Trabajo Fin de Grado
Ingeniería de Organización Industrial

Resolución exacta de problemas de programación operacional de trabajos en intervalos

Autor: Pedro Giménez Molina
Tutor: José Manuel García Sánchez

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

Sevilla, 2021

Trabajo Fin de Grado
Ingeniería de Organización Industrial

Resolución exacta de problemas de programación operacional de trabajos en intervalos

Autor:
Pedro Giménez Molina

Tutor:
José Manuel García Sánchez
Profesor titular:

Dpto. de Organización industrial y gestión de empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de Carrera: Resolución exacta de problemas de programación operacional de trabajos en intervalos

Autor: Pedro Giménez Molina

Tutor: José Manuel García Sánchez

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

Resumen

En este trabajo estudiamos de forma general la programación de trabajos en intervalos, centrándonos en los problemas de programación de trabajos en intervalos variables. En su inicio se desarrolla una visión más teórica de este tipo de problemas para, a continuación, adentrarse en la experimentación con tres modelos, los cuales se verán primero de forma matemática, para luego implementarlos a lenguaje Lingo. Se incluye, además, un algoritmo y su implementación en Python para extraer los datos del modelo desde una batería de problemas. Por último, a partir de los tiempos de ejecución de los problemas generados, realizaremos una comparación entre los comportamientos de los tres modelos estudiados frente a los problemas ejecutados para terminar encontrando el más adecuado para resolverlos.

Abstract

In this work we study in a general way the scheduling of jobs in intervals, focusing on the problems of scheduling jobs in variable intervals. At the beginning, a more theoretical vision of this type of problems is developed, then, to enter into the experimentation with three models, which will be seen first in a mathematical way, and then implement them in the Lingo language. It also includes an algorithm and its implementation in Python to extract the model data from a battery of problems. Finally, based on the execution times of the problems generated, a comparison will be made between the behaviors of the three models studied in relation to the problems executed to end up finding the most suitable one to solve them.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

La optimización de procesos de cualquier sector productivo puede ser llevada a cabo resolviendo un conjunto de problemas conocidos como problemas de programación de trabajos en intervalos. Esto permite determinar la mejor formar de organizar las tareas o trabajos, ya sea dentro de sistemas pequeños como un centro de logística o abarcando la totalidad del funcionamiento de una empresa, teniendo como meta una mejoría económica.

Dentro de esta categoría de problemas, existen numerosas variantes como el Fixed Job Scheduling, en el cual los trabajos deberán ser realizados en intervalos de tiempo que coincidan de forma exacta con la duración de cada trabajo.

Sin embargo, la variante que estudiaremos en este proyecto será el Variable Job Scheduling Problem, donde la extensión en el tiempo de cada trabajo no coincidirá con la del intervalo donde se realice, será menor. Por lo tanto, existirán inevitablemente holguras entre la realización de cada trabajo.

Este problema puede ser enfocado de diferentes formas. Por un lado, desde un punto de vista operacional, asignando los recursos (máquinas) que estén disponibles a los trabajos a realizar siguiendo un criterio de optimización, o desde un punto de vista táctico, en el que se buscará calcular la capacidad necesaria para la consecución de todos los trabajos pendientes. En el proyecto tendremos como meta el objetivo operacional.

Partiremos de tres modelos matemáticos. En ellos, el criterio de optimización será maximizar el peso de los trabajos procesados. Estudiaremos los elementos, las actividades de decisión, las especificaciones y las funciones objetivo de cada modelo.

Realizaremos su implementación en las librerías de optimización Lingo, con el que podremos simular cada modelo a partir de los datos necesarios procedentes de una batería de problemas, los cuales representan diferentes valores en los parámetros del problema, aportando numerosos datos como el número de trabajos a procesar o el número de máquinas de las que se dispone para tratarlos.

Una vez obtenidos los datos de las simulaciones de los tres modelos, desarrollaremos un análisis de esos resultados y compararemos la eficiencia y la rapidez entre los modelos.

A continuación, explicaremos la estructura general de este trabajo, adjuntando una breve descripción de lo que veremos en cada capítulo:

- El capítulo 2 contendrá una visión fundamentalmente teórica sobre la programación de trabajos en intervalos y más concretamente sobre el Variable Job Scheduling Problem. Además, explicaremos de una forma más gráfica dos casos prácticos de esta variante.
- El capítulo 3 ahondará en las explicaciones de los modelos matemáticos: Cómo interactuarán los elementos que los componen entre ellos, qué activades de decisión se realizarán, cuál será el objetivo a maximizar o minimizar en sus funciones objetivo, o el significado que encierra cada una de las

especificaciones que limitarán cada modelo.

- El capítulo 4 tratará la implementación de los modelos a las librerías de optimización Lingo, pasando por una previa descripción de este procesador.
- El capítulo 5 se dedica a presentar la clasficación de los problemas que se encuentran dentro de la batería y los datos que podremos obtener de cada uno de ellos. También se incluye la explicación de cómo obtener los archivos que representan cada problema en el formato adecuado para Lingo a partir de un algoritmo desarrollado en lenguaje Python.
- En el capítulo 6, veremos la ejecución de los modelos en Lingo introduciendo los datos de los problemas de la batería, comparando los resultados en términos de eficiencia y tiempo entre ellos para evaluar cuál es el mejor para afrontar cada problema que se ejecute.
- Y, en el capítulo 7, expondremos una valoración global del trabajo y de los resultados obtenidos.

2 PROGRAMACIÓN DE TRABAJOS EN INTERVALOS

EN este capítulo ahondaremos en la programación de trabajos en intervalos en general y sus diferentes variantes. Además, de forma más explicita más veremos el problema Variable Job Scheduling Problem, objetivo fundamental de nuestro trabajo. También hemos desarrollado dos casos prácticos sobre este problema, uno con máquinas que procesen los mismos trabajos y otro con distintas máquinas capaces de soportar distintos trabajos cada una.

2.1. Programación de trabajos en intervalos

La programación de trabajos en intervalos surge de la necesidad organizativa de las empresas de conseguir realizar tanto sus productos como sus proyectos en la menor cantidad de tiempo posible sin superar los límites que tenga establecidos tanto la propia empresa como sus clientes. También se pretende ordenar que cada uno de esos productos se procese en los recursos que tenga disponible la empresa adecuados para ellos, ya sea en máquinas o asignándolos a equipos de trabajo determinados en el caso de los proyectos más extensos.

Respecto a situaciones reales en las que se aplique este procedimiento tenemos como ejemplos la distribución de piezas que formen parte de un conjunto ensamblable en las diferentes máquinas de un taller tipo job shop; el reparto de clases de diferentes asignaturas en una universidad en las diferentes aulas disponibles sin que se rebase el tiempo disponible del que dispone el horario establecido de uso de cada aula; el proceso de fabricación y mantenimiento de las diferentes partes de un avión, el cual se va desplazando lentamente, consiguiendo así que los operarios acaben su parte del proceso de construcción en un tiempo determinado de forma extricta; el proceso de asignación de tareas a distintos equipos de trabajo dentro de una empresa, cada uno de ellos especializados en una tarea, cuyos posibles inicios dependan de que se hayan realizado anteriores tareas; el sistema de luces de los semáforos en una ciudad; el sistema de cita previa para un proceso de vacunación, distribuyendo a los pacientes en distintos centros según su edad y repartiéndolos en horarios lo suficientemente espaciados para no colapsar esos centros; la asignación de turistas a habitaciones libres de un hotel teniendo en cuenta el tiempo de su estancia; la distribución de conductores a coches disponibles y a zonas determinadas en una empresa de transporte privada,etc. [1,2]

Retomando el tema que nos ocupa, la programación de trabajos en intervalos en genenal consiste en adjudicar un conjunto de trabajos pendientes a un grupo de recursos, que desde ahora trataremos siempre como máquinas, en el menor tiempo posible, limitado por un horizonte temporal predeterminado de antemano. Esta limitación de tiempo se debe a que cada trabajo, en su descripción, tendrá asociado un intervalo de tiempo en el que deberá comenzar. En el caso de que el trabajo se programe fuera de ese intervalo, generará la prohibición de que ese trabajo se programe en cualquier otro momento fuera del intervalo, lo que derivará en costes importantes al responsable del proyecto o directamente a su empresario supervisor.

Cada intervalo de cada trabajo estará delimitado por el instante de posible inicio más temprano y el instante de

posible inicio más tardío. Si el trabajo se comienza a procesar fuera de esos límites, ya no será valido.

La otra característica de los trabajos, presente siempre en todas las variantes del problema, es su duración. Si el trabajo se empieza a realizar correctamente dentro de su intervalo asignado, lo que dure a partir de ese instante ya no será importante. Sin embargo, sí influirá respecto al número de recursos disponibles. Si los tiempos de duración de los trabajos que se encuentran en proceso son muy elevados y hay pocas máquinas con las que contar para acogerlos, puede ocurrir que no se cumplan los intervalos de los siguientes trabajos en la lista por no tener máquinas disponibles.

