	40	0	0	40	0	0	0	40
Total		635	85	0	635	85	0	360	360	0	327	40	353

%	88,2	11,8	0	88,2	11,8	0	50	50	0	45,4	5,56	49

Tabla 4-1 Análisis según tipología de soluciones. ***[Fuente: Elaboración propia]***

En la ilustración 4.1 se puede apreciar el porcentaje que representan los valores de la tabla 4.1 de manera gráfica en un diagrama de barras apiladas.

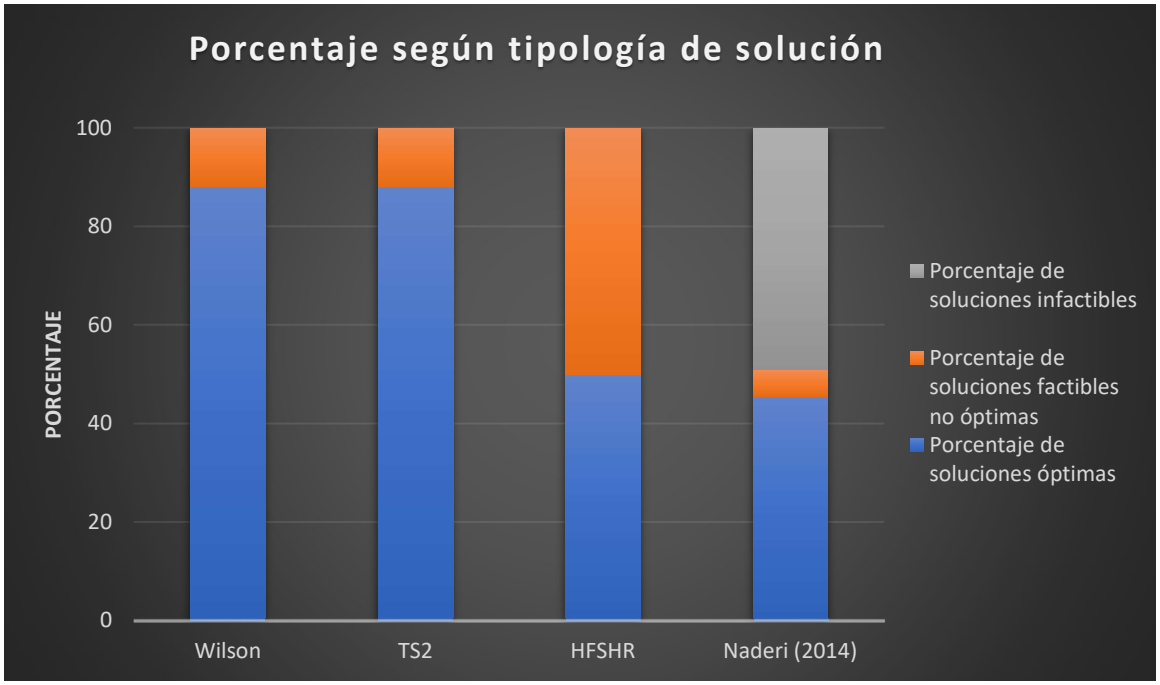

Ilustración 4-1 Diagrama porcentaje según tipología de solución. ***[Fuente: Elaboración propia]***

Analizando los resultados del apartado 4.2.1 se puede apreciar a primera vista una gran similitud entre los modelos Wilson y TS2 que aportan más de un 80% de soluciones óptimas en las diferentes instancias mientras que los modelos HFSHR y Naderi (2014) apenas llegan al 50% de soluciones óptimas. Cabe destacar también el gran número de soluciones infactibles del último modelo a partir de los 10 trabajos. Como se puede intuir, el tiempo que empleará el sistema en el último modelo será elevado debido a que ha consumido los 500 segundos sin encontrar una solución factible. Por último, hay que destacar también la no presencia de soluciones infactibles en los modelos Wilson, TS2 y HFSHR al aumentar el tamaño de las instancias.

4.2.2 Análisis del tiempo de computación

Para cada modelo, se calculará el promedio de los tiempos de computación medido en segundos que ha empleado el sistema en cada instancia con sus respectivos datos de máquinas, trabajos y servers. Se emplea para analizar el efecto que tiene el aumento del problema en el tiempo de computación. Así mismo, se presenta en la tabla 4.2 los tiempos medios para los diferentes tamaños, así como la media total y la desviación típica.

N	M	Wilson	TS2	HFSHR	Naderi (2014)
4	4	0,03	0,04	0,23	1,47
	5	0,06	0,07	0,41	3,79
	6	0,05	0,06	0,55	7,01
5	4	0,04	0,04	0,44	4,70
	5	0,04	0,05	0,56	7,93
	6	0,11	0,11	2,29	82,08
7	4	0,07	0,07	1,20	21,44
	5	0,39	0,39	16,49	366,84
	6	0,42	0,46	23,34	346,94
10	4	2,29	2,62	500,14	500,21
	5	4,64	3,59	500,24	500,18
	6	5,18	5,57	500,19	500,20
15	4	48,33	53,53	500,24	500,38
	5	119,59	129,86	500,19	500,38
	6	147,56	168,82	500,35	500,38
20	4	240,49	228,98	500,51	500,78
	5	497,75	497,48	500,68	501,18
	6	505,94	501,83	501,33	502,74
Promedio		87,39	88,53	252,74	297,14
Desviación		164,95	164,31	254,93	233,82

Tabla 4-2 Análisis tiempos de computación. ***[Fuente: Elaboración propia]***

A continuación, se presenta en la ilustración 4.2, los resultados de los promedios y deviaciones típicas para los diferentes modelos sobre un diagrama de barras para poder comprarlos de manera más gráfica.

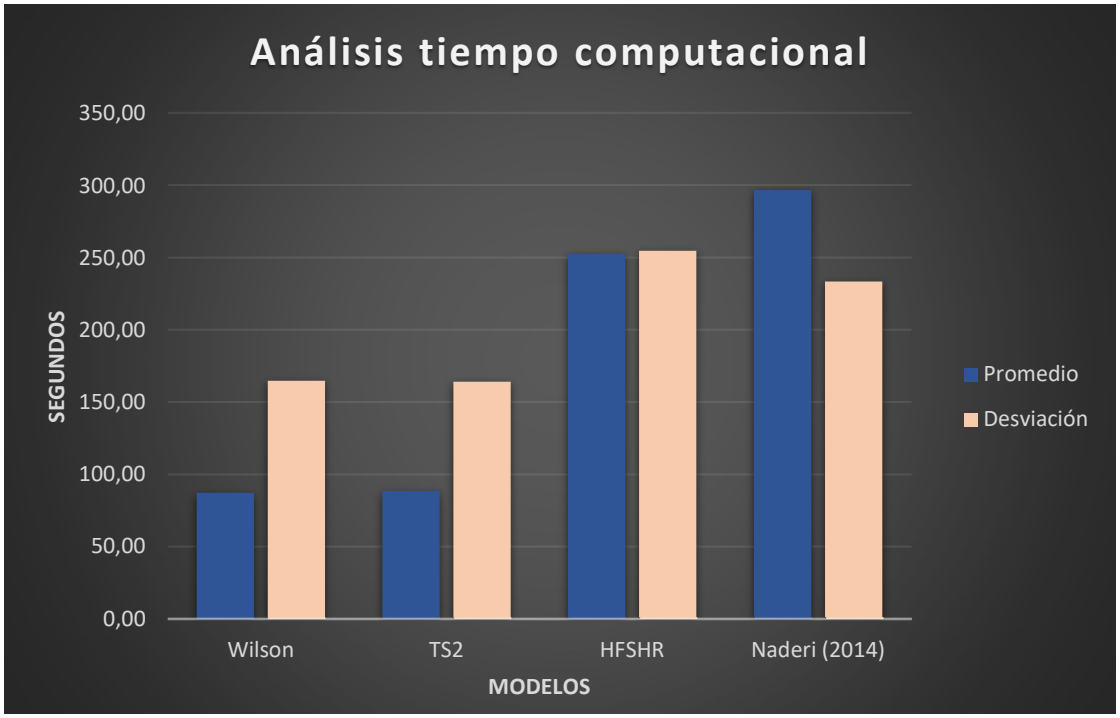

Ilustración 4-2 Gráfico análisis tiempo computacional. ***[Fuente: Elaboración propia]***

Como se podía prever, los modelos HFSHR y Naderi (2014) han tardado más tiempo computacional en resolver las diferentes instancias. Cabe destacar el rápido paso de estos dos modelos de tiempos menores a 100 segundos a llegar al tiempo límite de los 500 segundos lo que provoca que tengan además una gran desviación. Por otro lado, los modelos Wilson y TS2 han tardado menos y con un aumento progresivo en relación con el aumento en el tamaño de la instancia.

