PROYECTO FIN DE GRADO

INGENIERÍA DE LAS TECNOLOGÍAS INDUSTRIALES

ESTUDIO DE MODELOS DE PROGRAMACIÓN LINEAL ENTERA PARA EL TALLER DE FLUJO REGULAR CON ESPERAS NO PERMITIDAS

Autor: Jose Luis Fernández-Figueroa González

Tutor: Víctor Fernández-Viagas Escudero

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

Sevilla, 2018

Proyecto Fin de Grado
Ingeniería en Tecnologías Industriales

ESTUDIO DE MODELOS DE PROGRAMACIÓN LINEAL ENTERA PARA EL TALLER DE FLUJO REGULAR CON ESPERAS NO PERMITIDAS

Autor:

Jose Luis Fernández-Figueroa González

Tutor:

Víctor Fernández-Viagas Escudero

Profesor ayudante doctor

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

Sevilla, 2018

A mi familia
A mis amigos

Resumen

Este trabajo de Fin de Grado trata de resolver un problema de análisis de modelos de programación lineal en un entorno de un taller de flujo con permutación adaptando varios modelos generales de taller de flujo con permutación a un taller de flujo con esperas no permitidas.

La metodología seguida en este trabajo, es la utilización de un algoritmo, de los llamados algoritmos exactos, como es el método MILP que consiste en establecer una función objetivo sujeta a unas restricciones para resolver los modelos.

Finalmente, se hará una evaluación computacional de varios aspectos a todos los modelos a partir del cual se obtendrá cual es el mejor modelo comparándolos entre ellos.

Índice

Resumen VI

Índice VII

Índice de figuras IX

Índice de tablas X

1. Introducción 1

1.1. Organización de la producción 1

1.2. Programación de la producción 1

1.3. α|β|γ 2

1.4. FlowShop 3

1.5. No-wait (nwt) 4

1.6. Cmax y $\sum C_j$ 4

1.7. Diagramas de Gantt 5

2. Objeto del proyecto 6

2.1. Objeto del proyecto 6

2.2. Justificación 6

3. Modelos de programación lineal de las familias Wagner y Manne para el permutation flowshop clásico 8

3.1. Introducción 8

3.2. Descripción de los modelos y sus variables 8

3.2.1. Familia Wagner 8

3.2.2. Familia Manne 10

3.2.3. Modelo adicionales familia Wagner 11

4. Herramientas y metodologías utilizadas 12

4.1. MILP 12

4.2. CodeBlocks y C 13

4.3. Gurobi 15

4.4. Batch y Registro 15

5. MILPs para el problema bajo consideración 17

5.1. Familia Wagner 17

5.2. Familia Manne 21

5.3. Modelos adicionales familia Wagner 24

6. Evaluación computacional 27

6.1. Datos del problema 27

6.2. Evaluación numérica 28

6.2.1. Evaluación numérica Cmax 30
6.2.2. Evaluación numérica $\sum C_j$ 33
7. Conclusión 36
8. Bibliografía 37
9. Anexos 38
9.1. Familia Wagner 38
9.1.1. Modelo WST 38
9.1.2. Modelo Wilson 43
9.1.3. Modelo TS2 49
9.1.4. Modelo TBA 54
9.1.5. Modelo TS3 60
9.2. Familia Manne 69
9.2.1. Modelo SGST 70
9.2.2. Modelo LYeq 74
9.3. Archivo BATCH 79

Índice de figuras

Figura 1: Ejemplo diagrama de Gantt 5

Figura 2: Ejemplo cadena de frío 7

Figura 3: Modelo general maximizar MILP 12

Figura 4: Ejemplo del formato LP en Gurobi 15

Figura 5: Código del archivo .bat 16

Figura 6: Datos de archivo de ejemplo 27

Índice de tablas

Tabla 1: Resumen variables modelo WST....9

Tabla 2: Resumen variables modelo Wilson....9

Tabla 3: Resumen variables modelo TS2 9

Tabla 4: Resumen variables modelo SGST 10

Tabla 5: Resumen variables modelo LYeq....10

Tabla 6: Resumen variables modelo TBA 11

Tabla 7: Resumen variables modelo TS3 11

Tabla 8: Resumen de los comandos más usados en C 14

Tabla 9: Matriz de tiempos de proceso del ejemplo 28

Tabla 10: Diagramas de Gantt del ejemplo....29

Tabla 11: Análisis de resultados de la función objetivo Cmax....30

Tabla 12: Tiempos de computación en la función objetivo Cmax....31

Tabla 13: Tabla de ARPDs de la función objetivo Cmax....32

Tabla 14: Análisis de resultados de la función objetivo $\sum C_j$ 33

Tabla 15: Tiempos de computación en la función objetivo $\sum C_j$....34

Tabla 16: Tabla de ARPDs de la función objetivo $\sum C_j$ 34

1. Introducción

1.1. Organización de la producción

La organización de la producción, es una rama del conocimiento con cada vez mayor importancia dentro del organigrama de una empresa debido a su gran trascendencia a la hora de organizar una empresa, tanto, en el ámbito productivo como organizativo. Hoy en día, puede llegar a ser un factor diferencial entre las empresas de un mismo sector debido a la gran globalización en la que nos encontramos ahora mismo en la que los productos llegan a ser muy parecidos entre ellos y por ello, las empresas hoy en día basan sus estrategias en mejorar su logística, hecho que forma parte de la organización de la producción. Este concepto, tiene muchísimas aplicaciones en la vida real, tanto para organizar los turnos en un supermercado, como para repartir las tareas en la creación de un nuevo producto o incluso en el proceso de mantener las cadenas de frío de los alimentos, de esto último precisamente se hablará más adelante debido a la naturaleza y el estudio de este trabajo.

1.2. Programación de la producción

Dentro de la organización de la producción, uno de los aspectos más importantes es el de la programación de la producción debido a su complejidad en determinados casos y para mantener un control más cuantitativo y exacto en las asignaciones de las tareas en ciertos procesos o trabajos, algunos de los cuales, pueden ser, por ejemplo, la asignación de los tiempos de llegada y/o despegue de un aeropuerto o las etapas de un proyecto de construcción como bien se explica en el blog Why You Need Intelligent Scheduling Software (2014) publicado en Capterra.

La programación de la producción, es un problema que consiste en asignar los trabajos o tiempos correspondientes a varias máquinas, estaciones o etapas dentro de una fábrica u organización con el fin de minimizar o maximizar al máximo un objetivo determinado. Estos objetivos se establecerán según las necesidades de cada empresa y pueden ser tales como, el tiempo

de terminación de todos los procesos (*makespan*), el tiempo total en el que los trabajos acaban antes de tiempo (*total earliness*), el tiempo máximo que llega tarde un trabajo (*maximum tardiness*) o el tiempo máximo que excede un trabajo su tiempo de terminación (*máximum lateness*), entre otros. Adicionalmente, de ahora en adelante se referirá a las máquinas, estaciones o recursos humanos, como máquinas y a los diferentes procesos que se llevan a cabo en ellas como trabajos.

1.3. α|β|γ

Dentro de la programación de la producción, los problemas tienen una notación específica en la cual se especifican, las características de las máquinas (α), las características de los trabajos (β) y las características de la función objetivo (γ); estas tres características se denotan de la siguiente manera: α|β|γ (Graham, 1979).

Alpha (α) nos indica el modo en el que están dispuestas las máquinas con las que vamos a trabajar en nuestro problema. Puede ser de varios tipos. Por una parte, tenemos una serie de entornos en los que los trabajos tienen únicamente una etapa en su ruta: Pm, para máquinas idénticas dispuestas en paralelo, es decir, que es indiferente en qué máquina se realice el proceso; Qm, para máquinas uniformes dispuestas en paralelo, es decir, que es indiferente en qué máquina se realice el proceso, pero puede variar la velocidad de procesado entre las máquinas; Rm, para máquinas dispuestas en paralelo no relacionadas, es decir, modo en el que los tiempos de proceso de cada trabajo dependen de la máquina a la que sea asignado y además, está la *single machine*, con una sola máquina donde se procesan todos los trabajos. Además, existe otra disposición en la que los trabajos tienen que seguir una ruta específica como es: Jm, *jobshop*, en el que los trabajos tienen rutas predeterminadas diferentes. Otra disposición, es el caso del *openshop*, Om, es el entorno más general y no tienen una ruta predeterminada. El Hm, nos indicaría que estamos en un entorno híbrido, es decir, mezcla de varios modos o disposiciones diferentes. Aunque finalmente, tenemos Fm que se trata de un *flowshop* (se puede escribir junto o separado indistintamente) de *m* máquinas que es el que nos concierne en nuestro problema que también tiene una ruta establecida y que será descrito con mayor profundidad en el siguiente punto.

Beta (β) indica cuales son las restricciones del problema a resolver que pueden ser de diferente índole como, por ejemplo: de acuerdo a un árbol de precedencia (*prec*), no permite tiempos ociosos en las máquinas de trabajos (*no-idle*), fija las fechas de llegada (r_j), fija las fechas de entrega (d_j), fija los tiempos de setup en la máquina *i* para procesar el trabajo *k* después de procesar el trabajo *j* (s_{ijk}) o los trabajos no pueden esperar entre máquinas o estados (*nwt*) que es el que nos concierne en este trabajo, entre otros.

Gamma (γ) indica cual es la función objetivo del problema del cual queremos buscar su solución óptima, dentro de las funciones objetivo se pueden encontrar una gran variedad de opciones dependiendo de las necesidades de cada problema o de lo que se esté buscando como, por ejemplo, el tiempo de finalización del último trabajo (*makespan*) o la suma de los tiempos de finalización de todos los trabajos (SumCj), que son los que se van a tratar en este trabajo.

1.4. FlowShop

Definimos *flowshop*, como un tipo de taller en el que hay *m* máquinas pero que, en lugar de organizarse en paralelo, se organizan en serie. Además, se asume que cada máquina tiene una función diferente. Cada trabajo tiene que pasar por todas las máquinas en el mismo orden, es decir, que todos los trabajos tienen la misma ruta. El término *flow*, se refiere al hecho de que todos los trabajos fluyen de una máquina a otra en el mismo orden. Nótese, que el orden en el que se procesan los trabajos en cada máquina es diferente generalmente. Además, cada máquina solo puede procesar un trabajo a la vez y cada trabajo solo puede ser procesado en una misma máquina al mismo tiempo, que se dispone de buffer infinito y que los tiempos para pasar los trabajos de una máquina a otra son despreciables. Un aspecto interesante a comentar dentro de los *flowshops* es el de la existencia de los *missing operation* que consisten en actuar igual que los *flowshops* pero que durante la ruta que siguen los trabajos, estos se pueden saltar alguna máquina si fuera necesario.

1.5. No-wait (nwt)

Normalmente en un taller, las tareas se procesan en las máquinas justo en el instante en el que llegan, aunque hay veces en las que cuando llega una tarea a una máquina esta todavía no ha terminado de procesar el trabajo anterior por lo que la tarea tiene que esperar en una zona de espera o *buffer*. En la industria, se dan muchas veces situaciones en las que no conviene que esperen las tareas ya sea por el tipo de producto a procesar o por falta de espacio para crear un *buffer* donde esperen las tareas hasta que sean procesadas.

Esta condición de no espera (no-wait), se puede modelar en la programación de la producción como una restricción a la hora de asignar los trabajos a las máquinas correspondientes para encontrar la secuencia óptima.

1.6. Cmax y $\sum C_j$

Las funciones objetivo que se van a tratar en este trabajo son Cmax y $\sum C_j$, se van a calcular las mismas funciones objetivos para todos los modelos para que resulte más fácil la comparación entre ellos.

Como se dijo previamente, Cmax es el tiempo de terminación del último trabajo en la última máquina, es decir, el tiempo en el que finaliza todo el proceso. Mientras que $\sum C_j$ es la suma de los tiempos de finalización de todos los trabajos, es decir, se suman los instantes en los que acaban todos los trabajos en la última máquina.