También contarán los trabajos con otras dos carácterísticas, aunque estas no serán fijas para todos los tipos de problemas en la programación de trabajos en intervalos. Por un lado, tenemos los pesos de cada trabajo, es decir, el valor que tiene para el empresario el que se realize correctamente. Siempre se intentará priorizar la consecución de los trabajos con mayor peso. Por otra parte, está la clase o tipo de cada trabajo. Existirán casos en los que todos los trabajos a procesar sean del mismo tipo y puedan desarrollarse de la misma forma, con lo que no importará esta característica, pero en su mayoría tendremos casos en los que los trabajos se dividan en diferentes clases. En consecuencia, se dispondrá de distintas máquinas capaces de recibir cada uno de estos tipos de trabajos.

A continuación, disponemos de una gráfica que clasifica las distintas características que pueden tener los problemas de programación de trabajos en intervalos, las cuales, combinadas entre sí de distintas formas, dan lugar a numeros tipos de problemas diferentes.

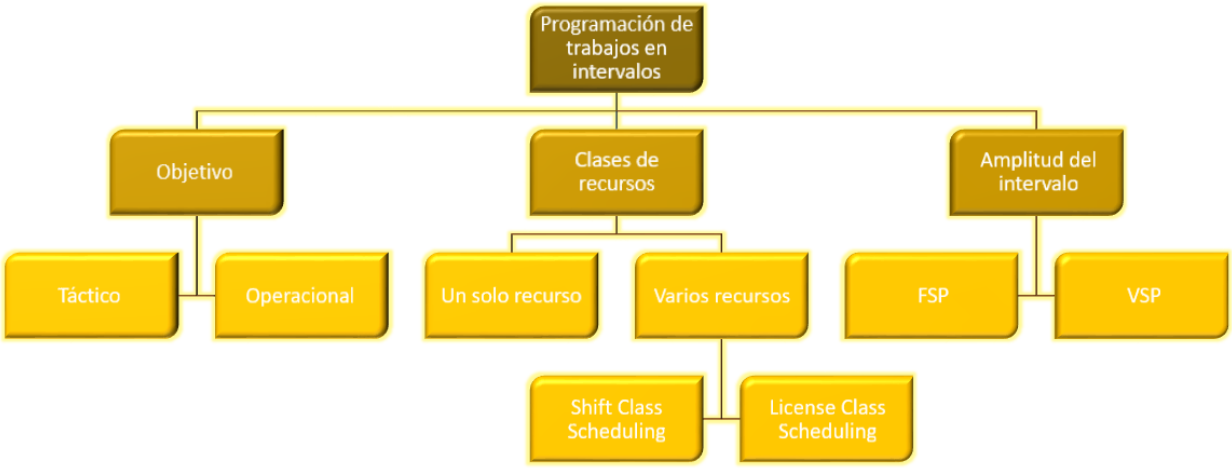

Figura 2-1. Clasificación de los problemas de programación de trabajos en intervalos

En la primera clasificación, se dividen los problemas según su objetivo, o lo que es lo mismo la solución última a la que se quiere llegar. Según la que se pretenda, cambiará la forma de resolver el problema. Contamos con dos tipos de objetivos principales:

- Objetivo táctico. Partimos de la idea de que no tenemos de antemano un número de recursos, máquinas, inalterable. A partir de ahí, buscaremos cuál sería el número de máquinas mínimo suficiente para procesar el conjunto de trabajos a cumplir dentro de sus intervalos señalados [1,3]. Esto se debe a que, para estos casos, el uso de cada máquina llevará asociado un coste que habrá que reducir. También se intentará utilizar más frecuentemente aquellas máquinas con costes de uso más bajo cuando sea posible.

  El hecho de tener cierta libertad en elegir el total de máquinas que queremos a nuestra disposición hará que, a la hora de programar los trabajos, lo hagamos siempre para casos en los que podemos programar todos ellos, aunque el coste del uso de las máquinas aumente.

- Objetivo operacional. Con este objetivo, sin embargo, el problema estará restringido desde el comienzo a un número de máquinas fijo. De esta manera, se genera un cierto grado de libertad para resolver el problema, aunque la forma más usada para hacerlo es teniendo en cuenta la característica del peso de

los trabajos. La meta será maximizar el peso total de los trabajos que se realicen. [1,3]

De esto último se deduce que, al contrario que con el objetivo táctico, ahora si puede existir la posibilidad de que alguno de los trabajos pendientes no se programe al tener fijado el número de máquinas disponible.

Estas dos formas de resolver los problemas, aunque lo pueda parecer, no son del todo incompatibles. Pueden aparecer problemas que tengan rasgos de ambos y podrán resolverse igualmente. [1,3]

La siguiente clasificación propuesta va en función de cuántas clases de máquinas tengamos disponibles. Existirá el caso en el que contemos con un único tipo de máquinas, provocando que todos los trabajos a realizar puedan acceder a cualquiera de esas máquinas. Es un caso sencillo, pero a la vez poco común. Es más corriente encontrarnos con problemas en los que tengamos varios grupos de máquinas cada uno de una clase distinta. Para estos tipos de problemas entrarán en juego la característica clase de cada trabajo. Estará establecida una compatibilidad previa entre determinadas clases de trabajos y máquinas. Cada trabajo irá únicamente a una máquina que sea aceptable para él.

Los problemas que contengan máquinas de varias clases, a su vez estarán divididos en otras dos categorías:

- Shift Class Scheduling. En esta ocasión, las máquinas tendrán una carácteristica extra, tendrán asociados intervalos de disponibilidad temporal. Esto quiere decir que, en determinados instantes de tiempo durante el procesamiento de los trabajos, las máquinas podrán estar bloqueadas según sus tiempos de descanso o indisponibilidad [1,3]. Los trabajos que lleguen a estas máquinas en esos intervalos deberán ser desviados a otras máquinas o introducidos en almacenes de espera dentro del proceso.

  Como ejemplo de este caso tenemos un horno, el cuál mientras no alcance las temperaturas adecuadas no podrá recibir unidades productivas.

- License Class Scheduling. Esta variante considera que todas las máquinas del proceso se encuentran en funcionamiento en todo momento, en definitiva, no estarán restringidas por el tiempo. De modo que su única limitación será el que sean o no compatibles con los trabajos a procesar. [1,3]

Para finalizar, la última forma de clasificar los problemas será según el tipo de intervalo que tengan sus trabajos:

- Fixed Job Scheduling Problem (FSP). En ellos, la duración de cada uno de los trabajos concuerda exactamente en el tiempo con cada uno de sus respectivos intervalos [1,4]. En la siguiente imagen tenemos una representación gráfica de un trabajo en concreto, i, dentro de su intervalo definido por los límites I y F, instantes de inicio y de finalización obligatorios respectivamente. Al ser un FSP, la duración del trabajo i, d, encaja de lleno en la duración permitida por el intervalo.

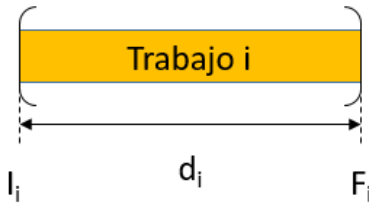

Figura 2-2. Fixed Job Scheduling Problem (FSP)

- Variable Job Scheduling Problem (VSP). Para este tipo de problemas, las duraciones de los trabajos no tendrán por qué coincidir con la longitud de sus intervalos preasignados. Esto supone que siempre tendremos duraciones menores o iguales que las extensiones de sus respectivos trabajos. El hecho de que también puedan coincidir hace posible considerar que los problemas VSP abarcan de alguna forma también a los problemas FSP [1,4].

  De nuevo, hemos realizado otro ejemplo gráfico para representar cómo se vería un intervalo en estos problemas. Ahora la duración d del trabajo i no abarca por completo la extension del intervalo. Existen holguras tanto antes como después del tiempo en el que se realiza el trabajo, por lo que hay más libertad

para procesarlo en un momento u otro.

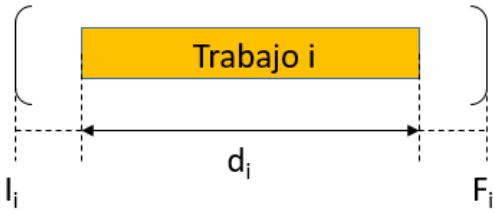

Figura 2-3. Variable Job Scheduling Problem (VSP)

Según los científicos Arkin y Silverberg, los problemas FSP pueden considerarse por lo general de tipo NP-complejo hasta el momento en el que cada trabajo individualmente, se pueda procesar en 3 o más de las máquinas disponibles. Por otra parte, los problemas VSP siempre serán tratados como problemas NP-complejo. [1,2]

A lo largo de los siguientes capítulos, los problemas que estudiaremos tendrán las mismas características. Partiremos de un objetivo operacional, en concreto el definido antes como el más común, maximizar el sumatorio de los pesos cuyos trabajos son finalmente realizados en el proceso. Tendremos máquinas y trabajos con diferentes clases y, por tanto, con posibles disponibilidades entre ellos. Consideraremos que todas las máquinas con las que dispongamos, estarán listas para procesar trabajos en todo momento, o lo que es lo mismo, seguiremos la variante License Class Scheduling. Además, todos los problemas serán del tipo VSP, es decir, la duración de cada trabajo no tendrá por qué coincidir en el tiempo exacto que proporciona el intervalo de posibles inicios de cada uno.