4.2.3 Análisis del indicador ARPD.

En la tabla 4.3 se pueden visualizar los promedios de los valores que ha tomado el indicador ARPD en los diferentes modelos e instancias. Para aquellos casos en los que el modelo no haya encontrado una solución factible, el ARPD se calculará solo con las soluciones factibles para esa instancia y en caso de no haber encontrado ninguna se

representará como E (*error)*. Posteriormente, en la figura 4.3 se puede visualizar el valor promedio total de este indicador para todas las instancias.

N	M	Wilson	TS2	HFSHR	Naderi (2014)
4	4	0,00	0,00	0,00	0,00
	5	0,00	0,00	0,00	0,00
	6	0,00	0,00	0,00	0,00
5	4	0,00	0,00	0,00	0,00
	5	0,00	0,00	0,00	0,00
	6	0,00	0,00	0,00	0,00
7	4	0,00	0,00	0,00	0,00
	5	0,00	0,00	0,00	23,78
	6	0,00	0,00	0,00	10,70*
10	4	0,00	0,00	4,32	21,75*
	5	0,00	0,00	11,23	E
	6	0,00	0,00	9,65	E
15	4	0,00	0,00	15,37	E
	5	0,00	0,00	22,17	E
	6	0,00	0,00	21,30	E
20	4	0,00	0,06	22,52	E
	5	1,15	1,02	37,07	E
	6	1,32	2,67	61,50	E
Promedio %		0,14	0,21	11,40	5,62**

Tabla 4-3 Análisis del indicador ARPD. ***[Fuente: Elaboración propia]***

* Valores calculados con las soluciones factibles alcanzadas al haber muchos casos en los que no se ha llegado a una solución factible.

** Valor calculado con el promedio de las instancias en las que se ha llegado a alguna solución factible dentro del tiempo límite.

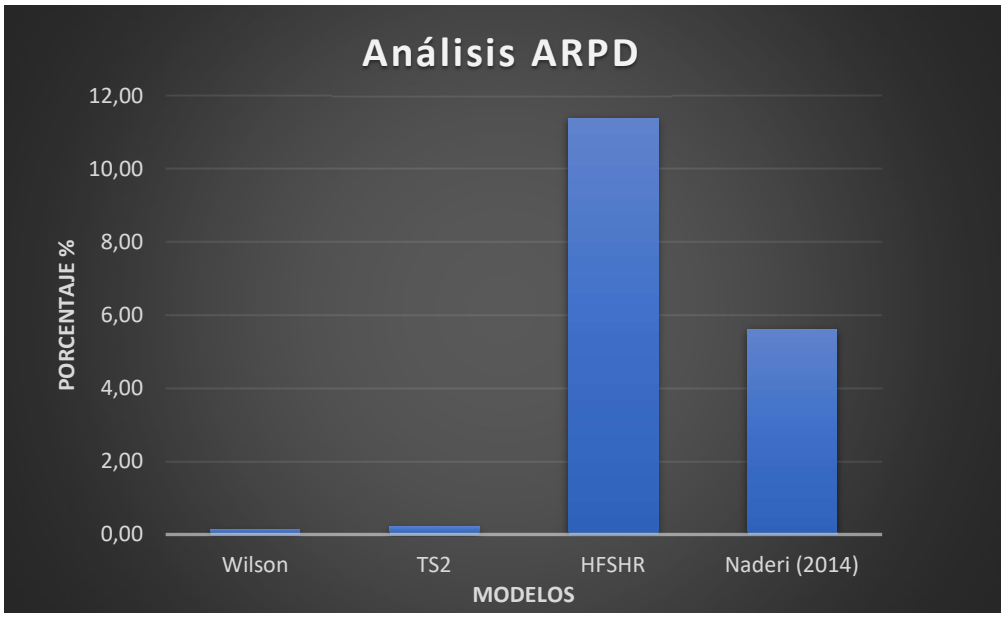

Ilustración 4-3 Gráfica análisis ARPD. ***[Fuente: Elaboración propia]***

En primer lugar, se puede apreciar que los modelos Wilson y TS2 presentan una variación mínima respecto a las soluciones óptimas encontradas mientras que los promedios de los modelos HFSHR adaptado y Naderi (2014) adaptado son más alto. Hay que destacar que, pese a que la adaptación del modelo Naderi (2014) presenta un promedio de solo un 5,62% este dato no es del todo real debido a que hay un gran número de instancias para las que no ha encontrado una solución factible por lo que, si se aumentase el tiempo límite, este valor seguiría creciendo conforme el sistema empiece a encontrar soluciones factibles para dichas instancias. Por último, haciendo referencia a lo mencionando anteriormente, se puede apreciar que en los casos en los que el modelo de Naderi encontraba soluciones factibles para todas las instancias (antes de problemas con siete trabajos y seis máquinas), el modelo HFSHR ha presentado un menor o igual ARPD.

Capítulo 5 Conclusiones

En este proyecto se ha realizado diseñado cuatro modelos de programación lineal entera partiendo de modelos ya existentes en la literatura de la programación de la producción sobre entornos *flowshop* con el fin de ser capaz de resolver aquellas situaciones en las que haya recursos humanos (servers) que realicen los *setups* entre los diferentes trabajos.

El estudio tiene como finalidad principal la búsqueda de la filosofía más eficiente para la resolución de las instancias con servers. Por ello se ha trabajado con tres filosofías diferentes para las restricciones de los servers que se podrían resumir en:

- Adaptación de la metodología planteada por Samarghandi (2015) en su artículo. Utilizado para modelos Wilson adaptado y TS2 adaptado.
- Asignación de *setups* a servers y emplear un variable de precedencia entre *setups.* Utilizado en modelo HFSHR adaptado.
- Asignación de *setups* a servers en una variable que marca la precedencia inmediata entre ellos. Utilizado en modelo Naderi (2014) "Modelo 1" adaptado.

Tras haber diseñado estos modelos, se procede a la resolución de 720 instancias diferentes con cada uno de los modelos para hacer un análisis del comportamiento de cada uno de ellos.

Los resultados presentados en el capítulo 4 son bastantes reveladores en cuanto al comportamiento de los modelos. Los modelos Wilson y TS2 presentan unos resultados más favorables en los tres indicadores empleados: mayor número de soluciones óptimas, menor tiempo de computación y menor ARPD. Esto se debe al menor número de restricciones y de variables que se emplean para modelar la instancia.

Por otro lado, los modelos HFSHR y Naderi (2014) han demostrado ser menos eficientes a la hora de resolver las diferentes instancias. Estos dos modelos presentan una gran cantidad de variables con las que se modelan las restricciones de los servers ($F_{i,j,h,k}$ en el caso del modelo HFSHR adaptado y $Y_{s,i,j,h,k}$ en el caso del de Naderi et al.,2014 "Modelo 1" adaptado) y sus respectivas restricciones asociadas lo que provoca un aumento en el tiempo de computación tal y como se ve reflejado en los resultados del capítulo 4.