1.7. Diagramas de Gantt

Los diagramas de Gantt son la forma más representativa y usada que existe en programación lineal para mostrar gráficamente la disposición de los trabajos en cada máquina, su orden, los tiempos de inicio y finalización en cada máquina, los tiempos de espera, los tiempos ociosos o los resultados finales, entre otras cosas. Esta herramienta creada por Karol Adamiecki y posteriormente, fue Henry Gantt quien publicó y divulgó estos diagramas con algunas pequeñas modificaciones, permite visualizar con mayor claridad las rutas de los trabajos, sus disposiciones y aquellos puntos donde se puede mejorar la función objetivo, así como los puntos críticos de las tareas. La aplicación de los diagramas de Gantt dentro de la industria es muy amplia, especialmente, en la programación lineal y en la gestión de proyectos donde resulta vital para la visualización de todas las etapas y actividades en un mismo lugar. Un ejemplo de un diagrama de Gantt es el siguiente:

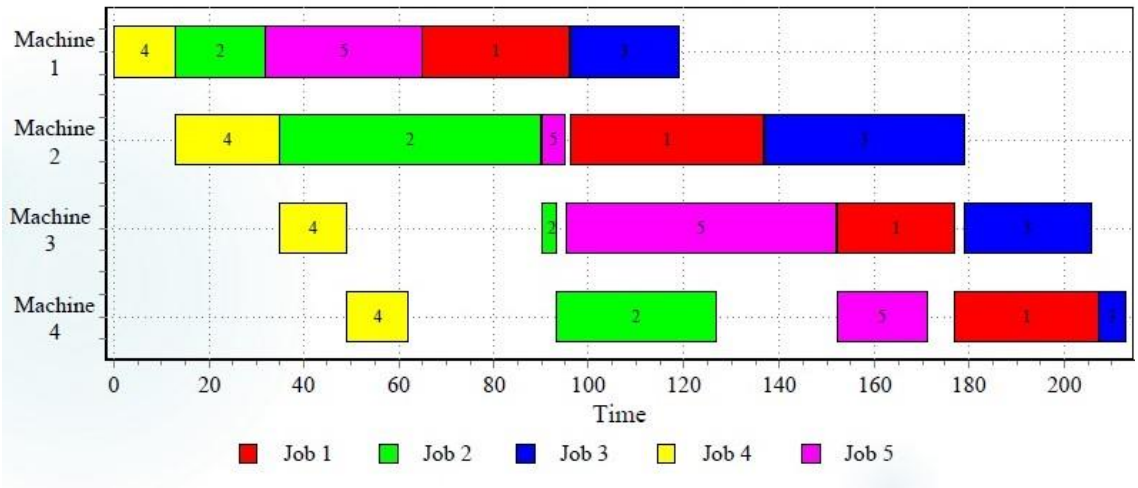

Figura 1: Ejemplo diagrama de Gantt

2. Objeto del proyecto

2.1. Objeto del proyecto

El objetivo del proyecto, consiste en darle una solución óptima a los problemas de producción en los que se tienen varias máquinas que trabajan en serie y cuyas tareas no pueden esperar en un *buffer* hasta que termine la tarea que les precede.

En este proyecto, se van a analizar siete modelos de programación lineal generales en los que se trabaja en un entorno flowshop con permutación para el cálculo del makespan. Una vez estudiados y analizados estos modelos provenientes de dos familias de modelos diferentes en los que cambia el enfoque del modelo y el uso de las variables, se procederá a adaptar y moldear esos modelos a la restricción no-wait que es la que nos ocupa en este proyecto y ver cómo se comportan y los resultados que se obtienen al comprobarlos con varias baterías de datos.

Para ello, se han analizado, estudiado y resuelto todos los modelos comprendiendo y entendiendo todas las variables y restricciones de los modelos, así como las funciones objetivo. Tras ello, en algunos casos, se han tenido que modificar algunas restricciones, en otros casos, ha bastado con añadir o eliminar alguna restricción, todo ello para adaptarlos a la restricción general del problema que es no-wait.

2.2. Justificación

Este proyecto, cuyo fin es analítico, trata de resolver un problema muy común y presente en una gran variedad de empresas como es el hecho de que los productos no puedan o deban esperar en un buffer una vez empiece su proceso de fabricación.

Esto suele ser muy común en empresas del sector químico y farmacéutico debido a la importancia de mantener una serie de condiciones de presión y temperaturas constantes a lo largo de los procesos, así como en el sector farmacéutico donde mantener los medicamentos en unas condiciones óptimas de luz y temperatura garantiza en todo momento la inmunogenicidad y eficacia de estos; en el sector del acero, por también tener que mantener los productos a cierta temperatura durante su tratamiento para mantener o variar las propiedades del acero.

Finalmente, se encuentra uno de los sectores de mayor importancia como es el sector alimenticio, donde la manutención de las cadenas de frío de los alimentos se revela de vital importancia para evitar que estos se pongan malos y haya que desecharlos o en los que puedan aparecer bacterias o virus y que deriven en enfermedades. Además, una vez se rompe la cadena de frío de los alimentos en algún momento, ya no hay vuelta atrás y estos quedan inservibles.

Figura 2: Ejemplo cadena de frío [Fuente: https://gpslogistica.com/services/cadena-de-frio/]

3. Modelos de programación lineal de las familias Wagner y Manne para el permutation flowshop clásico

3.1. Introducción

En esta sección, se van a presentar los modelos de programación lineal de las familias Wagner y Manne para el problema de flowshop con permutación clásico desarrollados en Stafford Jr et al. (2005) y Tseng y Stafford Jr. (2008), es decir, el problema genérico a partir del cual se adaptarán los modelos al problema bajo consideración en este documento, un problema de flowshop con la restricción global no-wait que conlleva permutación, sirviéndome de ayuda los libros de Pinedo (2011) y Framiñán et al. (2014).

3.2. Descripción de los modelos y sus variables

Los modelos que se van a utilizar este documento, tienen como función objetivo el makespan, Cmax y serán adaptados a la misma función objetivo Cmax y SumCj.

3.2.1. Familia Wagner

En primer lugar, se estudiaran los modelos de la familia Wagner, en este caso, el primer modelo es el modelo "WST" y tras analizar todas las restricciones de las que se compone el modelo, se ha podido comprobar que este primer modelo se basa en cuatro variables fundamentales, que son: T_{ri}, que normalmente se escribe como una matriz de M máquinas por N trabajos y que se define como los tiempos de proceso de cada trabajo i en la máquina r; la variable Z_{ij}, se trata una variable binaria que vale 1 si el trabajo i está asignado a la posición j de la secuencia o 0 en cualquier otro caso; $X_{r,j}$, que se define como, el tiempo ocioso de la máquina r antes de que empiece el trabajo en la posición j de la secuencia e $Y_{r,j}$, el tiempo de espera del trabajo de la posición j en la secuencia después de que haya sido procesado en la máquina r.

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{Z_{ij}}$	Variable binaria; 1 si el trabajo i está en la posición j; 0 eoc
$\mathbf{X_{rj}}$	Tiempo ocioso en la máquina r antes de que empiece el trabajo en la posición j
$\mathbf{Y_{rj}}$	Tiempo de espera del trabajo en la posición j después de que acabe de procesarse en la máquina r

Tabla 1: Resumen variables modelo WST

En segundo lugar, otro modelo de la familia Wagner es el modelo "WILSON" que en este caso se basa en tres variables que son: T_{ri} y Z_{ij}, definidas en el modelo anterior ya que tienen el mismo significado al ser de la misma familia y ejercen la misma función y la variable B_{rj}, que se define como, el tiempo de comienzo del trabajo en la posición j de la secuencia en la máquina r.

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{Z_{ij}}$	Variable binaria; 1 si el trabajo i está en la posición j; 0 eoc
$\mathbf{B_{rj}}$	Tiempos de comienzo del trabajo en la posición j en la máquina r

Tabla 2: Resumen variables modelo Wilson

En tercer lugar, analizamos el último modelo de la familia Wagner que es el modelo "TS2" que se basa en tres variables como son: T_{ri} y Z_{ij}, definidas en el primer modelo ya que tienen el mismo significado al ser de la misma familia y ejercen la misma función y en la variable E_{rj}, definida como, el tiempo de terminación del trabajo en la posición j de la secuencia en la máquina r.

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{Z_{ij}}$	Variable binaria; 1, si el trabajo i está en la posición j; 0, eoc
$\mathbf{E_{rj}}$	Tiempo de terminación del trabajo en la posición j

Tabla 3: Resumen variables modelo TS2

3.2.2. Familia Manne

A continuación, en cuarto lugar, pasamos a los modelos de la familia Manne, concretamente al modelo "SGST", cuyas principales variables son: C_{ri}, que se define como, el tiempo de procesado del trabajo i en la máquina r; T_{ri}, tiempos de proceso de cada trabajo i en la máquina r; D_{ik}, que se trata de una variable binaria que vale 1 si el trabajo i está programado en cualquier momento que el trabajo k o 0 en cualquier otro caso y P, una constante cuyo valor ha de ser bastante grande aunque también se suele calcular como la suma de todos los tiempos de proceso de todos los trabajos en todas las máquinas.

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{D_{ik}}$	Variables binaria; 1, si el trabajo i está programado antes que el k
$\mathbf{C_{ri}}$	Tiempo de finalización del trabajo i en la máquina r
P	Constante de valor igual a la suma de todos los tiempos de proceso

Tabla 4: Resumen variables modelo SGST

Finalmente, en quinto lugar, el otro modelo de la familia Manne a tratar es el modelo "Lyeq". Este modelo al ser ambos de la familia Manne comparten algunas variables como son: C_{ri}, T_{ri} y la constante P. Además, se caracteriza por el uso de Q_{rik}, una variable que sirve para evaluar el hecho de que un gran problema con muchas máquinas y trabajos puede ser resuelto en cualquier ordenador por conseguir agrupar dos restricciones en una sola variable.

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{D_{ik}}$	Variables binaria; 1, si el trabajo i está programado antes que el k
$\mathbf{C_{ri}}$	Tiempo de finalización del trabajo i en la máquina r
$\mathbf{Q_{rik}}$	Variables que recoge la agrupación de varias restricciones en una
P	Constante de valor igual a la suma de todos los tiempos de proceso

Tabla 5: Resumen variables modelo LYeq

3.2.3. Modelo adicionales familia Wagner

Por otra parte, como trabajo adicional, se van a analizar dos nuevos modelos derivados de la familia Wagner usando la técnica JALM, (job-adjacency and machine-linkage), trabajos adyacentes y vinculación de las máquinas, usado por primera vez por Tseng y Stafford (2001) para referirse a una serie de restricciones en uno de los modelos MILP para el problema de flowshop SDST, (sequence-dependent setup times), tiempos de setup dependientes de la secuencia. Estos dos modelos se caracterizan por aunar la mayor parte de las restricciones en una sola simplificando en una gran medida los tiempos de análisis de los modelos.

El primer modelo que se va a tratar de los modelos derivados de la familia Wagner es el modelo "TBA", este modelo se basa principalmente en tres variables las cuales ya conocemos sus significados expuestos en los modelos anteriores, principalmente en el modelo "WST" estas variables son: $X_{r,j}$, T_{ri} y Z_{ij}.

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{Z_{ij}}$	Variable binaria; 1 si el trabajo i está en la posición j; 0 eoc
$\mathbf{X_{rj}}$	Tiempo ocioso en la máquina r antes de que empiece el trabajo en la posición j

Tabla 6: Resumen variables modelo TBA

El segundo y último modelo que se va a tratar en este proyecto, es el modelo "TS3", este modelo se basa al igual que el modelo anterior en tres variables donde coinciden en las variables T_{ri} y Z_{ij} y la variable $Y_{r,j}$, también explicadas anteriormente en el modelo "WST".

VARIABLES	SIGNIFICADO
$\mathbf{T_{ri}}$	Tiempos de proceso del trabajo i en la máquina r
$\mathbf{Z_{ij}}$	Variable binaria; 1 si el trabajo i está en la posición j; 0 eoc
$\mathbf{Y_{rj}}$	Tiempo de espera del trabajo en la posición j después de que acabe de procesarse en la máquina r

Tabla 7: Resumen variables modelo TS3

4. Herramientas y metodologías utilizadas

4.1. MILP

La metodología que se va a utilizar en este caso son los modelos de programación linela entera MILP (mixed-integer linear programming). De esta forma, estos problemas se consideran como un híbrido entre varias categorías de modelamiento, siendo el caso más típico el de mezclar variables enteras, binarias y variables continuas.