Como consideraciones externas a las vistas en este capítulo tendremos que cada máquina que forme parte del sistema no tendrá la capacidad de tratar más de un trabajo a la vez y que, en el instante en el que se empieza a procesar un trabajo en el interior de una máquina, el tratamiento no podrá ser interrumpido ni para remplazarlo por otro trabajo, ni pausado para terminar el resto de su proceso más adelante.

2.2. Casos prácticos

Para facilitar la major comprensión de este tipo de problemas, hemos resuelto manualmente, de forma gráfica, dos casos en los que hay que optimizar de acuerdo a las condiciones de un problema VSP. En el primero, los elementos del problema serán iguales y, en el segundo, las máquinas solo podrán realizer determinados trabajos.

2.1.1 Con máquinas sin clases

En este primer caso, partimos de una situacíon determinada en la que se tienen que procesar 12 trabajos, numerados en la columna TR, cada uno con su correspondiente duración (di) y sus delimitaciones de inicio temprana (Ii) y tardía (Fi). Los datos pertenencientes a estos tres parámetros coinciden con la cuarta, segunda y tercera columnas de la siguiente tabla, respectivamente.

Por otro lado, suponemos que contamos con dos recursos o máquinas (R1 y R2), ambos con las mismas características. Cualquiera de los trabajos se podrá hacer en una u otra sin distinción.

En la cuarta columna de la tabla aparece un diagrama de Gantt en el que vienen representados los intervalos donde se podrá realizar cada trabajo (casillas blancas), cuya longitud dependerá de las condiciones de inicio ai y bi. Dentro de ellos, las casillas verdes marcan en qué momento del intervalo se ha terminado realizando el trabajo.

El eje de abcisas del diagrama marca el tiempo y el eje de ordenadas relaciona cada trabajo con su intervalo. Al final de cada fila aparece el recurso encargado de operar cada trabajo (R1 o R2). Como el objetivo que tendremos a lo largo del Proyecto es operacional y no táctico, el número de recursos disponibles siempre estará prefijado. Esto puede generar un problema grave que he representado en este caso.

TR	l_i	F_i	d_i
1	2	4	3
2	3	7	4
3	5	6	2
4	6	9	5
5	7	9	3
6	8	10	5
7	8	12	4
8	13	14	2
9	16	17	3
10	15	17	4
11	16	18	5
12	19	21	4

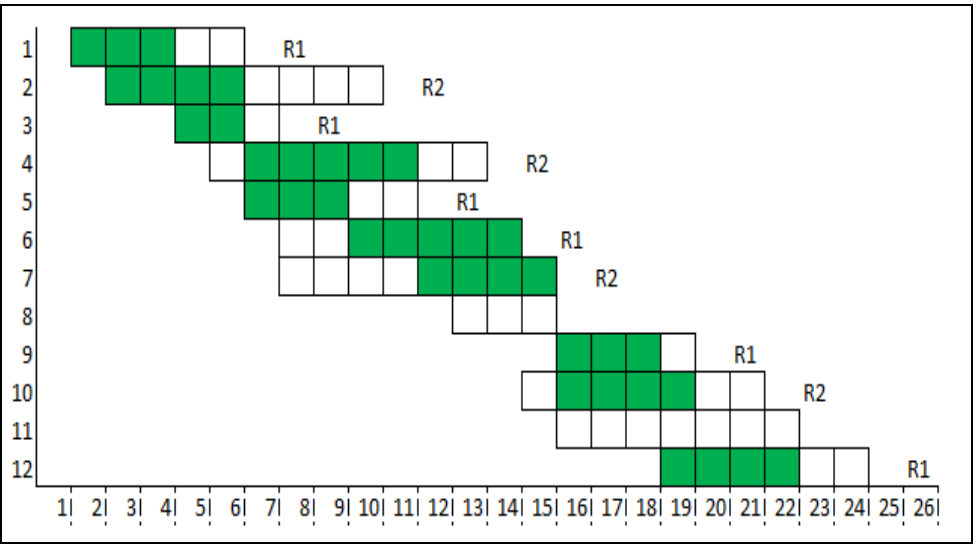

Tabla 2-1. Caso con máquinas sin clases

El trabajo 8 no puede comenzar a ser realizado dentro de sus límites prefijados, entre 13 y 14, debido a que durante ese tiempo los recursos 1 y 2 siguen estando ocupados con los trabajos 6 y 7 respectivamente. Lo mismo ocurre con el trabajo 11, ya que tampoco hay recursos disponibles en su plazo de realización.

El hecho de que no sea posible realizarlas a tiempo da lugar a que esos trabajos no sean procesados, ni siquiera fuera de plazo. Este problema, en una situación real, desembocaría en daños económicos y organizativos graves.

2.1.2 Con máquinas con clases

Existen otros casos más comunes, como el del siguiente ejemplo, en los que los trabajos son de diferentes tipos o clases y los recursos disponibles sólo están preparados para procesar trabajos de determinadas clases específicas. Para aclarar esto disponemos de la matriz de compatibilidad (A_{DC}), la cual determina si un recurso (columna) está capacitado para interactuar con una clase de trabajo (fila), marcando su casilla intersección con un 1. En el caso de que el recurso no soporte una clase de trabajo, marca un 0. [1,2]

A continuación, adjunto la matriz en cuestión para este ejemplo:

A_{DC}	R1	R2	R3
C1	0	1	0
C2	1	0	1
C3	1	1	0

Tabla 2-2. Matriz de compatiblidad(A_{DC})

A raíz de ella conocemos que hay tres clases de trabajos y tres recursos en total. Los de clase C1 solo pueden ser procesados por el recurso R2, los de clase C2, por los recursos R1 y R3, Y los de clase C3, por los recursos R1 y R2.

Por otro lado, volvemos a crear la misma tabla de datos que en el ejemplo anterior puesto que vuelven a ser 12 tareas a procesar, cada una con su instante de inicio más temprano (I_i), su instante de inicio más tardío (F_i) y su duración. Sin embargo, es necesario añadir una columna más (c_i) en la que se indique la clase del trabajo (1,2 o 3).

En la sexta columna de la tabla vuelve a aparecer otro diagrama de Gantt en el que vienen representados los

intervalos donde se podrá realizar cada trabajo (casillas blancas), cuya longitud dependerá de las condiciones de inicio I_i y F_i. Dentro de ellos, las casillas amarillas marcan en qué momento del intervalo se ha terminado realizando el trabajo si éste es de clase 2, las casillas naranjas para los de clase 1 y las casillas rojas para los de clase 3.

El eje de abcisas del diagrama marca el tiempo y el eje de ordenadas relaciona cada trabajo con su intervalo. Al final de cada fila aparece el recurso encargado de operar cada trabajo (R1, R2 o R3).

TR	I_i	F_i	d_i	c_i
1	1	4	4	2
2	3	4	3	2
3	2	4	5	1
4	5	7	2	1
5	4	5	3	3
6	7	9	4	1
7	6	8	3	2
8	7	8	4	3
9	10	12	2	3
10	11	13	1	1
11	11	12	5	2
12	13	16	3	3

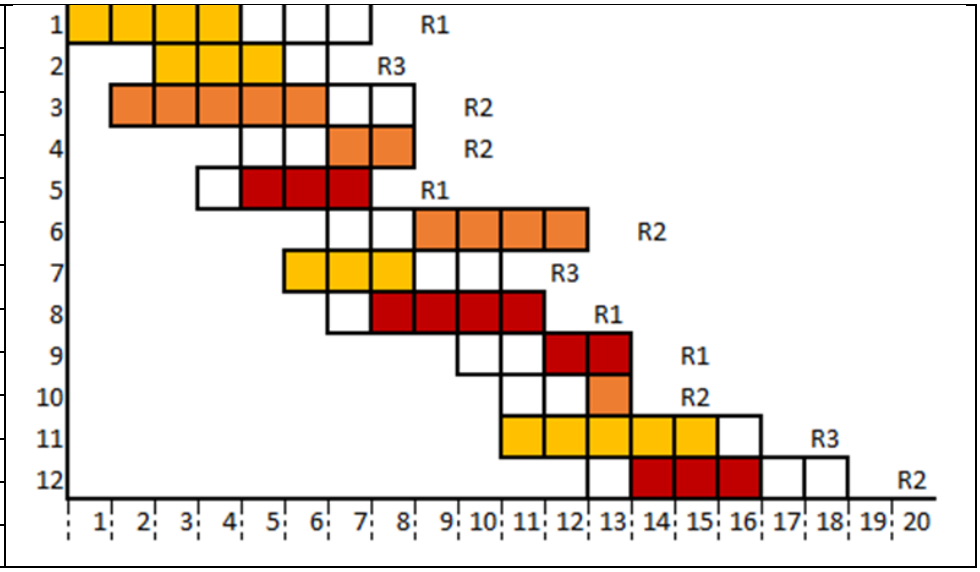

Tabla 2-3. Caso con máquinas con clases

Un detalle llamativo de este problema es que, al tener tareas que pueden ser realizadas por diferentes máquinas, aumentan las posibilidades de que todos o la mayoría de trabajos se realicen en tiempo dentro de su intervalo, cosa que ocurre en este ejemplo.