Si bien es cierto que tanto el modelo Wilson adaptado como el TS2 adaptado presentan los mejores resultados, se puede apreciar un ligero mejor comportamiento del modelo Wilson tardando un segundo menos de media en encontrar la solución óptima. Esta pequeña diferencia entre ambos se debe al número de variables que emplea cada uno trabajando Wilson solamente con la variable de tiempo de inicio de procesamiento tras *setup* mientras que el modelo de TS2 adaptado utiliza tanto esas como el tiempo de finalización de la operación. Se puede afirmar con estas leves diferencias que estos dos modelos presentan un comportamiento prácticamente idéntico.

Este estudio aporta como resultados, evidencias que afirman que la filosofía más eficiente en cuanto al modelado de la restricción de recursos humanos en el entorno *flowshop* es la planteada en los modelos Wilson y TS2 adaptados. Este resultado no implica que el comportamiento de los modelos originales sea menos eficiente que los modelos originales planteados en los artículos de Costa *et al.* (2020) y Naderi *et al*. (2014) ya que para cada uno de ellos se han empleado diferentes filosofías como ha sido mencionado al inicio de este capítulo por lo que una posible línea de estudio, sería el modelado y posterior análisis de estos cuatro modelos empleando una filosofía adaptada a la planteada para los modelos Wilson y TS2 adaptados.

Por otro lado, otra posible línea de estudio podría ser analizar estos modelos añadiendo una restricción para que la tarea que desempeñen los servers esté compensadas a fin de que el tiempo que estén ocupados los servers sea similar y facilitar, así, la implantación en una industria con estas características en cuanto a su entorno productivo.

Por último, es necesario constatar que actualmente no hay muchos estudios sobre este entorno productivo por lo que es un campo bastante abierto que permite una mayor profundización en cuanto a la resolución exacta y con posibles futuras resoluciones aproximadas para instancias más extensas.

Bibliografía

Brucker, P., Knust, S., & Wang, G. (2005). Complexity results for flow-shop problems with a single server. *European Journal of Operational Research, 165*(2), 398-407. doi:10.1016/j.ejor.2004.04.010.

Costa, A., Fernandez-Viagas, V., & Framinan, J. M. (2020). Solving the hybrid flow shop scheduling problem with limited human resource constraint. *Computers and Industrial Engineering, 146* doi:10.1016/j.cie.2020.106545

Fernandez-Viagas, V., Costa, A., & Framinan, J. M. (2020). Hybrid flow shop with multiple servers: A computational evaluation and efficient divide-and-conquer heuristics. *Expert Systems with Applications, 153* doi:10.1016/j.eswa.2020.113462

Framinan, J. M., Leisten, R., & García, R. R. (2014). *Manufacturing scheduling systems: An integrated view on models, methods and tools* (pp. 1-400) doi:10.1007/978-1-4471-6272-8

Gupta, J. N. D., & Darrow, W. P. (1986). The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research, 24*(3), 439-446. doi:10.1016/0377-2217(86)90037-8

Naderi, B., Gohari, S., & Yazdani, M. (2014). Hybrid flexible flowshop problems: Models and solution methods. *Applied Mathematical Modelling, 38*(24), 5767-5780. doi:10.1016/j.apm.2014.04.012

Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems, fifth edition* (pp. 1-670) doi:10.1007/978-3-319-26580-3

Samarghandi, H. (2015). Studying the effect of server side-constraints on the makespan of the no-wait flow-shop problem with sequence-dependent set-up times. *International Journal of Production Research, 53*(9), 2652-2673. doi:10.1080/00207543.2014.974846

Stafford Jr., E. F., Tseng, F. T., & Gupta, J. N. D. (2005). Comparative evaluation of MILP flowshop models. *Journal of the Operational Research Society, 56*(1), 88-101. doi:10.1057/palgrave.jors.2601805

Tseng, F. T., & Stafford Jr., E. F. (2008). New MILP models for the permutation flowshop problem. *Journal of the Operational Research Society, 59*(10), 1373-1386. doi:10.1057/palgrave.jors.2602455

Werner, F., & Kravchenko, S. A. (2010). Scheduling with multiple servers. *Automation and Remote Control, 71*(10), 2109-2121. doi:10.1134/S0005117910100103

Anexo

Declaración de variables y generación aleatoria de datos

```
public int trabajos, operations;
        public int posiciones, positions;
        public int maquinas, machines;
        public int servers, workers;
        public int[,] tp;
        public int[,,] set;
        public int i, h, j, k, l, p, s;
        public double V = 9999;

        public LecturaDatos(int maq, int trab, int serv)
        {
            maquinas = maq;
            servers = serv;
            trabajos = trab;
            posiciones = trabajos;
            machines = maquinas + 1;
            operations = trabajos + 1;
            workers = servers + 1;
            positions = posiciones + 1;
            tp = new int[machines, operations];
            set = new int[machines, operations, operations];
        }
        //Generación aleatoria de tiempos de proceso y de setup
        public void GeneracionAleatoria(Random rd, int range)
        {

             int i, j, k;
             for (i = 1; i <= maquinas; i++)
             {
                 for (j = 1; j <= trabajos; j++)
                 {
                     tp[i, j] = rd.Next(1,100);

                 }
             }
             for (i = 1; i <= maquinas; i++)
             {
                 for (j = 0; j <= trabajos; j++)
                 {
                     for (k = 1; k <= trabajos; k++)
                     {
                         if (j != k)
                         {
                             set[i, j, k] = rd.Next(1, range);

                         }
                     }
                 }
             }

        }
        //Fin generación aleatoria
```