La metodología MILP se caracteriza por tener la siguiente estructura:

$$\begin{aligned} &\text{maximize} && \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}, \\ &\text{and} && \mathbf{x} \in \mathbb{Z}^n, \end{aligned}$$

Figura 3: Modelo general maximizar MILP [Fuente: https://en.wikipedia.org/wiki/Integer_programming]

Esta estructura, se compone en primer lugar, de un primer término donde se indica si se trata de un problema de maximizar, por ejemplo, los beneficios, o en su defecto, minimizar, por ejemplo, los costes. Junto a este término, aparece representado por "c^Tx" la función objetivo en cuestión la cual se quiere maximizar o minimizar. La variable "x", representa todas las variables que forman parte del modelo de forma genérica, mientras que el término "c" son los coeficientes asociados a las variables en la función objetivo, estos costes, se llaman también costes relativos. En la siguiente línea, aparece el término "subject to", que significa ''sujeto a'' e indica que a continuación van a aparecer todas las restricciones que forman parte del modelo. Estas restricciones se representan de la forma "Ax < = > b", donde "A" representa la matriz de coeficientes tecnológicos del modeo, es decir, los coeficientes de las variables que forman las restricciones del modelo; "x", representa las variables del modelo y "b", los términos independientes de las restricciones. Además, las restricciones se pueden escribir con cualquier restricción de signo (<, =, >, ≤, ≥). Finalmente, el modelo se cierra acotando los límites de las variables indicando si estas son mayores o iguales que cero, variables libres o binarias.

4.2. CodeBlocks y C

El programa que se utilizará para aplicar esta metodología MILP es CodeBlocks en el que se utilizará el lenguaje de programación C/C++. Se trata de un sistema de compilación personalizado que se creó orientado hacia los lenguajes C, C++ y Fortran y que funciona con un conjunto de compiladores de software libre. Aunque en este proyecto, solo se utilizará el lenguaje C. Además, este programa tiene la posibilidad de hacer uso de librerías, que como su propio nombre indica son librerías de funciones ya creadas y definidas de las que se puede hacer uso en cualquier momento para facilitar y simplificar la creación de los códigos. Por ejemplo, una función que se utilizará en este proyecto, es la función <schedule_lib.h> especialmente diseñada para el scheduling (programación de operaciones) y que una de las funciones que trae, por ejemplo, es la función "loadPTimes_nrows" que sirve para leer los tiempos de proceso de una matriz de n filas.

En este programa, se desarrollarán los catorce modelos de programación (siete modelos por dos funciones objetivo cada uno) de modo que estos códigos generen un archivo .txt con el modelo enteramente desarrollado en el que deberá aparecer la función objetivo en cuestión, las restricciones, los límites de las variables y todas las variables que formen parte del modelo definidas, este archivo .txt, estará escrito en formato LP ".lp", linear programming - programación lineal, este formato LP es el que necesita Gurobi, programa de evaluación computacional que resuelve los modelos proporcionándonos la solución óptima del modelo al pasarle una batería de datos, para poder leer y analizar los modelos codificados en C en CodeBlocks.

El lenguaje C/C++ es uno de los lenguajes más utilizados en programación. A continuación, se explicarán los comandos más importantes que se usarán en el desarrollo del código de los modelos. Estos comandos son:

COMANDO	SIGNIFICADO
fprintf	Sirve para escribir en un fichero de texto
fscanf	Sirve para leer dentro de un fichero
for	Sirve para hacer bucle
if/else	Sirve para comparar dos índices
int	Sirve para declarar variables enteras
char	Sirve para declarar cadenas de caracteres
FILE	Sirve para indicar al programa el fichero que se está usando

sprintf	Sirve para escribir en una cadena de caracteres
DIM_MAT_INT	Sirve para establecer la dimensión de una matriz de números enteros
fclose	Sirve para cerrar un fichero

Tabla 8: Resumen de los comandos más usados en C

De esta forma con el comando fprintf podemos escribir el código dentro de un archivo de texto para su posterior lectura. Su modo de funcionamiento es el siguiente:

fprintf (nombre del fichero,"cadena de caracteres a escribir en el fichero", índices)

Por ejemplo:

```
int n=5;
for (i=1;i<=n;i++)
{
        for (j=1;j<=n;j++)
        {
                if (j=n)
                {
                        fprintf(ejemplo,"X_%d_%d + Y_%d_%d + ",i,j);
                }
                else
                {
                        fprintf(ejemplo,"X_%d_%d + Y_%d_%d = 0 ",i,j);
                }
        }
}
```

Esto hace que en el fichero 'ejemplo' se escriba: "$X_{11} + Y_{11} + X_{12} + Y_{12} + \ldots = 0$" hasta que los índices i y j lleguen al valor N, en este caso 5.

De esta forma se escribirán todas las restricciones de los modelos junto con su función objetivo en un fichero de texto para que posteriormente pueda ser resuelto por Gurobi.

4.3. Gurobi

En este proyecto, al trabajar con el programa de optimización Gurobi, se necesita un formato muy concreto para leer y analizar los modelos y resolverlos, detallado en la propia página de la guía de gurobi donde se explica el uso de este formato ver "Formato LP en Gurobi". Este formato, antes mencionado, se denomina Linear Programming (programación lineal) y su extensión es el .lp. Son pequeñas diferencias las que difieren entre el modelo general y el formato LP pero necesarias para su correcta lectura por parte de Gurobi, que se muestran en el siguiente ejemplo:

```
Maximize
  x + y + z
Subject To
  c0: x + y = 1
  c1: x + 5 y + 2 z <= 10
  qc0: x + y + [ x ^ 2 - 2 x * y + 3 y ^ 2 ] <= 5
Bounds
  0 <= x <= 5
  z >= 2
Generals
  x y z
```

Figura 4: Ejemplo del formato LP en Gurobi [Fuente: https://www.gurobi.com/documentation/8.0/refman/lp_format.html]

Una vez adaptados todas las restricciones de los modelos a la restricción global y escritas las dos funciones objetivo en cada modelo con sus variables correspondientes, al código en C se le pasa una batería de datos, también en formato .txt, en la que aparecen el número de máquinas, el número de trabajos y los tiempos de proceso de cada trabajo en cada máquina de cada instancia. Con estos datos y el código en C que genera el archivo .lp, se le pasa a Gurobi con un comando y éste te resuelve el modelo dándote como resultado la solución óptima del problema en cuestión.

4.4. Batch y Registro

Por otra parte, se va a hacer uso del lenguaje BATCH, que se trata de un archivo de procesamiento por lotes. Se trata de archivos de texto con extensión .bat que contienen un conjunto de instrucciones MS-DOS (Sistema Operativo de disco de Microsoft) permitiendo automatizar varias tareas como se detalla en el blog Comando FOR para archivos BAT (2012) de la página blogspot del profesor Emilio Barco. Este archivo se

utilizará para automatizar las llamadas a todas las instancias de datos de una sola vez haciendo que estas se ejecuten en el shell o cmd del ordenador, que es una interfaz de usuario que te permite acceder a los servicios del sistema operativo.

Además, al tener que analizar unas 160 instancias, la creación del archivo .bat en formato BATCH para que las llamadas a Gurobi se realicen todas de una sola vez con un solo comando se antoja realmente necesaria. Estas llamadas a Gurobi, se realizan mediante el cmd (símbolo del sistema), una aplicación del sistema operativo que te permite realizar ciertas acciones. En este caso, se escribiría "gurobi_cl" y las rutas de los modelos para llamar al programa, junto con un límite de tiempo de 180 segundos, para así reducir los tiempos de resolución ya que algunas instancias en algunos modelos al ser tan grandes tardan demasiado tiempo en encontrar la solución.

Por otra parte, para ver el valor de las variables al resolver los modelos para comprobar que no solo el resultado final está bien, sino que los valores de las variables internas son acordes y van en el sentido de las restricciones del modelo, se ha utilizado usando la extensión "gurobi_cl" el comando *ResultFile* que genera un documento de texto con el valor de todas las variables que participan en el modelo.

Posteriormente, una vez resuelto por parte de Gurobi todas las instancias en todos los modelos, se crea automáticamente un archivo .log que se encarga de recoger todos los resultados y guardarlos en un fichero de texto.

Finalmente, gracias a un programa proporcionado por el tutor se pasan todos esos datos a un archivo Excel que en forma de resumen nos muestra para cada instancia en cada modelo el resultado de la función objetivo y el tiempo que ha tardado en resolverse, que se analizará con mayor profundidad en la evaluación computacional.