Otra particularidad es que, si en el instante de inicio de un trabajo se encuentran disponibles dos o más recursos que puedan realizarlo, el programador tendrá libertad de elección del recurso. Esto ocurre en el momento de realizar la tarea 12 del ejemplo ya que en el instante 14 los recursos R1 y R2 están libres. He terminado escogiendo R2, pero si hubiese elegido R1 no habría afectado al resultado final de la programación.

3 MODELOS VSP

EN este capítulo veremos tres problemas dentro de la variante VSP aunque cada uno tendrá sus características específicas que lo diferenciarán de los otros dos. Se desarrollarán de forma extensa todas las partes que componen el modelo de cada problema y se irán traduciendo al lenguaje de programación de Lingo para su posterior simulación y análisis.

3.1 Variable scheduling problem-Versión 1 (Trabajos medibles)

Esta primera versión tiene como diferenciación particular que los trabajos tienen una duración continua, es decir, son medibles, en lugar de los dos ejemplos que aparecen en el capítulo anterior, cuyos trabajos son unitarios.

3.1.1 Tabla de elementos

Los elementos de un problema son las distintas entidades individuales que intervienen en él y vienen recogidos en la siguiente tabla:

ELEMENTOS	CJTO	NC	DATOS				
			Nombre	Parámetro	Tipo	Pertenencia	Valor
Trabajos	i=1..n	I_M	Inicio Ventana	I_i	C	P	-
			Fin Ventana	F_i	C	P	-
			Duración	d_i	C	P	-
			Clase	C_i	O	P	-
			Peso	B_i	C	P	-
Máquinas	j=1..m	I_U	Clase	D_j	O	P	-
Clases Trabajos	k=1..CT	I_U	Compatibilidad	A_{kr}	B	C	-
Clases Máquinas	r=1..CM	I_U		A_{kr}			

Tabla 3-1. Tabla de elementos de un problema VSP, versión 1

En la primera columna tenemos los diferentes tipos de elementos que aparecen en el problema.

En la segunda columna, el tamaño de cada grupo de elementos y el subíndice que los recorrerá durante el resto del modelo.

La tercera columna se refiere a la naturaleza cuantitativa de los elementos. En los problemas que veremos en este proyecto solo aparecen dos tipos:

- Individual unitario (I_U): El elemento en cuestión no tendrá ninguna característica que pueda ser medida de forma continua. [5]
- Individual medible (I_M): El elemento tendrá una o varias características medibles, como podrían ser la altura, el peso o el volumen. [5]

Dentro de la columna DATOS, tendremos los nombres y parámetros de las todas las características asociadas a cada elemento.

El tipo de cada característica podrá ser:

- Continuo (C): Se refiere a una magnitud entera o continua. Si un elemento tiene una característica de tipo continuo será individual medible (I_M). [5]
- Ordinal (O): El dato posee un valor entero que hace referencia a un ordinal de una lista de diferentes opciones. [5]
- Binario (B): Solo podrá tener dos valores, 1 y 0, que suelen definir si un elemento cumple o no una cierta propiedad. [5]

La pertenencia tendrá dos variantes:

- Propia (P): La característica será exclusiva de un elemento. [5]
- Compartida (C): La característica estará encadenada a dos o más elementos diferentes, por lo que llevará varios subíndices referidos a cada uno de esos elementos. [5]

La columna Valor se deja vacía porque implicaría que hubiese que mostrar una gran cantidad de números. Además, en este capítulo, todavía no dispondremos de los datos númericos para resolver los modelos, debido a que la obtención de los mismos se realizará en el siguiente capítulo, donde extraeremos las baterías de problemas disponibles en Lingo.

Volviendo a la tabla de modelo, de ella podemos extraer que:

- Los trabajos serán el primer elemento. Habrá n trabajos y los recorreremos con el subíndice i. Coincidiendo con los dos casos prácticos vistos en el apartado anterior, los trabajos tendrán 4 características, inicio y fin ventana (instantes de comienzo temprano y tardío del trabajo), la duración (d_i) y la clase (C_i) de cada trabajo. Las cuatro son de pertenencia propia, es decir, pertenecen exclusivamente a los trabajos y las tres primeras son continuas, dando lugar a que los trabajos sean individuales medibles. La clase de trabajo es ordinal. Por otra parte, cada trabajo tendrá un peso (B_i), característica continua y única también para los trabajos. Los trabajos con mayor peso serán los prioritarios a la hora de su realización.
- Las máquinas serán otro elemento individual medible. Existirán m máquinas y las recorreremos con el subíndice m. Tienen como única característica la clase (D_j) de cada máquina, de carácter propio y ordinal.
- Es necesario a su vez considerar las clases de trabajos y máquinas como elementos, cada uno con un tamaño, CT y CM, respectivamente. Tendrán una única característica, la compatibilidad (A_{kr}), la cual será compartida por ambos y será binaria. Tendrá valor 1 si una clase de trabajo es compatible con una clase de máquina, o 0, si no lo son. En definitiva, es la función que tenía la matriz de compatibilidad A del ejemplo de máquinas con clases del apartado anterior.

3.1.2 Función objetivo

La función objetivo de cualquier problema busca representar de forma matemática la meta del mismo, minimizando desventajas o maximizando pesos. [5] En el caso de este problema, el objetivo será maximizar pesos.

$$Max \sum_{i=1}^{n} \sum_{j / A_{C_i D_j}=1} B_i \alpha_{ij}$$

Esos pesos (Bi) están referidos al plus económico que supone la realización de un trabajo. De ahí que se recorra con su mismo subíndice (i).

La otra variable que aparece en la FO, alfa, la explicaremos en el subapartado de actividades de decisión.

La FO al completo consta del producto de esas dos variables dentro de dos sumatorios, uno que recorre los trabajos y, otro, las máquinas, aunque sólo las que cumplan con el criterio de compatibilidad con los trabajos.

3.1.3 Actividades de decisión

Las actividades de decisión son las diferentes acciones que se deducen de un problema, desembocando en las variables de decisión del problema, las cuales se introducen en el modelo para su resolución. En ellas pueden intervenir uno o varios de los elementos contenidos en la tabla de elementos. [5]

En este problema se identifican dos de ellas:

- Programar los trabajos. La variable que se deduce de esta acción la denominaremos "x" y estará en función de i (índice de cada trabajo). $\forall i : x_i =$ instante en el que se programa el trabajo i.
- Asignar las máquinas a los trabajos. Utilizaremos una variable binaria, alfa, para este caso. Alfa adoptará un valor u otro si se asigna o no una máquina a un trabajo determinado.

  $\forall i, j / A_{C_i D_j} = 1 : \alpha_{ij} = 1$ si asigno maquina j a trabajo i; 0 si no

  La variable alfa sólo podrá actuar si, además de que se asigne una máquina a un trabajo, ésta sea compatible con él.

3.1.4 Especificaciones

Las especificaciones de un problema escenifican las reglas y restricciones presentes en él. Dentro de cada problema las dividiremos en diferentes subgrupos. [5]

Algunas de las especificaciones que veremos en cada modelo estarán númeradas al final con un número entre paréntesis, lo que indica que son proposiciones simples. Se distinguen de las proposiciones lógicas compuestas en que no usan operadores o conectivas lógicas como sí solo sí (SSS), SI-ENTONCES, Y, O, O BIEN, NO. Este segundo grupo, las proposiciones lógicas compuestas, lo formaran el resto de las especificaciones que no están numeradas. [5]

Añadir también que la hora de realizer el modelo escrito, a partir de las proposiciones simples (numeradas), se van deduciendo el resto de proposiciones compuestas.