Main y creación de ficheros

```
        static void Main(string[] args)
         {
             Stopwatch reloj = new Stopwatch();
             //Creacion fichero
             System.IO.StreamWriter FicheroCSV1;
             creacionFicherosCSV_ARPD_nuevo_sinN(out FicheroCSV1);
             FicheroCSV1.Close();
             //Parametros de entrada
             int[] m = new int[] { 4,5,6 };
             int[] n = new int[] { 4, 5, 7, 10, 15, 20 };
             int[] server = new int[] {2,3};
             int[] rangesetup = new int[] {10, 50, 100, 125 };
             int numeroIteraciones = 5;
             int problema = 0;
             for (int a = 0; a < n.Length; a++)
             {
                 for (int b = 0; b < m.Length; b++)
                 {
                     for (int c = 0; c < server.Length; c++)
                     {
                         for (int d = 0; d < rangesetup.Length; d++)
                         {
                             for (int it = 0; it < numeroIteraciones; it++)
                             {
                                 Random rd = new Random(problema);
                                 problema++;
                                 creacionFicherosCSV_ARPD_sinN(out FicheroCSV1);
                                 LecturaDatos input = new LecturaDatos(m[b], n[a],
server[c]);
                                 input.GeneracionAleatoria(rd, rangesetup[d]);
                                 int optimo, otro, infactible, factible; double
tiempoMilisegundos, tiempo, FO1, FO3 , FO2, FO4 , tiempolimite = 500;

                                 //Llamada a modelo 1
                                 reloj.Restart();
                                 input.ModeloWilson(out optimo, out otro, out
infactible, out factible, out FO1, tiempolimite);
                                 tiempoMilisegundos = reloj.ElapsedMilliseconds;
                                 tiempo = tiempoMilisegundos / 1000;
                                 FicheroCSV1.Write(n[a] + ";" + m[b] + ";" + server[c]
+ ";" + rangesetup[d] + ";" + it + ";" + problema + ";");
                                 FicheroCSV1.Write("");

                                 //Escritura solución modelo 1

                                 FicheroCSV1.Write(";" + FO1);
                                 FicheroCSV1.Write(";" + tiempo);
                                 FicheroCSV1.Write(";" + optimo + ";" + factible + ";"
+ otro + ";" + infactible);
                                 FicheroCSV1.Write(";");

                                 //Llamada a modelo 2
                                 reloj.Restart();
                                 input.ModeloTS2(out optimo, out otro, out infactible,
out factible, out FO2, tiempolimite);
                                 tiempoMilisegundos = reloj.ElapsedMilliseconds;
                                 tiempo = tiempoMilisegundos / 1000;
```

```
                                //Escritura solución modelo 2

                                FicheroCSV1.Write(";" + FO2);
                                FicheroCSV1.Write(";" + tiempo);
                                FicheroCSV1.Write(";" + optimo + ";" + factible + ";"
+ otro + ";" + infactible);
                                FicheroCSV1.Write(";");

                                //Llamada a modelo 3
                                reloj.Restart();
                                input.ModeloHFSHR(out optimo, out otro, out
infactible, out factible, out FO3, tiempolimite);
                                tiempoMilisegundos = reloj.ElapsedMilliseconds;
                                tiempo = tiempoMilisegundos / 1000;

                                //Escritura solución modelo 3

                                FicheroCSV1.Write(";" + FO3);
                                FicheroCSV1.Write(";" + tiempo);
                                FicheroCSV1.Write(";" + optimo + ";" + factible + ";"
+ otro + ";" + infactible);
                                FicheroCSV1.Write(";");

                                //Llamada a modelo 4
                                reloj.Restart();
                                input.ModeloNaderi(out optimo, out otro, out
infactible, out factible, out FO4, tiempolimite);
                                tiempoMilisegundos = reloj.ElapsedMilliseconds;
                                tiempo = tiempoMilisegundos / 1000;

                                //Escritura solución modelo 4

                                FicheroCSV1.Write(";" + FO4);
                                FicheroCSV1.Write(";" + tiempo);
                                FicheroCSV1.Write(";" + optimo + ";" + factible + ";"
+ otro + ";" + infactible);
                                FicheroCSV1.Write(";");

                                FicheroCSV1.Write("\n");
                                FicheroCSV1.Close();
                            }
                        }
                    }
                }
            }
            Console.WriteLine("FIN");
        }

        private static void creacionFicherosCSV_ARPD_sinN(out System.IO.StreamWriter
FicheroCSV1)
        {
            String ruta = AppDomain.CurrentDomain.BaseDirectory;
            ruta = ruta.Replace("\\", "/").ToString();
            String ficCSV = ruta + "temp" + "/resultadosCSV1.csv";
            FicheroCSV1 = new System.IO.StreamWriter(ficCSV, true);
        }
        private static void creacionFicherosCSV_ARPD_nuevo_sinN(out
System.IO.StreamWriter FicheroCSV1)
        {
            String ruta = AppDomain.CurrentDomain.BaseDirectory;
            ruta = ruta.Replace("\\", "/").ToString();
            DirectoryInfo DIRESC = new DirectoryInfo(ruta + "/temp");
```

```
        if (!DIRESC.Exists)
        {
            DIRESC.Create();
        }
        String ficBatBorrar = ruta + "temp" + "/resultadosCSV1.csv";
        FileInfo FicheroEliminar = new FileInfo(ficBatBorrar);
        if (FicheroEliminar.Exists)
        {
            File.Delete(ficBatBorrar);
        }
        String ficCSV = ruta + "temp" + "/resultadosCSV1.csv";
        FicheroCSV1 = new System.IO.StreamWriter(ficCSV, true);
        FicheroCSV1.Write("\n\n");

    }
```