lanzarcomandos.bat: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
FOR /L %%x IN (5,5,20) DO (
        FOR /L %%y IN (2,1,5) DO (
                FOR /L %%z IN (1,1,10) DO (
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\WSTmodel_%%x_%%y_%%z.lp
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\Wilsonmodel_%%x_%%y_%%z.lp
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\TS2model_%%x_%%y_%%z.lp
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\SGSTmodel_%%x_%%y_%%z.lp
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\LYeqmodel_%%x_%%y_%%z.lp
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\TBAmodel_%%x_%%y_%%z.lp
                        gurobi_cl TimeLimit=600 LogFile=C:\gurobi.log C:\modelos\TS3model_%%x_%%y_%%z.lp
                                )
                        )
                )
```

Figura 5: Código del archivo .bat

5. MILPs para el problema bajo consideración

5.1. Familia Wagner

El modelo WST de la familia Wagner se caracteriza por basar sus restricciones en las variables X e Y, que representan los tiempos de espera y los tiempos ociosos de los trabajos y cuya función objetivo es el makespan, que quiere decir, el tiempo en el que termina de procesarse el último trabajo en la última máquina. En este sentido, el modelo general se representa por:

$$\text{min makespan} = F_{MAX} = C_{MAX} = C_{MN} = \sum_{i=1}^{N} T_{Mi} + \sum_{p=1}^{N} X_{Mp}$$

$$s.a: \quad \sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \le i \le N) \qquad (1)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \le j \le N) \qquad (2)$$

$$\sum_{i=1}^{N} T_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{N} T_{r+1,i} Z_{ij} + X_{r+1,j+1} + Y_{rj} \quad (1 \le r \le M-1; 1 \le j \le N-1) \qquad (3)$$

$$X_{r+1,1} = X_{r,1} + Y_{r,1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \qquad (1 \le r \le M-1) \quad (4)$$

$$Y_{r1} = 0 \qquad (1 \le r \le M-1) \quad (5)$$

En este modelo, tenemos las restricciones (1) y (2) que sirven para asegurarse de que en cada posición son hay un trabajo y que cada trabajo solo ocupa una posición. La restricción (3), hace referencia a que la suma de los tiempos de proceso en una máquina más los tiempos ociosos en esa máquina más los tiempos de espera, tiene que ser igual en la máquina siguiente. La restricción (4), hace referencia a que el tiempo ocioso de una máquina sea igual al tiempo ocioso de la máquina anterior más el tiempo de espera y el tiempo de proceso. Finalmente, la restricción (5), nos asegurar que el primer trabajo nunca en cada máquina empieza inmediatamente y no espera.

Una vez analizado el modelo, se procede a adaptar las restricciones de este a la restricción global no-wait. De este modo, en este modelo por la propia definición de la variable Y que representa los tiempos de espera de los trabajos en las máquinas, hace que la adaptación consista en obligar a esa variable a valer 0 siempre, con lo cual valdría con eliminar esa variable de todas las restricciones del modelo. De esta forma el modelo quedaría de esta manera:

$$\min \text{makespan} = F_{MAX} = C_{MAX} = C_{MN} = \sum_{i=1}^{N} T_{Mi} + \sum_{p=1}^{N} X_{Mp}$$

$$s.a: \quad \sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \le i \le N) \qquad (1)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \le j \le N) \qquad (2)$$

$$\sum_{i=1}^{N} T_{ri} Z_{i,j+1} + X_{r,j+1} = \sum_{i=1}^{N} T_{r+1,i} Z_{ij} + X_{r+1,j+1} \qquad (1 \le r \le M-1; 1 \le j \le N-1) \qquad (3')$$

$$X_{r+1,1} = X_{r,1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \qquad (1 \le r \le M-1) \quad (4')$$

Una vez adaptadas las restricciones del modelo, le damos forma a la función objetivo $\sum C_j$, que consiste en el sumatorio de todos los tiempos de finalización de todos los trabajos en la última máquina y que queda de la siguiente forma:

$$\min SumCj = \sum_{j=1}^{N} (n - j + 1) X_{Mj} + \sum_{j=1}^{N} \sum_{i=1}^{N} (n - j + 1) T_{Mi} Z_{ij}$$

El modelo Wilson de la familia Wagner se caracteriza por basar sus restricciones en la variable B, que representa los tiempos de inicio de los trabajos. Con ello, el modelo general se representa por:

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

$$s.a: \quad \sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \le i \le N) \qquad (1)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \le j \le N) \qquad (2)$$

$$B_{1j} + \sum_{i=1}^{N} T_{1i} Z_{ij} = B_{1,j+1} \qquad (1 \le j \le N-1) \qquad (3)$$

$$B_{11} = 0 \qquad (4)$$

$$B_{r1} + \sum_{i=1}^{N} T_{ri} Z_{i1} = B_{r+1,1} \qquad (1 \le r \le M-1) \qquad (5)$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \le B_{r+1,j} \qquad (1 \le r \le M-1; 2 \le j \le N) \qquad (6)$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \le B_{r,j+1} \qquad (2 \le r \le M; 1 \le j \le N-1) \qquad (7)$$

En este modelo, las dos primeras restricciones son iguales que en el modelo anterior ya que son de la misma familia. La restricción (3), establece que el tiempo de inicio de un trabajo más su tiempo de proceso tiene que ser igual que el tiempo de inicio en la máquina siguiente. La restricción (4), obliga a que el primer trabajo empiece en la primera máquina. La restricción (5), establece que el tiempo en el que entra el primer trabajo en una máquina más su tiempo de proceso sea igual al tiempo de inicio en la máquina siguiente. Las restricciones (6) y (7), son una pareja de restricciones que establecen que los tiempos de inicio de los trabajos en una máquina tienen que ser menor o igual que los tiempos de inicio del siguiente trabajo en la máquina siguiente.

Analizado el modelo Wilson, se procede a adaptarlo a la restricción general no-wait. De este modo, en este modelo, para conseguir el comportamiento no-wait se han modificado los signos de las restricciones (3) y (6) para que los tiempos de comienzo de los trabajos en una máquina más sus tiempos de procesado sean igual que los tiempos de comienzo

en la máquina siguiente para que los trabajos no tengan que esperar. De esta manera, las restricciones (3) y (6) del modelo quedarían de la siguiente forma:

$$B_{1j} + \sum_{i=1}^{N} T_{1i} Z_{ij} \leq B_{1,j+1} \qquad (1 \leq j \leq N-1) \qquad (3')$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} = B_{r+1,j} \qquad (1 \leq r \leq M-1; 2 \leq j \leq N) \qquad (6')$$

Con las restricciones ya adaptadas, se procede a escribir la función objetivo $\sum C_j$ del modelo Wilson que queda de la siguiente forma:

$$\min SumCj = \sum_{j=1}^{N} B_{Mj} + \sum_{j=1}^{N} \sum_{i=1}^{N} T_{Mi} Z_{ij}$$

El modelo TS2 de la familia Wagner se caracteriza por basar sus restricciones en la variable E, que representa el tiempo de terminación final de los trabajos. Por ello, el modelo general se representa por:

$$\min \text{makespan} = F_{MAX} = E_{MN}$$

$$s.a: \quad \sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \leq i \leq N) \qquad (1)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \leq j \leq N) \qquad (2)$$

$$E_{rj} + \sum_{i=1}^{N} T_{ri} Z_{i,j+1} \leq E_{r,j+1} \qquad (1 \leq r \leq M; 1 \leq j \leq N-1) \qquad (3)$$

$$E_{rj} + \sum_{i=1}^{N} T_{r+1,i} Z_{ij} \leq E_{r+1,j} \qquad (1 \leq r \leq M-1; 1 \leq j \leq N) \qquad (4)$$

$$E_{11} \geq \sum_{i=1}^{N} T_{1i} Z_{i1} \qquad (5)$$

En este modelo, al igual que en el modelo anterior, las restricciones (1) y (2) son iguales que en el primer modelo al ser también de la misma familia. Las restricciones (3) y (4), establecen que el tiempo de finalización de un trabajo más el tiempo de proceso del siguiente o de este en la siguiente máquina tiene que ser menor o igual que el tiempo de finalización de este o en la siguiente máquina. La restricción (5), establece que el tiempo de finalización del primer trabajo en la primera máquina tiene que ser mayor o igual que su tiempo de proceso en esa máquina.

Analizadas todas las restricciones del modelo TS2, se procede a adaptarlas a la restricción global no-wait. Para ello, de acuerdo con las variables de este modelo se requiere únicamente la adaptación de la restricción número 4, manteniendo igual el resto de restricciones, de modo que el tiempo de finalización de un trabajo en una máquina más el tiempo de proceso de ese trabajo en la siguiente máquina sea igual al tiempo de finalización del trabajo en esa siguiente máquina. De modo, que la restricción (4) queda de la siguiente forma:

$$E_{rj} + \sum_{i=1}^{N} T_{r+1,i} Z_{ij} = E_{r+1,j} \qquad (1 \le r \le M-1; 1 \le j \le N) \qquad (4')$$

Ahora vamos con la función objetivo $\sum C_j$ del modelo TS2 que queda de la siguiente forma:

$$\min SumCj = \sum_{i=1}^{N} E_{Mi}$$

5.2. Familia Manne

El modelo SGST de la familia Manne se caracteriza por basar sus restricciones en la variable C, que representa el tiempo de terminación de procesado de los trabajos. Con lo cual, el modelo general se representa por:

$$\min makespan = F_{MAX} = C_{MAX}$$

$$\begin{aligned}
s.a: \quad & C_{1i} \ge T_{1i} && (1 \le i \le N) && (1)\\
& C_{r+1,i} - C_{ri} \ge T_{r+1,i} && (1 \le r \le M-1; 1 \le i \le N) && (2)\\
& C_{ri} - C_{rk} + PD_{ik} \ge T_{ri} && (1 \le r \le M; 1 \le i < k \le N) && (3)\\
& C_{rk} - C_{ri} + P(1 - D_{ik}) \ge T_{rk} && (1 \le r \le M; 1 \le i < k \le N) && (4)\\
& C_{MAX} \ge C_{Mi} && (1 \le i \le N) && (5)
\end{aligned}$$

En este modelo, la primera restricción, establece que el instante de terminación de un trabajo en la primera máquina tiene que ser mayor o igual que el tiempo de proceso de

ese trabajo en la primera máquina. La segunda restricción, establece que el instante de terminación de un trabajo en una máquina menos ese instante en la máquina anterior tiene que ser mayor o igual al tiempo de proceso de ese trabajo en la primera máquina mencionada. La tercera restricción, establece que el instante de terminación de un trabajo menos el instante de terminación del trabajo que le sigue más la multiplicación de una constante de valor la suma de todos los tiempos de proceso de todos los trabajos en todas las máquinas por una variable binaria que vale 1 si el trabajo i está antes que el trabajo j y 0 en cualquier otro caso tiene que ser mayor o igual que el tiempo de proceso del primer trabajo en esa máquina. La cuarta restricción, es igual que la tercera, pero cambiando el orden de los índices i y k y por ello el tercer término es 1 menos la variable binaria. La quinta restricción, establece simplemente que el mayor instante de finalización tiene que ser mayor o igual que el instante de finalización de cada uno de los trabajos de la última máquina.

A la hora de adaptar el modelo a la restricción global no-wait, nos centramos en la segunda restricción de tal forma que, cambiamos el signo de la restricción de '≥' a '=' para que el tiempo de terminación de un trabajo en una máquina menos el tiempo de terminación de ese mismo trabajo en la máquina anterior sea igual al tiempo de proceso de ese trabajo en la máquina siguiente. De esta forma, la segunda restricción queda de la siguiente manera:

$$C_{r+1,i} - C_{ri} = T_{r+1,i} \qquad (1 \le r \le M-1; 1 \le i \le N) \qquad (2')$$

Para el modelo SGST, la función objetivo adaptada para $\sum C_j$ quedaría de la siguiente forma:

$$\min SumCj = \sum_{i=1}^{N} C_{Mi}$$

El modelo LYeq de la familia Manne se caracteriza por basar sus restricciones en la variable C al igual que el modelo anterior y la variable Q, que representa la combinación de una pareja de restricciones. Por lo tanto, el modelo general se representa por:

$$\min makespan = F_{MAX} = C_{MAX}$$

$$s.a: \quad PD_{ik} + C_{ri} - C_{rk} - T_{ri} = Q_{rik} \qquad (1 \le r \le M; 1 \le i < k \le N) \qquad (1)$$

$$Q_{rik} \le P - T_{ri} - T_{rk} \qquad (1 \le r \le M; 1 \le i < k \le N) \qquad (2)$$

$$C_{1i} \ge T_{1i} \qquad (1 \le i \le N) \qquad (3)$$

$$C_{r+1,i} - C_{ri} \ge T_{r+1,i} \qquad (1 \le r \le M-1; 1 \le i \le N) \qquad (4)$$

$$C_{MAX} \ge C_{Mi} \qquad (1 \le i \le N) \qquad (5)$$

En este modelo, como pasó con la familia anterior, al ser también de la misma familia comparten varias restricciones, en este caso, son las restricciones (3), (4) y (5) que coinciden con las restricciones (1), (2) y (5) del modelo anterior. La primera restricción, establece que la multiplicación de la constante P por la variable binaria más el instante de terminación del trabajo i en una máquina menos su tiempo de proceso y menos el instante de terminación del trabajo que le sigue en la misma máquina tiene que ser igual que la variable Q_{rik} que representa la unión de varias restricciones en una. La segunda restricción, establece que la variable Q_{rik} tiene que ser mayor o igual que la constante P menos el tiempo de proceso de un trabajo en una máquina menos el tiempo de proceso del trabajo que le sigue en la misma máquina.

En la adaptación del modelo LYeq, se procede igual que con el modelo anterior SGST, ya que la adaptación consiste en modificar la misma restricción, aunque en este caso sea el número 4 pero consiste en cambiar el signo '≥' por '=' por la misma razón que en el modelo anterior. Con lo cual, quedaría de la siguiente manera:

$$C_{r+1,i} - C_{ri} = T_{r+1,i} \qquad (1 \le r \le M-1; 1 \le i \le N) \qquad (4')$$

Si la adaptación de las restricciones es la misma en los dos modelos, lo mismo pasa con la función objetivo $\sum C_j$, con lo que la función objetivo queda de la siguiente forma:

$$\min SumCj = \sum_{i=1}^{N} C_{Mi}$$

5.3. Modelos adicionales familia Wagner

El modelo TBA de la familia Wagner se caracteriza por aunar las restricciones más características del modelo en una sola utilizando principalmente los tiempos de proceso y la variable X, que representa los tiempos ociosos de las máquinas. Con lo que el modelo queda de la siguiente forma:

$$\text{min makespan} = F_{MAX} = \sum_{p=1}^{M-1}\sum_{i=1}^{N} T_{pi} Z_{i1} + \sum_{i=1}^{N} T_{Mi} + \sum_{s=2}^{N} X_{Ms}$$

$$s.a: \quad \sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \le i \le N) \qquad (1)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \le j \le N) \qquad (2)$$

$$\sum_{i=1}^{N} T_{r-1,i} Z_{i1} + \sum_{q=1}^{j-1}\sum_{i=1}^{N} (T_{ri} - T_{r-1,i}) Z_{iq} + \sum_{s=2}^{j} (X_{rs} - X_{r-1,s}) - \sum_{i=1}^{N} T_{r-1,i} Z_{ij} \ge 0 \quad (2 \le r \le M; 2 \le j \le N) \quad (3)$$

$$X_{1j} = 0 \qquad (2 \le j \le N) \qquad (4)$$

En la adaptación de las restricciones del modelo TBA a la restricción general no-wait, nos encontramos con que este modelo tienes menos restricciones sobre las que trabajar que los modelos anteriores, este hecho, puede hacer que su adaptación tenga mayor o menor dificultad, aunque una vez estudiadas y analizadas las 4 restricciones, se puede comprobar que en las dos primeras no se puede hacer nada, mientras que a simple vista por definición de la variable X que la última restricción solo debería afectar al primer trabajo de la primera máquina y no a todos; mientras que en la tercera restricción basta con hacer que sea igual a cero. De tal forma que las dos restricciones quedarían de la siguiente forma:

$$\sum_{i=1}^{N} T_{r-1,i} Z_{i1} + \sum_{q=1}^{j-1}\sum_{i=1}^{N} (T_{ri} - T_{r-1,i}) Z_{iq} + \sum_{s=2}^{j} (X_{rs} - X_{r-1,s}) - \sum_{i=1}^{N} T_{r-1,i} Z_{ij} = 0 \quad (2 \le r \le M; 2 \le j \le N) \quad (3')$$

$$X_{11} = 0 \qquad (2 \le j \le N) \qquad (4')$$

Debido a la gran semejanza entre este modelo y el primer modelo, el modelo WST, se ha comprobado que a lo hora de escribir la función objetivo para 'SumCj', resulta ser la misma función debido a las variables que comparten. Por ello, la función objetivo quedaría de la siguiente manera:

$$\min SumCj = \sum_{j=1}^{N}(n-j+1)X_{Mj} + \sum_{j=1}^{N}\sum_{i=1}^{N}(n-j+1)T_{Mi}Z_{ij}$$

El modelo TS3 de la familia Wagner se caracteriza al igual que el modelo anterior por aunar las restricciones más características del modelo en una sola pero esta vez, en vez de usar la variable X, se utiliza la variable Y, que representa los tiempos de espera de los trabajos en las máquinas. Por lo tanto, el modelo general quedaría de la siguiente forma:

$$\min \text{makespan} = F_{MAX} = \sum_{p=1}^{N} T_{1p} + \sum_{q=2}^{M}\sum_{i=1}^{N} T_{qi}Z_{iN} + \sum_{q=1}^{M-1} Y_{qN}$$

$$s.a: \quad \sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \le i \le N) \qquad (1)$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \le j \le N) \qquad (2)$$

$$\sum_{i=1}^{N} T_{1i}Z_{i,j-1} - \sum_{i=1}^{N} T_{ri}Z_{i,j-1} + \sum_{q=1}^{r-1}\sum_{i=1}^{N} T_{qi}(Z_{ij} - Z_{i,j-1}) + \sum_{q=1}^{r-1}(Y_{qj} - Y_{q,j-1}) \ge 0 \quad (2 \le r \le M; 2 \le j \le N) \quad (3)$$

$$Y_{r1} = 0 \qquad (1 \le r \le M-1) \quad (4)$$

Finalmente, llegamos a la adaptación del último modelo del proyecto, el cual llega a ser el más difícil de modelar debido a las variables propias del modelo ya que únicamente se compone de los tiempos de proceso de los trabajos en las máquinas y de los tiempos de espera de los trabajos. Además, para mayor dificultad, sabemos que por definición de la variable Y que es a la que están asignados los tiempos de espera de los trabajos en las máquinas tiene que ser igual a cero para adecuar las restricciones a la restricción global no-wait (sin esperas), con lo cual nos quedamos únicamente con la variable T, es decir, los tiempos de proceso de los trabajos en las máquinas. Con ello el modelo no se podría resolver porque nos faltarían datos de modo que lo que se ha hecho ha sido modificar el significado de la variable Yq para que, en vez de mostrarnos el tiempo de espera entre un mismo trabajo entre varias máquinas, nos muestre el tiempo de espera entre los trabajos de la primera máquina. Con lo cual, al hacer este cambio, se ha tenido que cambiar

también la función objetivo del modelo adaptándola al nuevo significado de la variable Yq. Y con ello, desaparece la ecuación (4).

$$\min \text{makespan} = F_{MAX} = \sum_{p=1}^{N} T_{1p} + \sum_{q=2}^{M}\sum_{i=1}^{N} T_{qi} Z_{iN} + \sum_{q=2}^{N} Y_q$$

$$\sum_{i=1}^{N} T_{1i} Z_{i,j-1} - \sum_{i=1}^{N} T_{ri} Z_{i,j-1} + \sum_{q=1}^{r-1}\sum_{i=1}^{N} T_{qi}(Z_{ij} - Z_{i,j-1}) + \sum_{q=2}^{j}(Y_q - Y_{q-1}) \geq 0 \quad (2 \leq r \leq M; 2 \leq j \leq N) \; (3')$$

Por último, la función objetivo $\sum C_j$ con la modificación de la variable Yq quedaría de la siguiente manera:

$$\min SumCj = \sum_{q=2}^{M}\sum_{i=1}^{N} T_{qi} + \sum_{j=2}^{N}\sum_{q=2}^{N}(n-j+1)Yq + \sum_{j=1}^{N}\sum_{i=1}^{N}(n-j+1)T_{1i}Z_{ij}$$

6. Evaluación computacional

6.1. Datos del problema

Este proyecto, se basa en unos modelos generales de programación lineal para un entorno de flowshop con permutación para la función objetivo Cmax y la función objetivo $\sum C_j$. Estos modelos descritos en los documentos mencionados en la sección 3.2, descripción del modelo y sus variables, son la base del proyecto sobre la que se trabajará. Además, para posteriormente comprobar todas las adaptaciones de los modelos, se utilizarán una serie de instancias de datos que van desde 5 a 20 trabajo y desde 2 a 5 máquinas. Estas instancias de datos vienen recogidas en ficheros de texto de tal manera que se leen así (ver figura) donde el primer término '3' es el número de máquinas, '4' es el número de trabajos y la matriz que le sigue son los tiempos de proceso de cada trabajo en cada máquina.

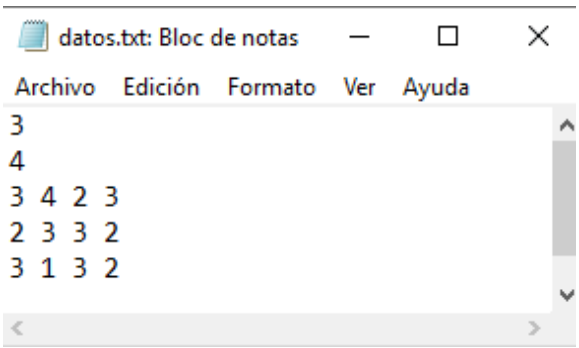

Figura 6: Datos de archivo de ejemplo

6.2. Evaluación numérica