3.1.4.1 Relación implícita entre actividades

Las especificaciones de este subgrupo son quizá, las más inmediatas de deducir.

$$\forall i: \sum_{j/A_{C_iD_j}=1} \alpha_{ij} > 0 \; sss \; x_i > 0$$

$$\forall i: \sum_{j/A_{C_iD_j}=1} \alpha_{ij} \geq 1 \; sss \; x_i \geq \xi$$

$$\forall i: x_i \geq \xi \sum_{j/A_{C_iD_j}=1} \alpha_{ij} \quad (1)$$

$$\forall i: x_i \leq F_i \sum_{j/A_{C_iD_j}=1} \alpha_{ij} \quad (2)$$

Las cuatro están referidas a cada uno de los trabajos individualmente:

- La primera remarca que, si el instante en el que se programa el trabajo es mayor que 0, entonces el trabajo se programará seguro en alguna de las máquinas que sean compatibles con él. En esta especificación aparece por primera vez la conectiva lógica SSS (sí solo si). Esta indica que la primera proposición de la especificación, en este caso que el sumatorio de las alfas sea mayor que cero se cumplirá solo si se cumple la segunda proposición, es decir, que la variable x para el trabajo que se este evaluando sea mayor que cero.
- La segunda tiene una estructura similar a la primera. Para cada trabajo se cumplirá que el sumatorio de todas las variables alfa, que recorre solo las máquinas (j) que son compatibles con ese trabajo, es mayor o igual que uno sí solo si la variable x de ese trabajo es mayor o igual que epsilón, que representa a un número muy pequeño. Esto tiene sentido, ya que, si el trabajo tiene un instante de inicio de programación en el tiempo, supondrá que ese trabajo se ha programado seguro en una máquina compatible.
- La tercera especificación es una proposición simple que regula que el valor del instante en el tiempo en que se programa el trabajo en cuestión (variable x), sea mayor que el sumatorio de las variables alfas asociadas a ese trabajo y que recorre solo las máquinas compatibles con él, multiplicado por un número muy pequeño (épsilon).
- Y la cuarta especificación recoge que el instante de inicio de cada trabajo (x) sea menor o igual que el producto entre el sumatorio de las alfas, recorrido únicamente por las máquinas (j) compatibles con el trabajo y el fin ventana (F) o instante de programación más tardío del trabajo. De esta forma se controla que cada trabajo que se vaya programando este dentro de su intervalo de inicio asignado.

3.1.4.2 Un trabajo procesado por una máquina a lo sumo

Para cada trabajo, el sumatorio de sus variables alfa asignadas recorrido por las máquinas (j) compatibles con él, tiene que ser menor o igual que uno.

$$\forall i: \sum_{j/A_{C_iD_j}=1} \alpha_{ij} \leq 1 \quad (3)$$

Esta especificación evita que un mismo trabajo sea programado en más de una de sus máquinas compatibles.

3.1.4.3 Trabajos asignados se programan dentro de su ventana temporal

En este subgrupo contamos con tres especificaciones.

$$\forall i : x_j \leq F_i \quad (4)$$

$$\forall i : Si \sum_{j / A_{C_i D_j}=1} \alpha_{ij} = 1 \; entonces \; x_i \geq I_i$$

$$\forall i : x_i \geq I_i \sum_{j / A_{C_i D_j}=1} \alpha_{ij} \quad (5)$$

- En la primera se vuelve a insistir que, para cada trabajo, su instante de programación (x), tendrá que ser igual o inferior a su último instante posible de inicio (F).
- En la segunda, surge otra nueva conectiva lógica, en este caso SI-ENTONCES. Con ella se quiere señala que si se cumple la primera proposición simple que constituye la proposición compuesta, en este caso que el sumatorio de las alfas asignadas a un trabajo y recorrido por las máquinas (j) en las que pueda realizarse sea igual a uno, entonces se cumplirá también la segunda proposición, que el instante de programación del trabajo (x) sea igual o superior al inicio de ventana o instante más temprano de programación posible de ese trabajo (I).

  Con esta especificación se consigue que, en el caso de que un trabajo se programe, no se haga antes del comienzo de su intervalo de posibles inicios.
- La última especificación de este subgrupo marca que para cada trabajo (i) el momento en el que es programado (x), tiene que ser mayor o igual que el producto entre su inicio de ventana (I) y el sumatorio de sus alfas recorrido solo por las máquinas (j) en las cuales es realizable ese trabajo.

  Como se puede observar partiendo de esta última especificación se puede deducir la segunda.

3.1.4.4 Solapamiento de trabajos en una máquina

En este último subgrupo tenemos dos conjuntos numerosos de especificaciones divididos según las cirscuntacias de inicio y fin de los trabajos que se van analizando con ellas. Su principal objetivo es el de evitar que se solapen los trabajos realizables en una misma máquina, es decir, que su programación no coincida en el tiempo.

Remarcar que a partir de aquí utilizaremos también un nuevo subíndice (i'). Hace referencia a un trabajo distinto al que se refiera en ese mismo momento el subíndice i. Servirá para comparar uno y otro.

También añadir que surgen dos nuevas conectivas lógicas, Y y O.

Y, es una conjunción que une dos proposiciones simples. Si aparece tendrán que cumplirse ambas o viceversa. Puede identifircarse también con los símbolos & y ^.

O, sin embargo, es una disyunción entre dos proposiciones simples. Tendrá que cumplirse una de las dos.

Comenzamos con el primer conjunto de especificaciones.

$$\forall i, i' / I_i \leq I_{i'} \,\&\, I_i + d_i > I_{i'} \wedge I_{i'} + d_{i'} \geq F_i - d_i, \; j / A_{C_i D_j} = 1 \wedge A_{C_{i'} D_j} = 1 :$$

$$Si \; \alpha_{ij} = 1 \; Y \; \alpha_{i'j} = 1 \; entonces \; x_i + d_i \leq x_{i'}$$

$$Si \; \alpha_{ij} + \alpha_{i'j} \geq 2 \; entonces \; x_i - x_{i'} \leq d_i$$

$$\alpha_{ij} + \alpha_{i'j} \geq 2\pi_{ii'j} \quad (6)$$

$$\alpha_{ij} + \alpha_{i'j} \leq 1 + \pi_{ii'j} \quad (7)$$

$$x_i - x_{i'} \leq d_i + (F_i - d_i)(1 - \pi_{ii'j}) \quad (8)$$

En todas ellas recorremos cada vez dos trabajos distintos, i y i', con las condiciones de que el instante de inicio más temprano del trabajo i, (I) sea menor que el instante de inicio más temprano del trabajo i', además de que el

resultado de la suma entre el instante de inicio del trabajo i y su duración (d), deben superar al instante de inicio más temprano del trabajo i'. La tercera condición es que la suma del instante de inicio más temprano del trabajo i' y su duración iguale o supere a la diferencia entre el instante de inicio más tardío y la duración del trabajo i. Como las tres condiciones están conectadas por conjunciones, las tres deberán de cumplirse a la vez.

También recorreremos las máquinas (j), aunque únicamente las que sean a la vez compatibles con el trabajo i y el i', o, lo que es lo mismo, que la compatibilidad de cada trabajo (A) con la máquina en cuestión, sea igual a uno.

- La primera especificación recogida dentro de las condiciones anteriores nos dice que en el momento en el que se confirme que se han programado dos trabajos distintos en la misma máquina, es decir, que las alfas de i y i' respecto a j sean igual a uno, se deberá cumplir que el instante de programación de i más su duración (d) sea menor que el instante en el que se programa i' para que no coincidan.
- La segunda es otra forma de ver la primera. Si se cumple que la suma de las alfas de los trabajos i y i' respecto a la misma máquina j es igual o mayor que 2, se dará también que la diferencia entre los instantes de programación (x) de los trabajos i e i' es menor o igual que la duración del trabajo i.
- Las otras tres especificaciones son proposiciones simples en las que interviene una nueva variable auxiliar, pi, la cual esta referida a un trabajo i, otro trabajo i' y una máquina j.

  En la primera de ellas, la suma de las alfas de cada trabajo i y i' procesados en la misma máquina j es superior o igual que dos veces la variable auxiliar pi de esos mismos trabajos.

  En la segunda, esa misma suma tiene que ser menor que la de la variable pi más una unidad más.

  Y en la tercera, la diferencia entre los instantes de programación x de los trabajos i e i' es menor que la duración d del trabajo i más el producto de la diferencia entre el instante de programación más tardío (F) y la duración del trabajo i entre la diferencia entre una unidad menos la variable pi.

Continuamos con el segundo grupo de especificaciones.

$$\forall i,i'/\, I_i \le I_{i'} \,\&\, I_i + d_i > I_{i'} \wedge I_{i'} + d_{i'} < F_i - d_i,\ j/\, A_{C_iD_j} = 1 \wedge A_{C_{i'}D_j} = 1:$$

$$Si\ \alpha_{ij} = 1\ Y\ \alpha_{i'j} = 1\ entonces\ x_i + d_i \le x_{i'}\ O\ x_{i'} + d_{i'} \le x_i$$

$$Si\ \alpha_{ij} + \alpha_{i'j} \ge 2\ entonces\ x_i - x_{i'} \le d_i\ O\ x_{i'} - x_i \le d_{i'}$$

$$x_i - x_{i'} \le d_i\ sss\ \delta_{ii'} = 1$$

$$x_i - x_{i'} \le d_i + (F_i - d_i)(1 - \delta_{ii'}) \quad (9)$$

$$x_i - x_{i'} \ge (d_i + \xi)(1 - \delta_{ii'}) - I_{i'}\delta_{ii'} \quad (10)$$

$$Si\ \alpha_{ij} + \alpha_{i'j} \ge 2\ entonces\ \delta_{ii'} = 1\ O\ \delta_{i'i} = 1$$

$$Si\ \alpha_{ij} + \alpha_{i'j} \ge 2\ entonces\ \delta_{ii'} + \delta_{i'i} \le 1$$

$$\alpha_{ij} + \alpha_{i'j} \ge 2\pi_{ii'j} \quad (6)$$

$$\alpha_{ij} + \alpha_{i'j} \le 1 + \pi_{ii'j} \quad (7)$$

$$\delta_{ii'} + \delta_{i'i} \le 2 - \pi_{ii'j} \quad (11)$$

La forma de recorrerlas será idéntica a la del primer grupo exceptuando la tercera condición al recorrer los trabajos i e i'. La suma del instante de inicio más temprano del trabajo i' (I) y su duración d, debrá ser inferior que la diferencia entre el instante de incio más tardío del trabajo i (F) y su duración (d).