Modelo Wilson adaptado

```
//Modelo Wilson
        public void ModeloWilson(out int optimo, out int otro, out int infactible,
out int factible,  out double FO1, double tiempolimite)
        {
            Console.WriteLine("Modelo Wilson Adaptado");
            GRBEnv env = new GRBEnv("ModeloWilson.log");
            GRBModel model = new GRBModel(env);
            //Declaración de variables
            GRBVar[,] B = new GRBVar[machines, positions];

            for (i = 1; i <= maquinas; i++)
            {
                for (p = 1; p <= posiciones; p++)
                {
                    B[i, p] = model.AddVar(0.0, GRB.INFINITY, 0.0, GRB.CONTINUOUS,
"B[" + i + "," + p + "]");
                }
            }
            GRBVar[,] Z = new GRBVar[operations, positions];
            for (j = 1; j <= trabajos; j++)
            {
                for (p = 1; p <= posiciones; p++)
                {
                    Z[j, p] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "Z[" + j + ","
+ p + "]");
                }
            }
            GRBVar[,,] alpha = new GRBVar[operations, positions, operations];
            for (j = 0; j <= trabajos; j++)
            {
                for (p = 0; p <= posiciones; p++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            alpha[j, p, k] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"alpha[" + j + "," + p + "," + k + "]");
                        }
                    }
                }
            }

            GRBVar[,,] gamma = new GRBVar[workers, machines, positions];
            for (s = 1; s <= servers; s++)
            {
                for (i = 1; i <= maquinas; i++)
                {
                    for (p = 1; p <= posiciones; p++)
                    {
                        gamma[s, i, p] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"gamma[" + s + "," + i + "," + p + "]");
                    }
```

```
            }
        }
        _ = new GRBVar();
        GRBVar makespan1 = model.AddVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
"Cmax");
        model.Update();

        //RESTRICCIONES

// Nota: Los números de las restricciones se corresponden a los que aparecen en el
documento del TFG.

        //Restricción 10
        for (j = 1; j <= trabajos; j++)
        {
            GRBLinExpr sum = 0.0;
            for (p = 1; p <= posiciones; p++)
            {
                sum.AddTerm(1.0, Z[j, p]);
            }
            model.AddConstr(sum == 1, "c10");
        }
        //Restricción 11

        for (p = 1; p <= posiciones; p++)
        {
            GRBLinExpr sum = 0.0;
            for (j = 1; j <= trabajos; j++)
            {
                sum.AddTerm(1.0, Z[j, p]);
            }
            model.AddConstr(sum == 1, "c11");
        }
        //Restricción 12
        for (j = 1; j <= trabajos; j++)
        {
            for (p = 1; p <= (posiciones - 1); p++)
            {
                GRBLinExpr sum = 0.0;
                for (k = 1; k <= trabajos; k++)
                {
                    if (j != k)
                    {
                        sum.AddTerm(1.0, alpha[j, p, k]);
                    }
                }
                model.AddConstr(sum == Z[j, p], "c12");
            }
        }
        //Restricción 13
        for (j = 1; j <= trabajos; j++)
        {
            for (p = 2; p <= posiciones; p++)
            {
                GRBLinExpr sum = 0.0;
                for (k = 1; k <= trabajos; k++)
                {
                    if (j != k)
                    {
                        sum.AddTerm(1.0, alpha[k, p - 1, j]);
                    }
                }
```

```
            model.AddConstr(sum == Z[j, p], "c13");
        }
    }
    //Restricción 13prima
    for (j = 1; j <= trabajos; j++)
    {
        model.AddConstr(alpha[0, 0, j] == Z[j, 1], "c13prima");
    }

    //Restricción 14
    for (p = 1; p <= (posiciones - 1); p++)
    {
        GRBLinExpr sum1 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            for (k = 1; k <= trabajos; k++)
            {
                if (j != k)
                {
                    sum1.AddTerm(set[1, j, k], alpha[j, p, k]);
                }
            }
        }
        GRBLinExpr sum2 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sum2.AddTerm(tp[1, j], Z[j, p]);
        }
        model.AddConstr(B[1, p] + sum1 + sum2 - B[1, p + 1] <= 0, "c14");
    }
    //Restricción 15
    for (i = 1; i <= (maquinas - 1); i++)
    {
        GRBLinExpr sum1 = 0.0;
        for (k = 1; k <= trabajos; k++)
        {
            sum1.AddTerm(set[i + 1, 0, k], alpha[0, 0, k]);
        }
        GRBLinExpr sum2 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sum2.AddTerm(tp[i, j], Z[j, 1]);
        }
        model.AddConstr(B[i, 1] + sum1 + sum2 - B[i + 1, 1] <= 0, "c15");
    }

    //Restricción 16
    GRBLinExpr summ = 0.0;
    for (k = 1; k <= trabajos; k++)
    {
        summ.AddTerm(set[1, 0, k], alpha[0, 0, k]);
    }
    model.AddConstr(B[1, 1] - summ == 0, "c16");

    //Restricción 17
    for (i = 1; i <= (maquinas - 1); i++)
    {
        for (p = 2; p <= posiciones; p++)
        {
            GRBLinExpr sum1 = 0.0;
            for (j = 1; j <= trabajos; j++)
            {
```

```
            for (k = 1; k <= trabajos; k++)
            {
                if (j != k)
                {
                    sum1.AddTerm(set[i + 1, j, k], alpha[j, p - 1, k]);
                }
            }
        }
        GRBLinExpr sum2 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sum2.AddTerm(tp[i, j], Z[j, p]);
        }
        model.AddConstr(B[i, p] + sum1 + sum2 - B[i + 1, p] <= 0, "c17");
    }
}

//Restricción 18
for (i = 2; i <= maquinas; i++)
{
    for (p = 1; p <= (posiciones - 1); p++)
    {
        GRBLinExpr sum1 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            for (k = 1; k <= trabajos; k++)
            {
                if (j != k)
                {
                    sum1.AddTerm(set[i, j, k], alpha[j, p, k]);
                }
            }
        }
        GRBLinExpr sum2 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sum2.AddTerm(tp[i, j], Z[j, p]);
        }
        model.AddConstr(B[i, p] + sum1 + sum2 - B[i, p + 1] <= 0, "c18");
    }
}
//Restricción 19 Servers
for (i = 1; i <= maquinas; i++)
{
    for (p = 1; p <= posiciones; p++)
    {
        GRBLinExpr sum = 0.0;
        for (s = 1; s <= servers; s++)
        {
            sum.AddTerm(1.0, gamma[s, i, p]);
        }
        model.AddConstr(sum == 1, "c19");
    }
}

//Restricción 20 Servers adaptación a variables del modelo: l=p';h=i'
for (s = 1; s <= servers; s++)
{
    for (h = 2; h <= maquinas; h++)
    {
        for (i = 1; i <h; i++)
        {
```

```
                        for (l = 2; l <= posiciones; l++)
                        {
                            for (p = 1; p < l; p++)
                            {
                                GRBLinExpr sum1 = 0.0;
                                for (j = 0; j <= trabajos; j++)
                                {
                                    for (k = 1; k <= trabajos; k++)
                                    {
                                        if (j != k)
                                        {
                                            sum1.AddTerm(set[h, j, k], alpha[j, p -
1, k]);
                                        }
                                    }
                                }
                                model.AddConstr(B[h, p] - B[i, l] - sum1 + V * (2 -
gamma[s, h, p] - gamma[s, i, l]) >= 0, "c11");
                            }
                        }
                    }
                }
            }


            //Función objetivo
            GRBLinExpr sumcmax = 0.0;
            for (j = 1; j <= trabajos; j++)
            {
                sumcmax.AddTerm(tp[maquinas, j], Z[j, posiciones]);
            }
            model.AddConstr(sumcmax + B[maquinas, posiciones] - makespan1 == 0,
"cmax");

            model.Write("modeloWilson.lp");
            model.Set(GRB.DoubleParam.TimeLimit, tiempolimite);
            model.Optimize();

            //Devolución de Gurobi
            optimo = 0;
            otro = 0;
            infactible = 0;
            factible = 0;
            FO1 = -1;

            int modelstatus = model.Get(GRB.IntAttr.Status);
            if (modelstatus == GRB.Status.OPTIMAL)
            {
                optimo = 1;
                otro = 0;
                infactible = 0;
                FO1 = model.Get(GRB.DoubleAttr.ObjVal);
                factible = 0;
            }
            else if (modelstatus == GRB.Status.INFEASIBLE)
            {
                optimo = 0;
                otro = 0;
                infactible = 1;
                factible = 0;
            }
            else if (modelstatus == 9)
```

```
    {
        optimo = 0;
        otro = 0;
        infactible = 0;
        factible = 1;
        FO1 = model.Get(GRB.DoubleAttr.ObjVal);
    }
    else
    {
        optimo = 0;
        otro = 1;
        infactible = 0;
        factible = 0;
    }
}
//Fin modelo Wilson
```