En este punto, se analizarán y evaluarán los resultados computacionales de los modelos ya adaptados descritos previamente en el punto anterior para comprobar numéricamente si realmente están bien adaptados a la restricción general no-wait o existe algún fallo en algunas restricciones o alguna función objetivo no funciona como debiera. Para ello, primero se comprobaron todos los modelos con un pequeño ejemplo previo que consistía en establecer 3 máquinas (m=3) y 4 trabajos (n=4) y cuya matriz de tiempos de procesos es:

m/n	**1**	**2**	**3**	**4**
1	3	4	2	3
2	2	3	3	2
3	3	1	3	2

Tabla 9: Matriz de tiempos de proceso del ejemplo

Con estos datos de ejemplo, se comprobaron previamente (con la ayuda del Excel y de los diagramas de Gantt) cuales eran las secuencias óptimas del problema y los valores de la función objetivo Cmax (makespan) y $\sum C_j$ óptimos de los modelos, dando como resultado las siguientes secuencias óptimas que dan lugar a las soluciones del problema (ver tabla 10): [1-3-2-4], [1-4-3-2], [3-2-1-4] y [4-1-3-2], a partir de estas secuencias óptimas, se obtienen también los resultados de las funciones objetivo. Por una parte, tenemos la función objetivo Cmax o makespan cuyo resultado o valor óptimo es 16, que quiere decir, que el menor tiempo en el que puede terminar de procesarse el último trabajo en la última máquina es 16, y, por otra parte, tenemos la función objetivo $\sum C_j$ cuyo resultado o valor óptimo es 48, es decir, que el conjunto de los 4 trabajos, en este caso, no acabaran nunca de procesarse en la última máquina antes de las 48 unidades de tiempo dadas como solución. A continuación, se presentan los 4 diagramas de Gantt que representan las 4 secuencias óptimas asociadas a los datos concretos y a las funciones objetivo propuestas.

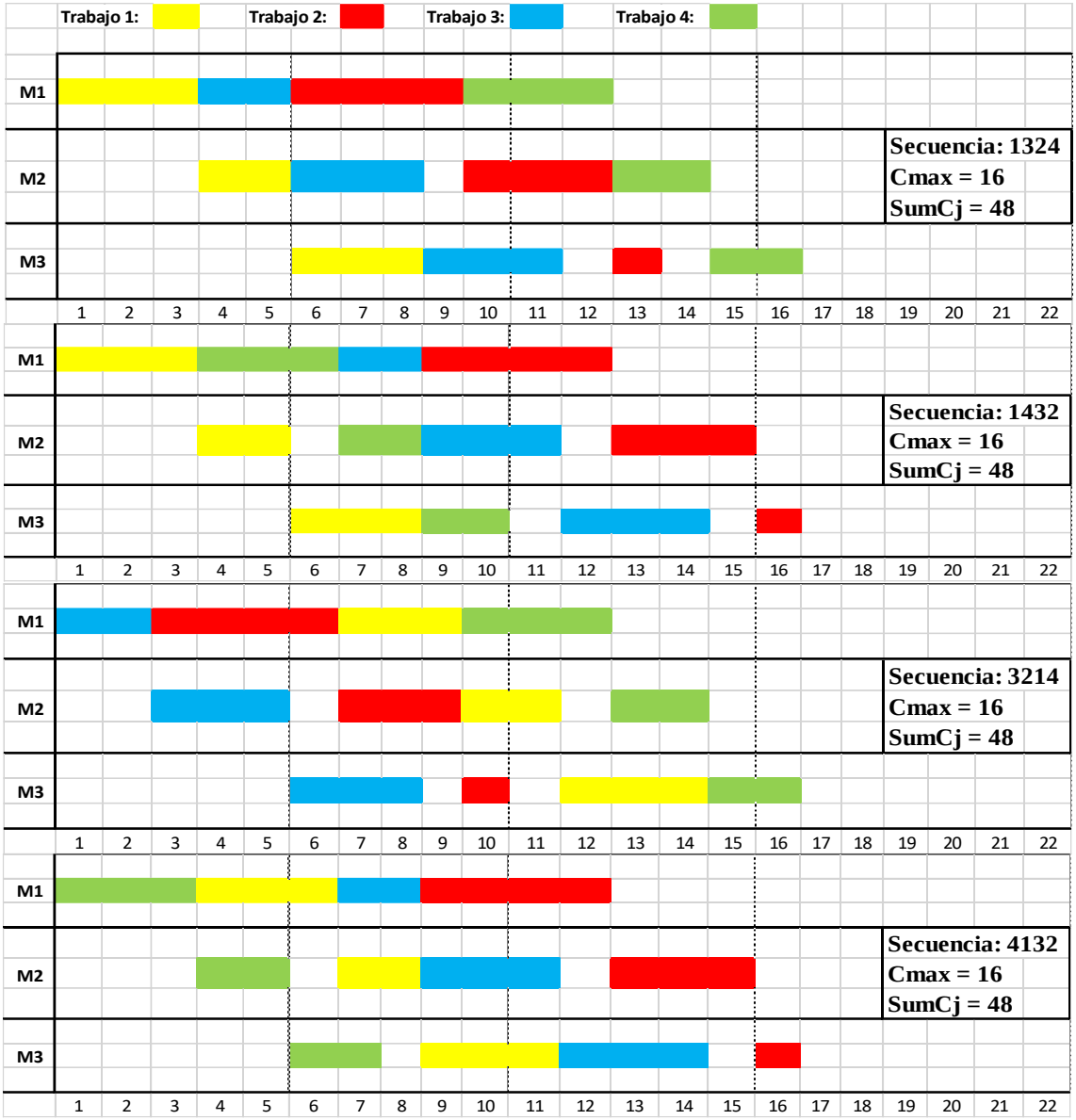

Tabla 10: Diagramas de Gantt del ejemplo

Tras mostrar estos ejemplos, se procede a reanalizar todos los modelos con una batería de datos para diferentes números de trabajos y de máquinas. Esta batería de datos consiste en 160 instancias o ficheros con máquinas que van de 2 a 5 y los trabajos de 5 a 20.

A continuación, se presentan los resultados del análisis de las 160 instancias en cada uno de los modelos y de sus variantes (cuando se habla de variantes, se habla de que la función objetivo principal es el makespan y que, por lo tanto, su variante es el $\sum C_j$).

En este análisis, se han recogido los datos correspondientes a los tiempos de computación de cada modelo para cada tamaño de problema, los valores de las funciones objetivo y los casos en los que el modelo no llega a encontrar el óptimo en el tiempo límite establecido, que era de 600 segundos.

6.2.1. Evaluación numérica Cmax

En primer lugar, analizaremos los casos en los que los modelos no te dicen cuál es el óptimo del problema por llegar al límite de tiempo. En este caso, definiremos #O, como el número de veces en las que el modelo encuentra el óptimo para cada tamaño de problema (siendo 0 el mínimo y 10 el máximo, al haber 10 instancias por cada bloque); #F, es el número de veces en las que el modelo encuentra una solución, es decir, es factible, pero no sabe si es o no la óptima al alcanzar el límite de tiempo de 600 segundos; y #N, es el número de veces en las que el modelo no es capaz de encontrar una solución al problema.

FO: Cmax	WST			WILSON			TS2			SGST			Lyeq			TBA			TS3		
Tamaño del problema	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N
5x2	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
5x3	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
5x4	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
5x5	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x2	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x3	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x4	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x5	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
15x2	10	0	0	10	0	0	10	0	0	0	10	0	0	10	0	10	0	0	10	0	0
15x3	10	0	0	10	0	0	10	0	0	1	9	0	0	10	0	10	0	0	10	0	0
15x4	10	0	0	9	1	0	8	2	0	2	8	0	0	10	0	9	1	0	10	0	0
15x5	10	0	0	7	3	0	6	4	0	5	5	0	2	8	0	7	3	0	10	0	0
20x2	6	4	0	5	5	0	5	5	0	0	10	0	0	10	0	5	5	0	6	4	0
20x3	3	7	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	4	6	0
20x4	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0
20x5	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0
MEDIA	8,1	1,9	0	7,6	2,4	0	7	3	0	6	5	0	5	5	0	8	2	0	8	2	0

Tabla 11: Análisis de resultados de la función objetivo Cmax

Como se puede apreciar en la tabla de la que se pueden sacar varias conclusiones, vemos que los modelos de la familia Wagner, tienen mayor facilidad de encontrar el óptimo del problema si se trata de la función objetivo Cmax; por otra parte, a los modelos de la familia Manne le pasa lo contrario y es que estos dos modelos son especialmente lentos ya que no llegan a encontrar el óptimo en ningún caso cuando se llegan a analizar 20 trabajos. Además, podemos comprobar que todos los modelos como es lógico siempre encuentran el óptimo en los problemas pequeños de pocos trabajos y pocas máquinas mientras que cuando nos vamos a 15 o 20 trabajos y 4 o 5 máquinas ya les cuesta más al tener que realizar muchas más iteraciones. Finalmente, podemos comprobar que los mejores modelos son los modelos WST y TS3, ambos de la familia Wagner.

En segundo lugar, analizaremos los tiempos de computación de los modelos, es decir, lo que tarda cada modelo en analizar todas las instancias de datos agrupadas por número de máquinas y trabajos.

CPU TIMES - FO: Cmax

Tamaño del problema	MODELO WST	MODELO WILSON	MODELO TS2	MODELO SGST	MODELO Lyeq	MODELO TBA	MODELO TS3
5X2	0,40	0,42	0,42	0,63	0,82	0,46	0,37
5X3	0,57	0,73	0,71	0,91	1,11	0,72	0,50
5X4	0,64	0,71	0,77	1,23	1,23	0,78	0,49
5X5	0,51	0,75	0,86	1,25	1,41	0,77	0,56
10X2	2,15	4,41	4,06	13,78	24,48	4,00	2,36
10X3	5,67	6,00	6,10	13,33	26,31	7,86	5,13
10X4	6,74	11,9	11,27	21,69	34,53	14,37	6,76
10X5	15,58	20,11	18,55	17,93	23,06	24,67	10,25
15X2	35,67	55,69	57,65	6000,27	6000,29	53,10	25,19
15X3	324,93	772,95	1233,94	5892,31	6000,27	819,07	326,40
15X4	1080,85	2509,65	3006,68	5463,90	6000,27	3044,64	748,96
15X5	1882,43	3370,10	3736,61	4546,18	5830,74	4706,34	1085,11
20X2	2557,29	3582,72	3152,36	6000,51	6000,29	3097,57	2575,32
20X3	5337,98	6000,31	6000,29	6000,31	6000,38	6000,36	5146,68
20X4	6000,49	6000,44	6000,29	6000,36	6000,42	6000,69	6000,44
20X5	6000,51	6000,37	6000,32	6000,43	6000,50	6000,74	6000,35
MEDIA	1.453,28	1771,08	1826,93	2873,44	2996,63	1861,01	1370,93

Tabla 12: Tiempos de computación en la función objetivo Cmax

Como se ve en la tabla, podemos apreciar los tiempos de proceso de todos los modelos para todos los bloques de instancias y vemos que al igual que se mencionó anteriormente, los modelos WST y TS3 son los mejores en este apartado y son los que menos tardan en procesar todas las instancias de media, recalcando que el modelo TS3 el ligeramente más rápido que el modelo WST.

Finalmente, en tercer lugar, se calculará el valor de ARPD (Average Relative Percentage Deviation) para todos los modelos, análisis de las desviaciones medias relativas de los porcentajes, es decir, se analizarán los valores de las funciones objetivo de cada modelo en cada instancia comparando el valor de la función objetivo de un modelo con el valor mínimo que ha obtenido otro modelo o él mismo en esa instancia (se compara con el mínimo ya que el problema es de minimizar, por lo tanto, si el problema fuera de maximizar, se compararía con el máximo).

ARPD - FO: Cmax							
	WST	WILSON	TS2	SGST	Lyeq	TBA	TS3
5X2	0	0	0	0	0	0	0
5X3	0	0	0	0	0	0	0
5X4	0	0	0	0	0	0	0
5X5	0	0	0	0	0	0	0
10X2	0	0	0	0	0	0	0
10X3	0	0	0	0	0	0	0
10X4	0	0	0	0	0	0	0
10X5	0	0	0	0	0	0	0
15X2	0	0	0	0,297	0,345	0	0
15X3	0	0	0	0	0,214	0	0
15X4	0	0,415	0,425	0,635	0,768	0,765	0
15X5	0	0,540	0,846	0,837	0,986	0,720	0
20X2	0	0,211	0	0,278	0,714	0,098	0
20X3	0,320	0,426	0,698	1,379	3,035	1,027	0,155
20X4	1,082	1,092	1,039	1,309	3,674	1,717	0,614
20X5	0,653	1,078	1,149	1,403	2,697	2,515	0,362

Tabla 13: Tabla de ARPDs de la función objetivo Cmax

Como vemos en la tabla y teniendo en cuenta los análisis anteriores, se puede comprobar y deducir fácilmente que en las primeras instancias de datos todos los modelos encuentran el óptimo y, por lo tanto, su ARPD es cero. No es hasta que se llega a los 15 trabajos y 2 máquinas cuando vemos que los modelo SGST y LYeq no encuentran el óptimo y comparado con los otros modelos que sí lo han hecho, su ARPD vale 0.297% y 0.345%, respectivamente para el caso de Cmax. Cabe resaltar, que el mayor ARPD, es decir la mayor diferencia entre la función objetivo de un modelo y el valor mínimo de la función objetivo de todos los modelos se encuentra para el bloque de 20 trabajos y 4 máquinas con un valor de 3.674%.

6.2.2. Evaluación numérica $\sum C_j$

Ahora, se realizarán los mismos cálculos que en el apartado anterior, pero en este caso, para la función objetivo $\sum C_j$. Por lo tanto, en primer lugar, analizaremos los casos en los que los modelos no te dicen cuál es el óptimo del problema por llegar al límite de tiempo.

FO: SumCj	WST			WILSON			TS2			SGST			Lyeq			TBA			TS3		
Tamaño del problema	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N	#0	#F	#N
5x2	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
5x3	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
5x4	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
5x5	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x2	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x3	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x4	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
10x5	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0
15x2	10	0	0	10	0	0	10	0	0	0	10	0	6	4	0	10	0	0	10	0	0
15x3	10	0	0	10	0	0	10	0	0	0	10	0	5	5	0	10	0	0	10	0	0
15x4	10	0	0	10	0	0	10	0	0	0	10	0	3	7	0	10	0	0	10	0	0
15x5	10	0	0	10	0	0	10	0	0	0	10	0	4	6	0	10	0	0	10	0	0
20x2	10	0	0	10	0	0	10	0	0	0	10	0	0	10	0	10	0	0	10	0	0
20x3	10	0	0	10	0	0	10	0	0	0	10	0	0	10	0	10	0	0	10	0	0
20x4	10	0	0	5	5	0	4	6	0	0	10	0	0	10	0	4	6	0	10	0	0
20x5	3	7	0	1	9	0	0	10	0	0	10	0	0	10	0	0	10	0	8	2	0
MEDIA	9,6	0,4	0	9,1	0,9	0	9	1	0	5	5	0	6	4	0	9	1	0	10	0	0

Tabla 14: Análisis de resultados de la función objetivo $\sum C_j$

Al igual que en el punto anterior, como se puede apreciar en la tabla de la que se pueden sacar varias conclusiones, vemos que los modelos de la familia Wagner, vuelven a ser lo que tienen una mayor facilidad para encontrar el óptimo del problema. En este caso, vemos que lo modelos de la familia Manne, vuelven a fallar para problemas de grandes dimensiones, llegan hasta el modelo SGST a no llegar a encontrar el óptimo para problemas con 20 trabajos como en la otra función objetivo y hasta con 15 trabajos. Además, vemos que todos los modelos como con la otra función objetivo encuentran el óptimo para los problemas de pequeñas dimensiones, siendo el modelo TS3 el mejor en todo el conjunto.

En segundo lugar, analizaremos los tiempos de computación de los modelos para la función objetivo $\sum C_j$.

CPU TIMES - FO: SumCj

Tamaño del	MODELO WST	MODELO WILSON	MODELO TS2	MODELO SGST	MODELO Lyeq	MODELO TBA	MODELO TS3
5X2	0,42	0,43	0,54	0,94	0,88	0,45	0,35
5X3	0,86	1,24	1,04	1,29	1,31	1,02	0,47
5X4	0,66	0,81	0,86	1,24	1,06	0,80	0,62
5X5	1,19	1,39	1,45	1,69	1,98	1,71	0,78
10X2	2,92	3,68	2,87	120,66	163,63	2,89	1,78
10X3	3,98	3,63	4,10	204,86	232,84	4,28	3,83
10X4	4,29	4,53	4,91	260,22	334,22	7,03	3,99
10X5	6,14	6,52	6,23	182,51	285,71	10,32	5,31
15X2	8,36	10,74	12,06	6000,40	6000,36	13,26	7,53
15X3	34,09	40,74	47,96	6000,48	6000,44	77,74	19,87
15X4	157,81	345,86	301,72	6000,46	6000,45	369,99	64,00
15X5	280,30	389,89	478,85	6000,53	6000,46	619,24	110,19
20X2	23,10	47,36	38,69	6000,72	6000,60	100,78	21,03
20X3	321,05	1095,20	1005,01	6000,77	6000,68	1902,22	206,05
20X4	2555,91	4384,67	4323,53	6000,72	6000,67	4513,18	1794,04
20X5	5536,63	5991,50	6000,39	6000,78	6000,61	6000,72	4266,29
MEDIA	558,61	770,51	764,39	3.048,64	3.064,12	851,60	406,63

Tabla 15: Tiempos de computación en la función objetivo $\sum C_j$

Como se ve en la tabla, podemos apreciar los tiempos de proceso de todos los modelos para todos los bloques de instancias y vemos que al igual que se mencionó anteriormente, los modelos WST y TS3 son los mejores en este apartado y son los que menos tardan en procesar todas las instancias de media, recalcando que el modelo TS3 el ligeramente más rápido que el modelo WST.

Por último, en tercer lugar, se calculará el valor de ARPD (Average Relative Percentage Deviation) para todos los modelos para la función objetivo $\sum C_j$.

ARPD - FO: SumCj

	WST	WILSON	TS2	SGST	Lyeq	TBA	TS3
5X2	0	0	0	0	0	0	0
5X3	0	0	0	0	0	0	0
5X4	0	0	0	0	0	0	0
5X5	0	0	0	0	0	0	0
10X2	0	0	0	0	0	0	0
10X3	0	0	0	0	0	0	0
10X4	0	0	0	0	0	0	0
10X5	0	0	0	0	0	0	0
15X2	0	0	0	0,347	0,571	0	0
15X3	0	0	0	0,843	0,701	0	0
15X4	0	0	0	0,514	0,572	0	0
15X5	0	0	0	0,881	0,656	0	0
20X2	0	0	0	1,115	1,167	0	0
20X3	0	0	0	1,312	2,380	0	0
20X4	0	0,425	0,463	1,375	2,588	0,609	0
20X5	0,593	0,714	0,875	2,183	2,475	1,228	0,056

Tabla 16: Tabla de ARPDs de la función objetivo $\sum C_j$

Como vemos en la tabla y teniendo en cuenta los análisis anteriores, se puede comprobar y deducir fácilmente que en las primeras instancias de datos todos los modelos encuentran el óptimo y, por lo tanto, su ARPD es cero. No es hasta que se llega a los 15 trabajos y 2 máquinas cuando vemos que los modelo SGST y LYeq no encuentran el óptimo y comparado con los otros modelos que sí lo han hecho, su ARPD vale 0.347% y 0.571% para el caso de $\sum C_j$. Cabe resaltar, que el mayor ARPD, se encuentra al igual que para la función objetivo anterior para el bloque de 20 trabajos y 4 máquinas siendo su valor de 2.588%.

7. Conclusión

Finalmente, para concluir este estudio, hemos podido comprobar que se ha sido capaz de adaptar y resolver todos los modelos para dos problemas tradicionales de la programación de la producción con dos funciones objetivo cada uno y que, a raíz de analizar todos los resultados de la evaluación computacional que incluye un análisis de resultados, un análisis de los tiempos de proceso y un análisis de los valores de las funciones objetivo, los modelos más eficientes son dos modelos pertenecientes a la familia Wagner. El primero de ellos, se trata del modelo WST, que se llama así ya que sus autores se llaman Wagner, Stafford y Tseng; y el segundo modelo, es el modelo TS3, cuyo nombre viene de los autores Tseng y Stafford y que cabe resaltar que este modelo es ligeramente mejor que el modelo WST.

Como hecho recalcable, se puede ver que los modelos con menos número de restricciones y variables son más rápidos a la hora de analizar todas las instancias como es lógico y por ello, son capaces de llegar o encontrar el óptimo del problema sin llegar al límite de tiempo. Ocurre completamente lo contrario a los modelos con más restricciones o variables como son los de la familia Manne (los modelos SGST y LYeq) que a la hora de analizar grandes modelos con muchas máquinas y muchos trabajos llegan a tener un tiempo de proceso quizás excesivo en algunos casos.

Además, se ha visto como estos modelos de flowshop con permutación se han podido adaptar a la restricción no-wait y esto hace que estos nuevos modelos se puedan utilizar en el análisis de problemas de esta índole con el fin de organizar y optimizar programas de producción en los cuales los trabajos no puedan o deban esperar – restricción global no-wait – para establecer mejoras logísticas en los programas y planes de producción. Estos tipos de problemas, son muy comunes cuando se trata de mantener, por ejemplo, las cadenas de frío de los alimentos o los medicamentos, ya que estos productos precisan de unas condiciones de humedad y temperatura muy concretas durante todas sus etapas de control, producción o almacenamiento.

8. Bibliografía

[1] Framinan, J., M., Leisten R., Ruiz R. (2014). Manufacturing Scheduling Systems, Springer.

[2] Pinedo, M., L. (2011). Scheduling: Theory, Algorithms, and Systems, 4ed, Springer.

[3] Stafford Jr, E. F., Tseng, F. T., Gupta, J. N. D. (2005). Comparative evaluation of MILP flowshop models. Journal of the Operational Research Society, 56:88-101.

[4] Tseng, F. T., Stafford Jr., E. F. (2008). New MILP models for the permutation flowshop problema. Journal of the Operational Research Society, 59:1373-1386.

[5] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. Annals of Discrete Mathematics, 5:287–326.

[6] Formato LP en Gurobi. Disponible en: https://www.gurobi.com/documentation/8.0/refman/lp_format.html

[7] Formato BATCH para los archivos .bat. Disponible en: http://profesoremiliobarco.blogspot.com/2012/05/comando-for-para-archivos-bat.html

[8] Introducción a la programación entera. Disponible en: https://en.wikipedia.org/wiki/Integer_programming

[9] ¿Por qué es necesario un software de programación? Disponible en: https://blog.capterra.com/you-need-intelligent-scheduling-software/

9. Anexos

En este anexo, se detallan los códigos en C y el archivo BATCH utilizados en el desarrollo del estudio. Para cada modelo, se muestra el código general para la función objetivo Cmax o makespan y aparte, se muestra el código únicamente de la función objetivo $\sum C_j$ para no volver a escribir el resto del modelo ya que es el mismo, con lo que el modelo es el mismo y lo único que cambia son las dos funciones objetivo.

9.1. Familia Wagner

9.1.1. Modelo WST

9.1.1.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The WST model
void modeloWST(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The WST model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/WSTmodel_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
```

```
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloWST(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
return cont;
}
void modeloWST (int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,j,r,p;
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"T_%d_%d + ",m,i);
}
else
{
fprintf(fichero2,"T_%d_%d + ",m,i);
}
}
for (p=1;p<=n;p++)
{
if(p<n)
{
fprintf(fichero2,"X_%d_%d + ",m,p);
}
else
{
```

```
fprintf(fichero2,"X_%d_%d \n",m,p);
}
}
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
//fprintf(fichero2,"Primera restriccion \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if (j<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (r=1;r<=m-1;r++)
```

```
{
for (j=1;j<=n-1;j++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r,i,i,j+1);
}
fprintf(fichero2,"X_%d_%d - ",r,j+1);
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - ",r+1,i,i,j);
}
fprintf(fichero2,"X_%d_%d = 0 \n",r+1,j+1);
}
}
//fprintf(fichero2,"Cuarta restriccion \n");
for (r=1;r<=m-1;r++)
{
fprintf(fichero2,"X_%d_1 - ",r+1);
fprintf(fichero2,"X_%d_1 - ",r);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_1 ] - ",r,i,i);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_1 ] = 0 \n",r,i,i);
}
}
}
//Tiempos de proceso
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
```

```
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
}
}
fprintf(fichero2,"X_1_1 = 0 \n");
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (i=1;i<=m-1;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"X_%d_%d >= 0 \n",i,j);
}
}
//VARIABLES
fprintf(fichero2,"Generals \n");
for (i=1;i<=m-1;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"X_%d_%d \n",i,j);
}
}
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Z_%d_%d \n",i,j);
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.1.1.2.FO: $\sum C_j$