3.2 Variable scheduling problem-Versión 2 (Trabajos unitarios)

Para este segundo problema, los trabajos a programar tendrán duración entera, es decir serán unitarios. Este caso

si coincidiría con los dos ejemplos vistos en el anterior capítulo.

3.2.1 Tabla de elementos

La tabla de elementos de este problema es muy similar a la del primer problema, por lo que incidiré sólo en las pequeñas variaciones apreciables.

ELEMENTOS	CJTO	NC	DATOS				
			Nombre	Parámetro	Tipo	Pertenencia	Valor
Trabajos	i=1..n	I_U	Inicio Ventana	I_i	O	P	-
			Fin Ventana	F_i	O	P	-
			Duración	d_i	E	P	-
			Clase	C_i	O	P	-
			Peso	B_i	C	P	-
Máquinas	j=1..m	I_U	Clase	D_j	O	P	-
Periodos	t=1…L	I_U					
Clases Trabajos	k=1..CT	I_U	Compatibilidad	A_{kr}	B	C	-
Clases Máquinas	r=1..CM	I_U		A_{kr}			

Tabla 3-2. Tabla de elementos de un problema VSP, versión 2

Se mantienen los mismos 4 elementos del primer problema, más un quinto, los períodos, que se recorrerán con el subíndice t, tendrán tamaño L y no contarán con características asociadas.

Las características de cada elemento vistas en el anterior problema se mantienen. Sin embargo, al ser los trabajos unitarios, su duración pasará a ser de tipo entero y sus instantes de inicio y fin ventana serán ordinales.

3.2.2 Función objetivo

La función objetivo es exactamente la misma que en el primer problema, maximizar el peso de la realización de los trabajos.

$$Max \sum_{i=1}^{n} \sum_{j / A_{C_i D_j}=1} B_i \alpha_{ij}$$

3.2.3 Actividades de decisión

Identificamos de nuevo dos actividades de decisión:

- Programar los trabajos dentro de los periodos. Usaremos la variable binaria beta, que estará compartida por los trabajos y los periodos. Tomara un valor u otro según se programe un trabajo dentro de un periodo o no.

$$\forall i, t = I_i \dots F_i : \beta_{it} = 1 \text{ si programo trabajo } i \text{ en el periodo } t; 0 \text{ si no}$$

Los periodos estarán comprendidos entre el inicio y el fin ventana de cada elemento.

- Asignar las máquinas a los trabajos. Para cada trabajo i junto con cada una de las máquinas j compatibles con cada uno, creamos la variable binaria alfa (α), la cual tendrá valor uno en el caso de que el trabajo i se haya asignado a la máquina compatible j, o valor cero si no se ha producido esa asignación.

$$\forall i, j / A_{C_i D_j} = 1 : \alpha_{ij} = 1 \text{si asigno maquina } j \text{ a trabajo } i;\ 0 \text{ si no}$$

3.2.4 Especificaciones

3.2.4.1 Relación implícita entre actividades

En este grupo de especificaciones tenemos dos proposiciones compuestas y una simple.

$$\forall i : \sum_{j / A_{C_i D_j} = 1} \alpha_{ij} > 0 \ sss \sum_{t=I_i}^{F_i} \beta_{it} > 0$$

$$\forall i : \sum_{j / A_{C_i D_j} = 1} \alpha_{ij} \geq 1 \ sss \sum_{t=I_i}^{F_i} \beta_{it} \geq 1$$

$$\forall i : \sum_{j / A_{C_i D_j} = 1} \alpha_{ij} = \sum_{t=I_i}^{F_i} \beta_{it} \quad (1)$$

- La primera indica que para cada trabajo i, se cumplirá que el sumatorio de sus variables alfa asignadas recorrido exclusivamente por las máquinas j compatibles con el trabajo es mayor que cero con la extricta condición de que el sumatorio de las variables beta para periodos t que se encuentren dentro del intervalo de posibles inicios de ese trabajo tiene que ser mayor que cero.

  De esta manera conseguimos que, si cualquier trabajo se asigna a una máquina, previamente tenga ya asignado también un periodo o instante de programación.
- La segunda, viene con la misma estructura que la primera, aunque esta vez los resultados de los dos sumatorios tendrán que ser mayor o igual que uno.
- Y la tercera es la proposición simple, que iguala los dos sumatorios vistos en las otras dos especificaciones.

3.2.4.2 Un trabajo procesado por una máquina a lo sumo

Esta restricción evita que un mismo trabajo sea programado en más de una de sus máquinas compatibles.

$$\forall i : \sum_{j / A_{C_i D_j} = 1} \alpha_{ij} \leq 1 \quad (2)$$

Para cada trabajo, el sumatorio de sus variables alfa recorrido por las máquinas j compatibles con cada uno, debe

ser inferior o igual a uno. Esto tiene sentido ya que si valiese 2 significaría que ese trabajo se ha asignado a dos máquinas diferentes, situación prohibida en el modelo.

3.2.4.3 Solapamiento de trabajos en una máquina

En este grupo constan seis especificaciones recorridas de la misma manera.

$$\forall i,i',j / A_{C_iD_j}=1 \wedge A_{C_{i'}D_j}=1, t\in[I_i,F_i], t'\in[I_{i'},F_{i'}] / t+d_i>t':$$
$$Si\ \beta_{it}=1\ Y\ \beta_{i't'}=1\ entonces\ NO(\alpha_{ij}=1\ Y\ \alpha_{i'j}=1)$$
$$Si\ \beta_{it}=1\ Y\ \beta_{i't'}=1\ entonces\ \alpha_{ij}+\alpha_{i'j}\leq 1$$
$$Si\ \beta_{it}+\beta_{i't'}\geq 2\ entonces\ \alpha_{ij}+\alpha_{i'j}\leq 1$$
$$\beta_{it}+\beta_{i't'}\geq 2\omega_{ii't} \quad (3)$$
$$\beta_{it}+\beta_{i't'}\leq 1+\omega_{ii't} \quad (4)$$
$$\alpha_{ij}+\alpha_{i'j}\leq 2-\omega_{ii't} \quad (5)$$

Para cada trabajo i y cada trabajo i' se van probando una a una las máquinas j que sean compatibles con ambos trabajos y los periodos t y t', cada uno para su correspondiente trabajo, verificando que se encuentren dentro de los respectivos intervalos delimitados por los inicio y fin de ventana de cada trabajo. Además, tendrá que darse la condición de que el periodo (t) del trabajo i más su duración (d) sea mayor que el periodo (t) del trabajo i'.

Comentar que en la primera especificación de este grupo, surge una nueva conectiva lógica, NO, que representa una negación. Todas las proposiciones que se encuentren incluidas dentro de esta conectiva no se cumplirán.

Si retomamos la primera especificación, en ella se establece que en el momento en el que dos trabajos distintos i e i' se programen en dos periodos distintos t y t' respectivamente, es decir, que sus variables beta (β) se activen, será imposible que esos dos trabajos se programen en la misma máquina. Sus respectivas variables alfa (α) para una misma máquina j no podrán valer 1 a la vez.

Con la segunda se quiere incidir en la misma prohibición que la primera expresadolo de una forma diferente. Si las dos variables beta de cada trabajo se activan, la suma de las variables alfa tendrá que ser igual o inferior a uno.

En la tercera se vuelve a repetir el enfoque de las otras dos, aunque la primera proposición que tiene que cumplirse en ella es que la suma de las dos variables beta sea mayor o igual que dos.

Las tres últimas son proposiciones simples más sencillas. Añadir que en todas ellas se usa una nueva variable auxiliar binaria denominada omega (ω). Estará en función de los trabajos i e i' y de los periodos t.

3.3 Variable scheduling problem-Versión 3 (Trabajos unitarios)

Esta última versión es parecida al segundo problema, de nuevo con trabajos unitarios. Las diferencias las comentaremos en el subapartado de la tabla de elementos.

3.3.1 Tabla de elementos

En este problema se prescinde de tratar a las máquinas como elementos.

ELEMENTOS	CJTO	NC	DATOS				
			Nombre	Parámetro	Tipo	Pertenencia	Valor
Trabajos	i=1..n	I_U	Inicio Ventana	I_i	O	P	-
			Fin Ventana	F_i	O	P	-
			Duración	d_i	E	P	-
			Clase	C_i	O	P	-
			Peso	B_i	C	P	-
Periodos	t=1…L	I_U					
Clases Trabajos	k=1..CT	I_U	Compatibilidad	A_{kr}	B	C	-
Clases Máquinas	r=1..CM	I_U	Num máquinas	M_r	E	P	-
				A_{kr}			

Tabla 3-3, Tabla de elementos de un problema VSP, versión 3

El resto de elementos y sus características se mantienen con respecto al segundo problema. El otro cambio es que los elementos Clases Máquinas tendrán una nueva característica, el número de máquinas (M_r), el cual indicará cuantas máquinas habrá de cada clase. Será de tipo entero y exclusiva de los elementos Clases máquinas.