Modelo TS2 adaptado

```
//Modelo TS2
        public void ModeloTS2(out int optimo, out int otro, out int infactible, out
int factible, out double FO2, double tiempolimite)
        {
            Console.WriteLine("Modelo TS2 Adaptado");
            GRBEnv env = new GRBEnv("ModeloTS2.log");
            GRBModel model = new GRBModel(env);
            //Declaración de variables
            GRBVar[,] E = new GRBVar[machines, positions];
            for (i = 1; i <= maquinas; i++)
            {
                for (p = 1; p <= posiciones; p++)
                {
                    E[i, p] = model.AddVar(0.0, GRB.INFINITY, 0.0, GRB.CONTINUOUS,
"E[" + i + "," + p + "]");
                }
            }

            GRBVar[,] Z = new GRBVar[operations, positions];
            for (j = 1; j <= trabajos; j++)
            {
                for (p = 1; p <= posiciones; p++)
                {
                    Z[j, p] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "Z[" + j + ","
+ p + "]");
                }
            }
            GRBVar[,,] alpha = new GRBVar[operations, positions, operations];
            for (j = 0; j <= trabajos; j++)
            {
                for (p = 0; p <= posiciones; p++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            alpha[j, p, k] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"alpha[" + j + "," + p + "," + k + "]");
                        }
                    }
                }
            }

            GRBVar[,,] gamma = new GRBVar[workers, machines, positions];
            for (s = 1; s <= servers; s++)
            {
                for (i = 1; i <= maquinas; i++)
                {
                    for (p = 1; p <= posiciones; p++)
                    {
                        gamma[s, i, p] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY,
"gamma[" + s + "," + i + "," + p + "]");
                    }
                }
            }
            _ = new GRBVar();
```

```
            GRBVar makespan2 = model.AddVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
"Cmax");
            model.Update();
            //RESTRICCIONES
            // Nota: Los números de las restricciones se corresponden a los que
aparecen en el documento del TFG.
            //Restricción 8
            for (j = 1; j <= trabajos; j++)
            {
                GRBLinExpr sum = 0.0;
                for (p = 1; p <= posiciones; p++)
                {
                    sum.AddTerm(1.0, Z[j, p]);
                }
                model.AddConstr(sum == 1, "c8");
            }
            //Restricción 9
            for (p = 1; p <= posiciones; p++)
            {
                GRBLinExpr sum = 0.0;
                for (j = 1; j <= trabajos; j++)
                {
                    sum.AddTerm(1.0, Z[j, p]);
                }
                model.AddConstr(sum == 1, "c9");
            }
            //Restricción 10
            for (j = 1; j <= trabajos; j++)
            {
                for (p = 1; p <= (posiciones - 1); p++)
                {
                    GRBLinExpr sum = 0.0;
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(1.0, alpha[j, p, k]);
                        }
                    }
                    model.AddConstr(sum == Z[j, p], "c10");
                }
            }

//Restricción 11
            for (j = 1; j <= trabajos; j++)
            {
                for (p = 2; p <= posiciones; p++)
                {
                    GRBLinExpr sum = 0.0;
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(1.0, alpha[k, p - 1, j]);
                        }
                    }
                    model.AddConstr(sum == Z[j, p], "c11");
                }
            }
            //Restricción 11prima
            for (k = 1; k <= trabajos; k++)
            {
```

```
    model.AddConstr(alpha[0, 0, k] == Z[k, 1], "c11prima");
}
//Restricción 12
for (i = 1; i <= maquinas; i++)
{
    for (p = 1; p <= (posiciones - 1); p++)
    {
        GRBLinExpr sum1 = 0.0;

        for (j = 1; j <= trabajos; j++)
        {
            for (k = 1; k <= trabajos; k++)
            {
                if (j != k)
                {
                    sum1.AddTerm(set[i, j, k], alpha[j, p, k]);
                }
            }
        }
        GRBLinExpr sum2 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sum2.AddTerm(tp[i, j], Z[j, p + 1]);
        }
        model.AddConstr(E[i, p] + sum1 + sum2 - E[i, p + 1] <= 0, "c12");
    }
}
//Restricción 13
for (i = 1; i <= maquinas - 1; i++)
{
    for (p = 2; p <= posiciones; p++)
    {
        GRBLinExpr sum1 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            for (k = 1; k <= trabajos; k++)
            {
                if (j != k)
                {
                    sum1.AddTerm(set[i + 1, j, k], alpha[j, p - 1, k]);
                }
            }
        }
        GRBLinExpr sum2 = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sum2.AddTerm(tp[i + 1, j], Z[j, p]);
        }
        model.AddConstr(E[i, p] + sum1 + sum2 - E[i + 1, p] <= 0, "c13");
    }
}
//Restricción 14
for (i = 1; i <= maquinas - 1; i++)
{
    GRBLinExpr sum1 = 0.0;
    for (k = 1; k <= trabajos; k++)
    {
        sum1.AddTerm(set[i + 1, 0, k], alpha[0, 0, k]);
    }
    GRBLinExpr sum2 = 0.0;
    for (j = 1; j <= trabajos; j++)
    {
```

```
                sum2.AddTerm(tp[i + 1, j], Z[j, 1]);
            }
            model.AddConstr(E[i, 1] + sum1 + sum2 - E[i + 1, 1] <= 0, "c14");
        }
        //Restricción 15
        GRBLinExpr suma = 0.0;
        for (k = 1; k <= trabajos; k++)
        {
            suma.AddTerm(set[1, 0, k], alpha[0, 0, k]);
        }
        GRBLinExpr sumb = 0.0;
        for (j = 1; j <= trabajos; j++)
        {
            sumb.AddTerm(tp[1, j], Z[j, 1]);
        }
        model.AddConstr(E[1, 1] - suma - sumb == 0, "c15");

        //Restricción 16 Servers
        for (i = 1; i <= maquinas; i++)
        {
            for (p = 1; p <= posiciones; p++)
            {
                GRBLinExpr sum = 0.0;
                for (s = 1; s <= servers; s++)
                {
                    sum.AddTerm(1.0, gamma[s, i, p]);
                }
                model.AddConstr(sum == 1, "c16");
            }
        }

        //Restricción 17 Servers adaptación a variables del modelo: l=p';h=i'

        for (s = 1; s <= servers; s++)
        {
            for (h = 2; h <= maquinas; h++)
            {
                for (i = 1; i < h; i++)
                {
                    for (l = 2; l <= posiciones; l++)
                    {
                        for (p = 1; p < l; p++)
                        {
                            GRBLinExpr sum1 = 0.0;
                            for (j = 0; j <= trabajos; j++)
                            {
                                for (k = 1; k <= trabajos; k++)
                                {
                                    if (j != k)
                                    {
                                        sum1.AddTerm(set[h, j, k], alpha[j, p -
1, k]);
                                    }
                                }
                            }

                            GRBLinExpr sum2 = 0.0;
                            for (j = 1; j <= trabajos; j++)
                            {
                                sum2.AddTerm(tp[h, j], Z[j, p]);
                            }
                            GRBLinExpr sum3 = 0.0;
```

```
                                for (j = 1; j <= trabajos; j++)
                                {
                                    sum3.AddTerm(tp[i, j], Z[j, l]);
                                }


                                model.AddConstr(E[h, p]-sum2 -( E[i, l]-sum3 ) - sum1
+ V * (2 - gamma[s, h, p] - gamma[s, i, l]) >= 0, "c17");
                                }
                            }
                        }
                    }
                }
                //Función objetivo
                model.AddConstr(makespan2 - E[maquinas, posiciones] == 0, "cmax");

                model.Write("modeloTS2.lp");
                model.Set(GRB.DoubleParam.TimeLimit, tiempolimite);
                model.Optimize();

                 //Devolución de Gurobi
                optimo = 0;
                otro = 0;
                infactible = 0;
                factible = 0;
                FO2 = -1;

                int modelstatus = model.Get(GRB.IntAttr.Status);
                if (modelstatus == GRB.Status.OPTIMAL)
                {
                    optimo = 1;
                    otro = 0;
                    infactible = 0;
                    FO2 = model.Get(GRB.DoubleAttr.ObjVal);
                    factible = 0;
                }
                else if (modelstatus == GRB.Status.INFEASIBLE)
                {
                    optimo = 0;
                    otro = 0;
                    infactible = 1;
                    factible = 0;
                }
                else if (modelstatus == 9)
                {
                    optimo = 0;
                    otro = 0;
                    infactible = 0;
                    factible = 1;
                    FO2 = model.Get(GRB.DoubleAttr.ObjVal);
                }
                else
                {
                    optimo = 0;
                    otro = 1;
                    infactible = 0;
                    factible = 0;
                }
            }
                //Fin modelo TS2
```