```
//fprintf(fichero2,"Funcion objetivo SUMCj \n");
fprintf(fichero2,"Minimize \n");
for (j=1;j<=n;j++)
```

```
{
fprintf(fichero2,"%d X_%d_%d + ",n-j+1,m,j);
}
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i==n && j==n)
{
fprintf(fichero2,"[ %d T_%d_%d * Z_%d_%d ] \n",n-j+1,m,i,i,j);
}
else
{
fprintf(fichero2,"[ %d T_%d_%d * Z_%d_%d ] + ",n-j+1,m,i,i,j);
}
}
}
```

9.1.2. Modelo Wilson

9.1.2.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The Wilson model
void modeloWilson (int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The Wilson model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
```

```
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/Wilsonmodel_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloWilson(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
return cont;
}
void modeloWilson(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,j,r;
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
fprintf(fichero2,"B_%d_%d + ",m,n);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",m,i,i,n);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] \n",m,i,i,n);
}
```

```
}
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
//fprintf(fichero2,"Primera restriccion \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if (j<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (j=1;j<=n-1;j++)
{
fprintf(fichero2,"B_1_%d + ",j);
```

```
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_1_%d * Z_%d_%d ] + ",i,i,j);
}
else
{
fprintf(fichero2,"[ T_1_%d * Z_%d_%d ] - ",i,i,j);
}
}
fprintf(fichero2,"B_1_%d <= 0 \n",j+1);
}
//fprintf(fichero2,"Cuarta restriccion \n");
fprintf(fichero2,"B_1_1 = 0 \n");
//fprintf(fichero2,"Quinta restriccion \n");
for (r=1;r<=m-1;r++)
{
fprintf(fichero2,"B_%d_1 + ",r);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_1 ] + ",r,i,i);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_1 ] - ",r,i,i);
}
}
fprintf(fichero2,"B_%d_1 = 0 \n",r+1);
}
//fprintf(fichero2,"Sexta restriccion \n");
for (r=1;r<=m-1;r++)
{
for (j=2;j<=n;j++)
{
```

```
fprintf(fichero2,"B_%d_%d + ",r,j);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r,i,i,j);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - ",r,i,i,j);
}
}
fprintf(fichero2,"B_%d_%d = 0 \n",r+1,j);
}
}
//fprintf(fichero2,"Séptima restriccion \n");
for (r=2;r<=m;r++)
{
for (j=1;j<=n-1;j++)
{
fprintf(fichero2,"B_%d_%d + ",r,j);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r,i,i,j);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - ",r,i,i,j);
}
}
fprintf(fichero2,"B_%d_%d <= 0 \n",r,j+1);
}
}
//Tiempos de proceso
for (r=1;r<=m;r++)
```

```
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
}
}
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (r=1;r<=m;r++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"B_%d_%d > 0 \n",r,j);
}
}
//VARIABLES
fprintf(fichero2,"Generals \n");
for (r=1;r<=m;r++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"B_%d_%d \n",r,j);
}
}
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Z_%d_%d \n",i,j);
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.1.2.2. FO: $\sum C_j$