3.3.2 Función objetivo

La función objetivo vuelve a querer maximizar los pesos de la realización de trabajos. La otra variable que aparece, además del peso, beta (β), será desarrollada en el siguiente subapartado.

$$Max \sum_{i=1}^{n} \sum_{r/A_{C_i r}=1} \sum_{t=I_i}^{F_i} B_i \beta_{itr}$$

La FO está constituida por el producto del peso por beta dentro de tres sumatorios. Uno recorre los n trabajos, otro, las clases de maquinas compatibles con los trabajos y, el último, los periodos, comenzado cada uno por el inicio ventana de cada trabajo y terminando con el fin ventana.

3.3.3 Actividades de decisión

Contaremos en este caso con una única actividad de decisión, programar cada trabajo en un periodo con una clase de máquina determinada. A raíz de ella definimos la variable binaria beta (β), que estará en función de los trabajos, los periodos y las clases, y se activará si se programan a la vez cada uno de esos elementos.

$$\forall i, t = I_i \ldots F_i, r / A_{C_i r} = 1 : \beta_{itr} = 1 \text{ si programo trabajo } i \text{ en el periodo } t \text{ con la clase } r; 0 \text{ si no}$$

También será necesario que la clase del trabajo sea compatible con la clase de la máquina.

3.3.4 Especificaciones

3.3.4.1 Un trabajo procesado por una clase máquina y un periodo a lo sumo

Esta especificación evita que un mismo trabajo se programe de nuevo en una misma máquina y un mismo periodo.

$$\forall i : \sum_{r/A_{C_i r}=1} \sum_{t=I_i}^{F_i} \beta_{itr} \leq 1 \quad (1)$$

Para cada trabajo el sumatorio de su variable beta asignada, recorriendo todas las clases de máquinas compatibles

y los instantes de tiempo dentro de su periodo t, deberá ser menor o igual que 1.

3.3.4.2 Solapamiento de trabajos en una máquina

Por otra parte, tenemos la restricción que evita que los trabajos que se activen en un intervalo de tiempo determinado y en máquinas de una misma clase, sean los mismos o menos que las máquinas disponibles de esa clase (M_r), evitando así el solapamiento de trabajos.

$$\forall r,t : \sum_{i} \sum_{t'/t' \in [I_i, F_i] \& t'+d_i > t} \beta_{it'r} \leq M_r \quad (2)$$

También especifica que los períodos t donde finalmente se comienzan a operar los trabajos se encuentren dentro de los límites de inicio temprano y tardío de cada uno y que el momento de inicio del siguiente trabajo sea después del fin de la duración del anterior si ambos trabajos se realizan seguidos en una misma máquina.

4 IMPLEMENTACIÓN DE MODELOS EN LINGO

En este capítulo explicaremos brevemente el funcionamiento del programa Lingo en el que introduciremos los problemas desarrollados en el capítulo anterior para su posterior análisis y simulación.

Veremos cómo se traspasa cada parte de los modelos al lenguaje Lingo y los cambios que se producen.

4.1 Introducción a Lingo

Lingo es un software capacitado para recrear modelos de optimización lineales o no, y encontrarles la mejor solución posible. Consigue que, tanto la construcción del modelo como la búsqueda de su solución óptima, se realicen de forma más sencilla y efectiva que de la forma manual. [6]

Posee, además, un lenguaje de programación fácil de entender, con el que, con unas pocas líneas de código, podremos representar modelos matemáticos con un gran número de variables y restricciones. [6]

Otra facultad clave de Lingo es que es capaz de tratar los datos númericos de forma externa al resto del modelo. Lee datos de sitios ajenos al modelo, ya sean hojas de cálculo, archivos de texto o bases de datos. De esta forma consigue evaluar un mismo problema con diferentes datos sin necesidad de cambiar el código, evitando así posibles errores. [6]

4.2 Versión 1 en Lingo

Al ser éste el primer modelo de Lingo descrito, además de comentar el modelo en sí, veremos también de forma general la estructura del modelo en Lingo.

Comenzamos creando la sección SETS, donde designamos los distintos conjuntos referidos a los diferentes elementos que aparecían en la tabla de elementos del modelo visto en el capítulo anterior. Cada conjunto referido a un solo elemento se denomina conjunto primitivo. En estos conjuntos, aparece primero el nombre, que corresponde con el del elemento en cuestión, seguido del tamaño del conjunto y, por ultimo, de los datos pertenecientes al elemento, que estaban recogidos igualmente en la tabla de elementos. [5,6]

```
MODEL:
SETS:
TRABAJOS/1..n/:I,F,d,C,x,B;
MAQUINAS/1..m/:D;
ASIGNAR(TRABAJOS,MAQUINAS):Alfa;
CLASES_TRABAJOS/1..CT/;
CLASES_MAQUINAS/1..CM/;
COMPATIBILIDAD(CLASES_TRABAJOS,CLASES_MAQUINAS):A;
AUXILIAR(TRABAJOS,TRABAJOS,MAQUINAS):pi;
AUXILIAR2(TRABAJOS,TRABAJOS):Sigma;
ENDSETS



DATA:

ENDDATA
```

Figura 4-1. MODEL del modelo 1 en Lingo.

En el modelo, los conjuntos referidos a los elementos trabajos, máquinas, clases de trabajos y clases de máquinas son los únicos conjuntos primitivos.

Los variables dato que aparecen dentro de cada conjunto son las que en la tabla de elementos estaban recorridas por el mismo subíndice del elemento al que se refiere el conjunto. Por ejemplo, las variables inicio ventana (I) y fin ventana (F) aparecen con el subíndice i en la tabla de elementos, el correspondiente a los trabajos. Sin embargo, en esta parte del modelo en Lingo, no es necesario introducir los subíndices que recorrían cada elemento.

Como puede observarse en la imagen anterior, estos conjuntos no son los únicos que aparecen en la sección SETS. Tenemos también otras variables compartidas entre dos o más elementos y, por tanto, con dos o más subíndices que las recorren, como son la compatibilidad entre máquinas y trabajos (A), o variables que creamos en el desarrollo de las especificaciones y las actividades de decisión, como la variable alfa que marcaba si se asignaba o no un trabajo a una máquina o las variables auxiliares pi y sigma.

Para estas variables compartidas es necesario crear los llamados conjuntos derivados a partir de la combinación de dos o más conjuntos primitivos. Véase el caso del conjunto derivado ASIGNAR, que recoje los conjuntos primitivos TRABAJOS y MÁQUINAS, los cuales usan los subíndices i y j respectivamente. A su vez dentro del conjunto ASIGNAR se encuentra la variable Alfa recorrida por los dos subíndices de los conjuntos primitivos que constituyen el conjunto derivado. [5,6]

Otro tema clave a tratar es que, en el interior de algunos de los conjuntos derivados, aparece repetido uno de los conjuntos primitivos que lo forman. Esto ocurre debido a que hay algunas variables en el modelo recorridas por un mismo subíndice duplicado. Por ejemplo, la variable auxiliar sigma tiene como subíndices i e i'. Recordamos que la i hace referencia a un trabajo en concreto y la i' a otro trabajo diferente a el primero. Por ello, este tipo de variables necesitarán de la creación de un nuevo conjunto derivado que contenga doblemente ese conjunto primitivo.

Una vez completada la sección SETS, procedemos con la sección DATA. En esta sección, se definen todos los posibles valores numéricos de todas las variables con las que cuenta el modelo. [5,6] Sin embargo, en el código anterior aparece completamente vacía. Esto se debe a que los datos que vamos a usar provienen de una batería de problemas externa que comentaremos en el siguiente capítulo. Esta forma de obtención de los datos es la que exponíamos al inicio de este capítulo en la introdución a Lingo.

Tras la sección DATA, se agrupa el resto de partes de modelo matemático traducidas a lenguaje Lingo, es decir, la función objetivo, las actividades de decision y las especificaciones.

```
MAX=@SUM(TRABAJOS(i):@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:B(i)*Alfa(i,j)));

!ACTIVIDADES DE DECISION;

!Programar trabajos;
@FOR(TRABAJOS(i):@GIN(x));

!Asignar máquinas a trabajos;
@FOR(TRABAJOS(i):@FOR(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:@BIN(Alfa)));

!ESPECIFICACIONES;

!Relacion implicita entre actividades;
@FOR(TRABAJOS(i):x(i)>=E*@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:Alfa(i,j)));
@FOR(TRABAJOS(i):x(i)<=F(i)*@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:Alfa(i,j)));

!Un trabajo procesado por una maquina a lo sumo;
@FOR(TRABAJOS(i):@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:Alfa(i,j)<=1));

!Trabajos asignados se programan dentro de su ventana temporal;
@FOR(TRABAJOS(i):x(i)<=F(i));
@FOR(TRABAJOS(i):x(i)>=I(i)*@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:Alfa(i,j)));
```

Figura 4-2. Función objetivo, actividades de decisión y especificaciones del modelo 1 en Lingo.