Modelo HFSHR adaptado

```
  // Modelo HFS/HR
            public void ModeloHFSHR(out int optimo, out int otro, out int infactible,
out int factible, out double FO3, double tiempolimite)
            {
                Console.WriteLine("Modelo HFSHR Adaptado");
                GRBEnv env = new GRBEnv("ModeloHFSHR.log");
                GRBModel model = new GRBModel(env);
                //Declaración de variables
                GRBVar[,,] X = new GRBVar[machines, operations, workers];
                for (i = 1; i <= maquinas; i++)
                {
                    for (j = 1; j <= trabajos; j++)
                    {
                        for (s = 1; s <= servers; s++)
                        {
                            X[i, j, s] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "X["
+ i + "," + j + "," + s + "]");
                        }
                    }
                }
                GRBVar[,,,] F = new GRBVar[machines, operations, machines,
operations];
                for (i = 1; i <= maquinas; i++)
                {
                    for (j = 1; j <= trabajos; j++)
                    {
                        for (h = 1; h <= maquinas; h++)
                        {
                            for (k = 1; k <= trabajos; k++)
                            {
                                if (i == h && j == k)
                                { }
                                else
                                {
                                    F[i, j, h, k] = model.AddVar(0.0, 1.0, 0.0,
GRB.BINARY, "F[" + i + "," + j + "," + h + "," + k + "]");
                                }
                            }
                        }
                    }
                }
                GRBVar[,] D = new GRBVar[operations, operations];
                for (j = 0; j <= trabajos; j++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (k != j)
                        {
                            D[j, k] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "D[" +
j + "," + k + "]");
                        }
                    }
                }
                GRBVar[,] C = new GRBVar[machines, operations];
                for (i = 1; i <= maquinas; i++)
                {
```

```
                    for (j = 0; j <= posiciones; j++)
                    {
                        C[i, j] = model.AddVar(0.0, GRB.INFINITY, 0.0,
GRB.CONTINUOUS, "C[" + i + "," + j + "]");
                    }
                }
                GRBVar[,] U = new GRBVar[machines, operations];
                for (i = 1; i <= maquinas; i++)
                {
                    for (j = 0; j <= posiciones; j++)
                    {
                        U[i, j] = model.AddVar(0.0, GRB.INFINITY, 0.0,
GRB.CONTINUOUS, "U[" + i + "," + j + "]");
                    }
                }
                GRBVar makespan3 = new GRBVar();
                makespan3 = model.AddVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
"Cmax");
            model.Update();
            //Fin declaración de variables
            //RESTRICCIONES
            // Nota: Los números de las restricciones se corresponden a los que
aparecen en el documento del TFG.
            //Restricción 9
            for (k = 1; k <= trabajos; k++)
                {
                    GRBLinExpr sum = 0.0;
                    for (j = 0; j <= trabajos; j++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(1.0, D[j, k]);
                        }
                    }
                    model.AddConstr(sum == 1, "c9");
                }

                //Restricción 10
                for (j = 0; j <= trabajos; j++)
                {
                    GRBLinExpr sum = 0.0;
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(1.0, D[j, k]);
                        }
                    }
                    model.AddConstr(sum <= 1, "c10");
                }


                //Restricción 11
                for (j = 1; j <= trabajos; j++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            GRBLinExpr sum = 0.0;
                            for (h = 0; h <= trabajos; h++)
                            {
```

```
                            if (h != j)
                            {
                                sum.AddTerm(1.0, D[h, j]);
                            }
                        }
                        model.AddConstr(sum - D[j, k] >= 0, "c11");
                    }
                }
            }
            //Restricción 12
            for (i = 1; i <= maquinas; i++)
            {
                for (j = 1; j <= trabajos; j++)
                {
                    GRBLinExpr sum = 0.0;
                    for (s = 1; s <= servers; s++)
                    {
                        sum.AddTerm(1.0, X[i, j, s]);
                    }
                    model.AddConstr(sum == 1, "c12");
                }
            }
            //Restricción 13
            for (i = 1; i <= maquinas; i++)
            {
                for (j = 1; j <= trabajos; j++)
                {
                    for (h = 1; h <= maquinas; h++)
                    {
                        for (k = 1; k <= trabajos; k++)
                        {
                            if (i == h && j == k)
                            { }

                            else
                            {
                                for (s = 1; s <= servers; s++)
                                {
                                    model.AddConstr(-X[i, j, s] - X[h, k, s] +
F[i, j, h, k] + F[h, k, i, j] + 1 >= 0, "c13");
                                }
                            }
                        }
                    }
                }
            }
            //Restricción 14
            for (i = 1; i <= maquinas; i++)
            {
                for (j = 0; j <= trabajos; j++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            model.AddConstr(U[i, k] - C[i, j] + V * (1 - D[j, k])
>= 0, "c3");
                        }
                    }
                }
            }
```

```
                //Restricción 15
                for (i = 1; i <= maquinas; i++)
                {
                    for (j = 1; j <= trabajos; j++)
                    {
                        for (h = 1; h <= maquinas; h++)
                        {
                            for (k = 1; k <= trabajos; k++)
                            {
                                if (i == h && j == k)
                                { }
                                else
                                {
                                    GRBLinExpr sum1 = 0.0;
                                    for (l = 0; l <= trabajos; l++)
                                    {
                                        if (j != l)
                                        {
                                            sum1.AddTerm(set[i, l, j], D[l, j]);
                                        }
                                    }
                                    model.AddConstr(U[h, k] - U[i, j] - sum1 + V * (1
- F[i, j, h, k]) >= 0, "c15");
                                }
                            }
                        }
                    }
                }

                //Restricción 16
                for (i = 1; i <= maquinas; i++)
                {
                    for (j = 1; j <= trabajos; j++)
                    {
                        GRBLinExpr sum1 = 0.0;
                        for (k = 0; k <= trabajos; k++)
                        {
                            if (j != k)
                            {
                                sum1.AddTerm(set[i, k, j], D[k, j]);
                            }
                        }
                        model.AddConstr(C[i, j] - U[i, j] - sum1 - tp[i, j] >= 0,
"c16");
                    }
                }
                //Restricción 17
                for (i = 2; i <= maquinas; i++)
                {
                    for (j = 1; j <= trabajos; j++)
                    {
                        model.AddConstr(U[i, j] - C[i - 1, j] >= 0, "c17");
                    }
                }
                //Restricciones 18 y 19
                for (i = 1; i <= maquinas; i++)
                {
                    model.AddConstr(U[i, 0] == 0, "c18");
                    model.AddConstr(C[i, 0] == 0, "c19");
                }
```

```
                //Restricción 20
                for (j = 1; j <= trabajos; j++)
                    {
                        model.AddConstr(makespan3 - C[maquinas, j] >= 0, "c20");
                    }
                model.Write("modeloHFSHR.lp");
                model.Set(GRB.DoubleParam.TimeLimit, tiempolimite);
                model.Optimize();

//Devolución de Gurobi
            optimo = 0;
                otro = 0;
                infactible = 0;
                factible = 0;
                FO3 = -1;

                int modelstatus = model.Get(GRB.IntAttr.Status);
                if (modelstatus == GRB.Status.OPTIMAL)
                {
                    optimo = 1;
                    otro = 0;
                    infactible = 0;
                    FO3 = model.Get(GRB.DoubleAttr.ObjVal);
                    factible = 0;
                }
                else if (modelstatus == GRB.Status.INFEASIBLE)
                {
                    optimo = 0;
                    otro = 0;
                    infactible = 1;
                    factible = 0;
                }
                else if (modelstatus == 9)
                {
                    optimo = 0;
                    otro = 0;
                    infactible = 0;
                    factible = 1;
                    FO3 = model.Get(GRB.DoubleAttr.ObjVal);
                }
                else
                {
                    optimo = 0;
                    otro = 1;
                    infactible = 0;
                    factible = 0;
                }
            }
            //Fin modelo HFSHR
```