```
//fprintf(fichero2,"Funcion objetivo SUMCj \n");
fprintf(fichero2,"Minimize \n");
for (j=1;j<=n;j++)
{
fprintf(fichero2,"B_%d_%d + ",m,j);
for (i=1;i<=n;i++)
{
if (i==n && j==n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] \n",m,i,i,j);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",m,i,i,j);
}
}
}
```

9.1.3. Modelo TS2

9.1.3.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The TS2 model
void modeloTS2(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The TS2 model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
```

```
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/TS2model_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloTS2(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
return cont;
}
void modeloTS2(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,j,r;
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
fprintf(fichero2,"E_%d_%d \n",m,n);
//RESTRICCIONES DEL MODELO
//fprintf(fichero2,"Primera restriccion \n");
fprintf(fichero2,"Subject To \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if (j<n)
{
```

```
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (r=1;r<=m;r++)
{
for (j=1;j<=n-1;j++)
{
fprintf(fichero2,"E_%d_%d + ",r,j);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r,i,i,j+1);
}
else
{
```

```
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] ",r,i,i,j+1);
}
}
fprintf(fichero2,"- E_%d_%d <= 0 \n",r,j+1);
}
}
//fprintf(fichero2,"Cuarta restriccion \n");
for (r=1;r<=m-1;r++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"E_%d_%d + ",r,j);
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r+1,i,i,j);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] ",r+1,i,i,j);
}
}
fprintf(fichero2,"- E_%d_%d = 0 \n",r+1,j);
}
}
//fprintf(fichero2,"Quinta restriccion \n");
fprintf(fichero2,"E_1_1 ");
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"- [ T_1_%d * Z_%d_1 ] ",i,i);
}
else
{
fprintf(fichero2,"- [ T_1_%d * Z_%d_1 ] >= 0 \n",i,i);
```

```
}
}
//Tiempos de proceso
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
}
}
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (r=1;r<=m;r++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"E_%d_%d >= 0 \n",r,j);
}
}
//VARIABLES
fprintf(fichero2,"Generals \n");
for (r=1;r<=m;r++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"E_%d_%d \n",r,j);
}
}
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Z_%d_%d \n",i,j);
}
}
//FIN
```

```
fprintf(fichero2,"End");
}
```

9.1.3.2. FO: $\sum C_j$

```
//fprintf(fichero2,"Funcion objetivo SUMCj \n");
fprintf(fichero2,"Minimize \n");
for (i=1;i<=n;i++)
{
if (i==n)
{
fprintf(fichero2,"E_%d_%d \n",m,i);
}
else
{
fprintf(fichero2,"E_%d_%d + ",m,i);
}
}
```

9.1.4. Modelo TBA

9.1.4.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The TBA model
void modeloTBA(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The TBA model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
```

```
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/TBAmodel_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloTBA(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
return cont;
}
void modeloTBA(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,j,p,q,r,s;
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
for (p=1;p<=m-1;p++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_1 ] + ",p,i,i);
}
}
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d + ",m,i);
```

```
}
for (s=2;s<=n;s++)
{
if (s<n)
{
fprintf(fichero2,"X_%d_%d + ",m,s);
}
else
{
fprintf(fichero2,"X_%d_%d \n",m,s);
}
}
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
//fprintf(fichero2,"Primera restriccion \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if (j<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
```

```
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (r=2;r<=m;r++)
{
for (j=2;j<=n;j++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_1 ] + ",r-1,i,i);
}
for (q=1;q<=j-1;q++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - [ T_%d_%d * Z_%d_%d ]
+ ",r,i,i,q,r-1,i,i,q);
}
}
for (s=2;s<=j;s++)
{
if (s<j)
{
fprintf(fichero2,"X_%d_%d - ",r,s);
fprintf(fichero2,"X_%d_%d + ",r-1,s);
}
else
{
fprintf(fichero2,"X_%d_%d - ",r,s);
fprintf(fichero2,"X_%d_%d - ",r-1,s);
}
}
```

```
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - ",r-1,i,i,j);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] = 0 \n",r-1,i,i,j);
}
}
}
}
//fprintf(fichero2,"Cuarta restriccion \n");
{
fprintf(fichero2,"X_1_1 = 0 \n");
}
//Tiempos de proceso
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
}
}
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (r=1;r<=m;r++)
{
for (s=2;s<=n;s++)
{
fprintf(fichero2,"X_%d_%d >= 0 \n",r,s);
}
}
//VARIABLES
fprintf(fichero2,"Generals \n");
for (r=1;r<=m;r++)
```

```
{
for (s=2;s<=n;s++)
{
fprintf(fichero2,"X_%d_%d \n",r,s);
}
}
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Z_%d_%d \n",i,j);
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.1.4.2. FO: $\sum C_j$

```
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo SumCj \n");
fprintf(fichero2,"Minimize \n");
for (j=1;j<=n;j++)
{
fprintf(fichero2,"%d X_%d_%d + ",s,m,j);
s--;
}
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i==n && j==n)
{
fprintf(fichero2,"[ %d T_%d_%d * Z_%d_%d ] \n",c,m,i,i,j);
}
else
{
```

```
fprintf(fichero2,"[ %d T_%d_%d * Z_%d_%d ] + ",c,m,i,i,j);
}
}
c--;
}
```

9.1.5. Modelo TS3

9.1.5.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The TS3 model
void modeloTS3(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The TS3 model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/TS3model_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloTS3(n,m,Tri,fichero2);
fclose(fichero);
```

```
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
return cont;
}
void modeloTS3(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,j,r,p,q;
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
for (p=1;p<=n;p++)
{
fprintf(fichero2,"T_1_%d + ",p);
}
for (q=2;q<=m;q++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",q,i,i,n);
}
}
for (q=2;q<=n;q++)
{
if (q<n)
{
fprintf(fichero2,"Y_%d + ",q);
}
else
{
fprintf(fichero2,"Y_%d \n",q);
}
}
```

```
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
//fprintf(fichero2,"Primera restriccion \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if (j<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (r=2;r<=m;r++)
{
for (j=2;j<=n;j++)
{
```

```
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_1_%d * Z_%d_%d ] + ",i,i,j-1);
}
else
{
fprintf(fichero2,"[ T_1_%d * Z_%d_%d ] - ",i,i,j-1);
}
}
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - ",r,i,i,j-1);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r,i,i,j-1);
}
}
for (q=1;q<=r-1;q++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ Z_%d_%d * T_%d_%d ] - [ Z_%d_%d * T_%d_%d ]
+ ",i,j,q,i,i,j-1,q,i);
}
}
for (q=2;q<=j;q++)
{
if (q<j)
{
fprintf(fichero2,"Y_%d - Y_%d + ",q,q-1);
}
else
```

```
{
fprintf(fichero2,"Y_%d - Y_%d >= 0 \n",q,q-1);
}
}
}
}
//Tiempos de proceso
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
}
}
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Y_%d >= 0 \n",j);
}
//VARIABLES
fprintf(fichero2,"Generals \n");
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Y_%d \n",j);
}
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Z_%d_%d \n",i,j);
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.1.5.2. FO: $\sum C_j$

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The TS3 model 2
void modeloTS3_2(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The TS3 model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos2/TS3model2_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloTS3_2(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
```

```
return cont;
}
void modeloTS3_2(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,j,r,p,q;
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo SumCj \n");
fprintf(fichero2,"Minimize \n");
for (q=2;q<=m;q++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d + ",q,i);
}
}
for (q=2,j=2;q<=n && j<=n;q++,j++)
{
{
fprintf(fichero2,"%d Y_%d + ",n-j+1,q);
}
}
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i==n && j==n)
{
fprintf(fichero2,"[ %d T_1_%d * Z_%d_%d ] \n",n-j+1,i,i,j);
}
else
{
fprintf(fichero2,"[ %d T_1_%d * Z_%d_%d ] + ",n-j+1,i,i,j);
}
}
}
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
```

```
//fprintf(fichero2,"Primera restriccion \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
if (j<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (j=1;j<=n;j++)
{
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"Z_%d_%d + ",i,j);
}
else
{
fprintf(fichero2,"Z_%d_%d = 1 \n",i,j);
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (r=2;r<=m;r++)
{
for (j=2;j<=n;j++)
{
for (i=1;i<=n;i++)
{
```

```
if (i<n)
{
fprintf(fichero2,"[ T_1_%d * Z_%d_%d ] + ",i,i,j-1);
}
else
{
fprintf(fichero2,"[ T_1_%d * Z_%d_%d ] - ",i,i,j-1);
}
}
for (i=1;i<=n;i++)
{
if (i<n)
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] - ",r,i,i,j-1);
}
else
{
fprintf(fichero2,"[ T_%d_%d * Z_%d_%d ] + ",r,i,i,j-1);
}
}
for (q=1;q<=r-1;q++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"[ Z_%d_%d * T_%d_%d ] - [ Z_%d_%d * T_%d_%d ]
+ ",i,j,q,i,i,j-1,q,i);
}
}
for (q=2;q<=j;q++)
{
if (q<j)
{
fprintf(fichero2,"Y_%d - Y_%d + ",q,q-1);
}
else
{
fprintf(fichero2,"Y_%d - Y_%d >= 0 \n",q,q-1);
```

```
}
}
}
}
//Tiempos de proceso
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
}
}
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Y_%d >= 0 \n",j);
}
//VARIABLES
fprintf(fichero2,"Generals \n");
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Y_%d \n",j);
}
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (j=1;j<=n;j++)
{
fprintf(fichero2,"Z_%d_%d \n",i,j);
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.2. Familia Manne

9.2.1. Modelo SGST

9.2.1.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The SGST model
void modeloSGST(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The SGST model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/SGSTmodel_%d_%d_%d.lp",j,k,i);
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloSGST(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
```

```
}
return cont;
}
void modeloSGST(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int i,k,r,s;
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
fprintf(fichero2,"CMAX \n");
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
//fprintf(fichero2,"Primera restriccion \n");
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_1_%d - T_1_%d >= 0 \n",i,i);
}
//fprintf(fichero2,"Segunda restriccion \n");
for (r=1;r<=m-1;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_%d_%d - C_%d_%d - T_%d_%d = 0
\n",r+1,i,r,i,r+1,i);
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"C_%d_%d - C_%d_%d + [ P * D_%d_%d ] - T_%d_%d
>= 0 \n",r,i,r,k,i,k,r,i);
```

```
}
}
}
}
//fprintf(fichero2,"Cuarta restriccion \n");
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"C_%d_%d - C_%d_%d + P - [ P * D_%d_%d ] -
T_%d_%d >= 0 \n",r,k,r,i,i,k,r,k);
}
}
}
}
//fprintf(fichero2,"Quinta restriccion \n");
for (i=1;i<=n;i++)
{
fprintf(fichero2,"CMAX - C_%d_%d >= 0 \n",m,i);
}
//Tiempos de proceso
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
s=s+Tri[r-1][i-1];
}
}
fprintf(fichero2,"P = %d \n",s);
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (r=1;r<=m;r++)
```

```
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_%d_%d >= 0 \n",r,i);
}
}
fprintf(fichero2,"CMAX >= 0 \n");
//VARIABLES
fprintf(fichero2,"Generals \n");
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_%d_%d \n",r,i);
}
}
fprintf(fichero2,"CMAX \n");
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (k=1;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"D_%d_%d \n",i,k);
}
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.2.1.2. FO: $\sum C_j$

```
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo SUMCj \n");
fprintf(fichero2,"Minimize \n");
for (i=1;i<=n;i++)
```

```
{
if (i==n)
{
fprintf(fichero2,"C_%d_%d \n",m,i);
}
else
{
fprintf(fichero2,"C_%d_%d + ",m,i);
}
}
```

9.2.2. Modelo LYeq

9.2.2.1. Modelo general y FO: Cmax

```
#include <stdio.h>
#include <stdlib.h>
#include <schedule_lib.h>
//The LYeq model
void modeloLYeq(int n, int m, MAT_INT Tri, FILE *fichero2);
int main()
{
printf("The LYeq model\n");
int i,j,k,m,n;
int cont=0;
char cadena[50]={0};
char cadena2[50]={0};
for (j=5;j<=20;)
{
for (k=2;k<=5;k++)
{
for (i=1;i<=10;i++)
{
sprintf(cadena,"Bateria Sin fechas/inst_%d_%d_p_%d",j,k,i);
FILE * fichero = fopen(cadena,"r");
fscanf(fichero,"%d %d",&m,&n);
printf("El numero de maquinas es: %d \n",m);
printf("El numero de trabajos es: %d \n",n);
sprintf(cadena2,"modelos/LYeqmodel_%d_%d_%d.lp",j,k,i);
```

```
FILE * fichero2=fopen(cadena2,"w");
MAT_INT Tri = DIM_MAT_INT(m,n);
Tri=loadPTimes_nrows(cadena,&n,&m,YES);
modeloLYeq(n,m,Tri,fichero2);
fclose(fichero);
fclose(fichero2);
fflush(stdin);
cont=cont+1;
}
}
j=j+5;
}
return cont;
}
void modeloLYeq(int n, int m, MAT_INT Tri, FILE *fichero2)
{
int r,i,k,s,j;
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo CMAX \n");
fprintf(fichero2,"Minimize \n");
fprintf(fichero2,"CMAX \n");
//RESTRICCIONES DEL MODELO
fprintf(fichero2,"Subject To \n");
//fprintf(fichero2,"Primera restriccion \n");
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"[ P * D_%d_%d ] + C_%d_%d - C_%d_%d - T_%d_%d
- Q_%d_%d_%d = 0 \n",i,k,r,i
,r,k,r,i,r,i,k);
}
}
```

```
}
}
//fprintf(fichero2,"Segunda restriccion \n");
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"Q_%d_%d_%d - P + T_%d_%d + T_%d_%d <= 0
\n",r,i,k,r,i,r,k);
}
}
}
}
//fprintf(fichero2,"Tercera restriccion \n");
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_1_%d - T_1_%d >= 0 \n",i,i);
}
//fprintf(fichero2,"Cuarta restriccion \n");
for (r=1;r<=m-1;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_%d_%d - C_%d_%d - T_%d_%d = 0
\n",r+1,i,r,i,r+1,i);
}
}
//fprintf(fichero2,"Quinta restriccion \n");
for (i=1;i<=n;i++)
{
fprintf(fichero2,"CMAX - C_%d_%d >= 0 \n",m,i);
}
//Tiempos de proceso
```

```
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"T_%d_%d = %d \n",r,i,Tri[r-1][i-1]);
s=s+Tri[r-1][i-1];
}
}
fprintf(fichero2,"P = %d \n",s);
//ACOTACIONES
fprintf(fichero2,"Bounds \n");
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
fprintf(fichero2,"C_%d_%d >= 0 \n",r,i);
}
}
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"Q_%d_%d_%d >= 0 \n",r,i,k);
}
}
}
}
fprintf(fichero2,"CMAX >= 0 \n");
//VARIABLES
fprintf(fichero2,"Generals \n");
for (i=1;i<=m;i++)
{
for (j=1;j<=n;j++)
```

```
{
fprintf(fichero2,"C_%d_%d \n",i,j);
}
}
for (r=1;r<=m;r++)
{
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"Q_%d_%d_%d \n",r,i,k);
}
}
}
}
fprintf(fichero2,"CMAX \n");
fprintf(fichero2,"Binary \n");
for (i=1;i<=n;i++)
{
for (k=2;k<=n;k++)
{
if (k>i)
{
fprintf(fichero2,"D_%d_%d \n",i,k);
}
}
}
//FIN
fprintf(fichero2,"End");
}
```

9.2.2.2. FO: $\sum C_j$

```
//FUNCION OBJETIVO
//fprintf(fichero2,"Funcion objetivo SUMCj \n");
fprintf(fichero2,"Minimize \n");
```

```
for (i=1;i<=n;i++)
{
if (i==n)
{
fprintf(fichero2,"C_%d_%d \n",m,i);
}
else
{
fprintf(fichero2,"C_%d_%d + ",m,i);
}
}
```

9.3. Archivo BATCH

```
FOR /L %%x IN (5,5,20) DO (
FOR /L %%y IN (2,1,5) DO (
FOR /L %%z IN (1,1,10) DO (
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\WSTmodel_%%x_%%y_%%z.lp
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\Wilsonmodel_%%x_%%y_%%z.lp
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\TS2model_%%x_%%y_%%z.lp
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\SGSTmodel_%%x_%%y_%%z.lp
```

```
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\LYeqmodel_%%x_%%y_%%z.lp
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\TBAmodel_%%x_%%y_%%z.lp
gurobi_cl TimeLimit=600
LogFile=C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_lo
g1\Debug\gurobi.log
C:\Users\Usuario\Dropbox\TFG\EjecutableLectura_gurobi_log\Debug\
modelos\TS3model_%%x_%%y_%%z.lp
)
)
)
```