Modelo Naderi (2014) adaptado

```
// Modelo Naderi

            public void ModeloNaderi(out int optimo, out int otro, out int
infactible, out int factible,out double FO4, double tiempolimite)
            {
                Console.WriteLine("Modelo Naderi Adaptado");
                GRBEnv env = new GRBEnv("ModeloNaderi.log");
                GRBModel model = new GRBModel(env);
                //Declaración de variables
                GRBVar[,] D = new GRBVar[operations, operations];
                for (j = 0; j <= trabajos; j++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            D[j, k] = model.AddVar(0.0, 1.0, 0.0, GRB.BINARY, "D[" +
j + "," + k + "]");
                        }
                    }
                }
                GRBVar[,] C = new GRBVar[machines, operations];
                for (i = 1; i <= maquinas; i++)
                {
                    for (k = 0; k <= trabajos; k++)
                    {
                        C[i, k] = model.AddVar(0.0, GRB.INFINITY, 0.0,
GRB.CONTINUOUS, "C[" + i + "," + k + "]");
                    }
                }

                GRBVar[,,,,] Y = new GRBVar[workers, machines, operations, machines,
operations];
                for (s = 1; s <= servers; s++)
                {
                    for (i = 1; i <= maquinas; i++)
                    {
                        for (j = 1; j <= trabajos; j++)
                        {
                            for (h = 1; h <= maquinas; h++)
                            {
                                for (k = 1; k <= trabajos; k++)
                                {
                                    if (i == h && j == k)
                                    { }
                                    else
                                    {
                                        Y[s, i, j, h, k] = model.AddVar(0.0, 1.0,
0.0, GRB.BINARY, "Y[" + s + "," + i + "," + j + "," + h + "," + k + "]");
                                    }
                                }
                            }
                        }
                    }
                }
                //Trabajo inicial servers
```

```
                for (s = 1; s <= servers; s++)
                {
                    for (h = 1; h <= maquinas; h++)
                    {
                        for (k = 1; k <= trabajos; k++)
                        {
                            Y[s, 0, 0, h, k] = model.AddVar(0.0, 1.0, 0.0,
GRB.BINARY, "Y[" + s + "," + 0 + "," + 0 + "," + h + "," + k + "]");
                        }
                    }
                }
                GRBVar makespan4 = new GRBVar();
                makespan4 = model.AddVar(0.0, GRB.INFINITY, 1.0, GRB.CONTINUOUS,
"Cmax");
            model.Update();
            //Fin declaración de variables
            //RESTRICCIONES
            // Nota: Los números de las restricciones se corresponden a los que
aparecen en el documento del TFG.
            //Restricción 10

            for (k = 1; k <= trabajos; k++)
                {
                    GRBLinExpr sum = 0.0;
                    for (j = 0; j <= trabajos; j++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(1.0, D[j, k]);
                        }
                    }
                    model.AddConstr(sum == 1, "c10");
                }

                //Restricción 11
                for (j = 0; j <= trabajos; j++)
                {
                    GRBLinExpr sum = 0.0;
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(1.0, D[j, k]);
                        }
                    }
                    model.AddConstr(sum <= 1, "c11");
                }
               //Restricción 12
                for (j = 1; j <= trabajos; j++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            GRBLinExpr sum = 0.0;
                            for (h = 0; h <= trabajos; h++)
                            {
                                if (h != j)
                                {
                                    sum.AddTerm(1.0, D[h, j]);
                                }
                            }
```

```
                        model.AddConstr(sum - D[j, k] >= 0, "12");
                    }
                }
            }

            //Restricción 13
            for (k = 1; k <= trabajos; k++)
            {
                GRBLinExpr sum = 0.0;
                for (j = 0; j <= trabajos; j++)
                {
                    if (j != k)
                    {
                        sum.AddTerm(set[1, j, k], D[j, k]);
                    }
                }
                model.AddConstr(C[1, k] - sum - tp[1, k] >= 0, "c13");
            }
            //Restricción 14
            for (i = 2; i <= maquinas; i++)
            {
                for (k = 1; k <= trabajos; k++)
                {
                    GRBLinExpr sum = 0.0;
                    for (j = 0; j <= trabajos; j++)
                    {
                        if (j != k)
                        {
                            sum.AddTerm(set[i, j, k], D[j, k]);
                        }
                    }
                    model.AddConstr(C[i, k] - C[i - 1, k] - sum - tp[i, k] >= 0,
"c14");
                }
            }

            //Restricción 15
            for (i = 1; i <= maquinas; i++)
            {
                for (j = 0; j <= trabajos; j++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (j != k)
                        {
                            model.AddConstr(C[i, k] - C[i, j] + V * (1 - D[j, k])
- tp[i, k] - set[i, j, k] >= 0, "c15");
                        }

                    }
                }
            }



            //Restricción 16
            for (h = 1; h <= maquinas; h++)
            {
                for (k = 1; k <= trabajos; k++)
                {
                    GRBLinExpr sum = 0.0;
                    GRBLinExpr sum1 = 0.0;
```

```
            for (s = 1; s <= servers; s++)
            {
                for (i = 1; i <= maquinas; i++)
                {
                    for (j = 1; j <= trabajos; j++)
                    {
                        if (i == h && j == k)
                        { }
                        else
                        {
                            sum.AddTerm(1.0, Y[s, i, j, h, k]);
                        }
                    }
                }
                sum1.AddTerm(1.0, Y[s, 0, 0, h, k]);
            }
            model.AddConstr(sum + sum1 == 1, "c16");
        }
    }
    //Restricción 17
    for (i = 1; i <= maquinas; i++)
    {
        for (j = 1; j <= trabajos; j++)
        {
            GRBLinExpr sum = 0.0;
            for (s = 1; s <= servers; s++)
            {
                for (h = 1; h <= maquinas; h++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        if (i == h && j == k)
                        { }
                        else
                        {
                            sum.AddTerm(1.0, Y[s, i, j, h, k]);
                        }
                    }
                }
            }
            model.AddConstr(sum <= 1, "c17");
        }
    }
    //Restricción 18
    for (s = 1; s <= servers; s++)
    {
        GRBLinExpr sum = 0.0;
        for (h = 1; h <= maquinas; h++)
        {
            for (k = 1; k <= trabajos; k++)
            {
                sum.AddTerm(1.0, Y[s, 0, 0, h, k]);
            }
        }
        model.AddConstr(sum == 1, "c18");
    }

    //Restricción 19
    for (s = 1; s <= servers; s++)
    {
        for (i = 1; i <= maquinas; i++)
        {
```

```
                        for (j = 1; j <= trabajos; j++)
                        {
                            GRBLinExpr sum1 = 0.0;
                            GRBLinExpr sum2 = 0.0;
                            int o;
                            for (o = 1; o <= maquinas; o++)
                            {
                                for (l = 1; l <= trabajos; l++)
                                {
                                    if (i == o && j == l)
                                    { }
                                    else
                                    {
                                        sum1.AddTerm(1.0, Y[s, o, l, i, j]);
                                    }
                                }
                            }
                            for (h = 1; h <= maquinas; h++)
                            {
                                for (k = 1; k <= trabajos; k++)
                                {
                                    if (i == h && j == k)
                                    { }
                                    else
                                    {
                                        sum2.AddTerm(1.0, Y[s, i, j, h, k]);
                                    }

                                }
                            }
                            model.AddConstr(sum1 + Y[s, 0, 0, i, j] - sum2 >= 0,
"c19");
                        }
                    }
                }
                //Restricción 20
                for (s = 1; s <= servers; s++)
                {
                    for (i = 1; i <= maquinas; i++)
                    {
                        for (j = 1; j <= trabajos; j++)
                        {
                            for (h = 1; h <= maquinas; h++)
                            {
                                for (k = 1; k <= trabajos; k++)
                                {
                                    if (i == h && j == k)
                                    { }
                                    else
                                    {
                                        GRBLinExpr sum1 = 0.0;
                                        for (l = 0; l <= trabajos; l++)
                                        {
                                            if (l != k)
                                            {
                                                sum1.AddTerm(set[h, l, k], D[l, k]);
                                            }
                                        }
                                        model.AddConstr(C[h, k]-tp[h,k] - (C[i, j]-
tp[i,j]) - sum1 + V * (1 - Y[s, i, j, h, k]) >= 0, "c20");

                                    }
```

```
                            }
                        }
                    }
                }
            }
            //Restricción 21
            for (s = 1; s <= servers; s++)
            {
                for (h = 1; h <= maquinas; h++)
                {
                    for (k = 1; k <= trabajos; k++)
                    {
                        GRBLinExpr sum1 = 0.0;
                        for (l = 0; l <= trabajos; l++)
                        {
                            if (l != k)
                            {
                                sum1.AddTerm(set[h, l, k], D[l, k]);
                            }
                        }
                        model.AddConstr(C[h, k]-tp[h,k] - sum1 + V * (1 - Y[s, 0,
0, h, k]) >= 0, "c21");
                    }
                }
            }

            //Función Objetivo
            for (k = 1; k <= trabajos; k++)
            {
                model.AddConstr(makespan4 - C[maquinas, k] >= 0, "c22");
            }
            model.Write("modeloNaderi.lp");
            model.Set(GRB.DoubleParam.TimeLimit, tiempolimite);
            model.Optimize();

        //Devolución de Gurobi
        optimo = 0;
            otro = 0;
            infactible = 0;
            factible = 0;
         FO4 = -1;

            int modelstatus = model.Get(GRB.IntAttr.Status);
            if (modelstatus == GRB.Status.OPTIMAL)
            {
                optimo = 1;
                otro = 0;
                infactible = 0;
                FO4 = model.Get(GRB.DoubleAttr.ObjVal);
                factible = 0;
            }
            else if (modelstatus == GRB.Status.INFEASIBLE)
            {
                optimo = 0;
                otro = 0;
                infactible = 1;
                factible = 0;
            }
            else if (modelstatus == 9)
            {
                optimo = 0;
                otro = 0;
```

```
            infactible = 0;
            factible = 1;
            FO4 = model.Get(GRB.DoubleAttr.ObjVal);
        }
        else
        {
            optimo = 0;
            otro = 1;
            infactible = 0;
            factible = 0;
        }
    }
```