La primera línea del código representa la función objetivo. En nuestro caso siempre será de tipo maximizar (MAX). Los sumatorios que aparecen tanto dentro de ella como en el resto de los modelos aparecen en lenguaje Lingo como @SUM (∑). Dentro de sus paréntesis se engloba todo lo que estaba dentro de cada sumatorio en el modelo matemático escrito. Se puede incluir sumatorios dentro de otros, como es el caso en esta función objetivo. [6]

La simbología matemática entre esta última parte del modelo escrito y en Lingo es muy similar. A simple vista se puede identificar qué especificación corresponde a qué linea de código en Lingo. Además, se han añadido comentarios (frases en verde) para una mejor diferenciación. Sin embargo, dentro del calificador condicional de los sumatorios, el formato de signo de las expresiones condicionales cambia en Lingo. Una prueba de ello está en la función objetivo: Su segundo sumatorio, el que recorre las máquinas con el subíndice j, solo recorrerá las máquinas cuya compatibilidad (A) con el trabajo en cuestión sea apta, es decir igual a 1. Ese signo igual es sustituido por la expression #EQ#. A lo largo del resto de los modelos se irán comentando este tipo de cambios en cuanto vayan apareciendo. [6]

Hay que aclarar también que, a partir de aquí, sí usaremos los subíndices entre paréntesis junto a sus respectivos conjuntos y variables.

Respecto a las actividades de decisión, es donde empiezan a aparecer los @FOR (), los cuales sirven para delimitar a los miembros de un conjunto sobre los que se va a trabajar. La forma de leer la actividad de decisión "Programar trabajos" sería: para todo trabajo i, hay una variable x que representa el instante en el tiempo en el que se programa ese trabajo. El que la variable x esté dentro de un @GIN quiere decir que esa variable será entera. [5,6]

El funcionamiento de los @FOR es muy similar al de los @SUM. Estos también pueden tener calificadores condicionales con las mismas características que los de los @SUM. Los @FOR pueden contener además otros @FOR u otros @SUM, algo que ocurrirrá con mucha frecuencia en las especificaciones.

De la misma forma que el @GIN se refería a una variable entera, el @BIN que aparece al final de la actividad de decision "Asignar máquinas a trabajos", marca la variable alfa como binaria. [5,6]

Al momento de entrar en las especificaciones del grupo Relación implícita de actividades, se puede apreciar que sólo hay dos especificaciones en lugar de las cuatro que había en el modelo matemático original. Esto es debido a qué Lingo no es capaz de procesar las proposiciones lógicas compuestas que utilizan operadores o conectivas lógicas, tales como sí solo sí (SSS), SI-ENTONCES, Y, O, O BIEN, NO. [5,6]

De todas formas, Lingo, solamente con las proposiciones simples del modelo, es decir, las especificaciones que aparecían numeradas al final en el modelo matemático escrito, es capaz de comprender el modelo y simularlo.

En el modelo escrito sí que era necesario deducir y escribir todas las proposiciones compuestas a partir de las proposiciones simples.

Respecto al resto de especificaciones pertenecientes al grupo de solapamiento de trabajos en una máquina, he optado por no incluirlas en el trabajo escrito dada su extensión.

4.3 Versión 2 en Lingo

El segundo problema tiene una estructura muy similar a la del primero en lenguaje Lingo. Así que comentaremos solo las pequeñas variaciones que se producen en él.

```
MODEL:
SETS:
TRABAJOS/1..n/:I,F,d,C,x,B;
MAQUINAS/1..m/:D;
ASIGNAR(TRABAJOS,MAQUINAS):Alfa;
PERIODOS/1..L/:;
CLASES_TRABAJOS/1..CT/:;
CLASES_MAQUINAS/1..CM/:;
COMPATIBILIDAD(CLASES_TRABAJOS,CLASES_MAQUINAS):A;
PROGRAMAR(TRABAJOS,PERIODOS):Beta;
AUXILIAR(TRABAJOS,TRABAJOS,PERIODOS):Omega;
ENDSETS


DATA:

ENDDATA
```

Figura 4-3. MODEL del modelo 2 en Lingo.

Respecto a los conjuntos, se mantienen casi todos, añadiendo el referido al grupo de elementos de los periodos (PERIODOS), el correspondiente a la actividad de decisión programar un trabajo en un periodo determinado (PROGRAMAR) representada con la variable beta, y otro relacionado con una nueva variable auxiliar, omega, que se usará en algunas de las especificaciones de este modelo.

A esta variable auxiliar le ocurre como a las variables auxiliares del modelo anterior. Al tener dos subíndices propios de un mismo elemento, en este caso los trabajos, en su conjunto derivado aparece incluido por partida doble el conjunto primitivo TRABAJOS.

Un detalle importante es la necesidad de incluir conjuntos primitivos, aunque no cuenten con ninguna variable dato propia asociada, para que, además de que así todos los elementos de la tabla de elementos esten representados en los conjuntos, se puedan usar esos conjuntos primitivos en otros derivados que los necesiten, como es el caso de los conjuntos PERIODOS, CLASES_TRABAJOS o CLASES_MAQUINAS.

La sección DATA de nuevo vuelve a estar vacía por los mismos motivos que en el primer modelo. Obtendremos los datos externamente desde la batería de problemas.

```
MAX=@SUM(TRABAJOS(i):@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:B(i)*Alfa(i,j)));

!ACTIVIDADES DE DECISION;

!Programar trabajos en periodos;

@FOR(TRABAJOS(i):@FOR(PERIODOS(t)|t#GE#I(i)#And#t#LE#F(i):@BIN(Beta)));

!Asignar máquinas a trabajos;
@FOR(TRABAJOS(i):@FOR(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:@BIN(Alfa)));

!ESPECIFICACIONES;

!Relacion implicita entre actividades;

@FOR(TRABAJOS(i):@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:Alfa(i,j))= @SUM(TRABAJOS(t):Beta(i,t)));

!Un trabajo procesado por una maquina a lo sumo;

@FOR(TRABAJOS(i):@SUM(MAQUINAS(j)|A(C(i),D(j))#EQ#=1:Alfa(i,j)<=1));
```

Figura 4-4. Función objetivo, actividades de decisión y especificaciones del modelo 2 en Lingo.

Como novedades llamativas respecto al código en Lingo de las actividades de decision, tenemos en el calificador condicional de la actividad Programar trabajos en periodos, la expresión condicional #GE# y la #LE#, las cuales representan a los signos mayor o igual (>=) y menor o igual (<=) respectivamente. Esa misma actividad define también la variable beta como binaria. [5,6]

La última parte del código que ocupan las especificaciones de solapamiento no aparece en el documento por su extensa longitud.

4.4 Versión 3 en Lingo

En el tercer problema apenas tenemos novedades respecto a nuevos símbolos o contenido dentro de su código Lingo que no hayamos visto anteriormente en los otros dos problemas.

```
MODEL:
SETS:
TRABAJOS/1..n/:I,F,d,C,B;
PERIODOS/1..L/:;
CLASES_TRABAJOS/1..CT/:;
CLASES_MAQUINAS/1..CM/:M;
COMPATIBILIDAD(CLASES_TRABAJOS,CLASES_MAQUINAS):A;
PROGRAMAR(TRABAJOS,PERIODOS,CLASES_MAQUINAS):Beta;
ENDSETS

DATA:

ENDDATA
```

Figura 4-5. MODEL del modelo 3 en Lingo.

En relación con la sección SETS, solo destaca la desaparición del conjunto asociado a las máquinas y la inclusión de un tercer conjunto primitivo dentro del conjunto derivado PROGRAMAR, el conjunto CLASES_MÁQUINAS, necesario para el subíndice r de la variable beta.

```
!FUNCION OBJETIVO;

MAX=@SUM(TRABAJOS(i):@SUM(CLASES_MAQUINAS(r)|A(C(i),r)#EQ#1:@SUM(PERIODOS(t)|t#GE#I(i)#And#t#LE#F(i):B(i)*Beta(i,t,r))));

!ACTIVIDADES DE DECISION;

!Programar trabajos en periodos en clases;

@FOR(TRABAJOS(i):@FOR(PERIODOS(t=I(i)..F(i)):@FOR(CLASES_MAQUINAS(r)|A(C(i),r)#EQ#1:@BIN(Beta))));
```

Figura 4-6. Función objetivo y actividades de decisión del modelo 3 en Lingo.

Respecto a la función objetivo, solo añadir que consta de un tercer sumatorio, el que recorrerá los periódos, dentro de los dos anteriores referidos a los trabajos y a las clases de máquinas.

Para finalizar, dentro de las especificaciones de este modelo se pueden encontrar dos variaciones más en el código.

```
!ESPECIFICACIONES;

!Un trabajo procesado por una clase de máquina y un periodo a lo sumo;

@FOR(TRABAJOS(i):@SUM(CLASES_MAQUINAS(r)|A(C(i),r)#EQ#1:@SUM(PERIODOS(t)|t#GE#I(i)#And#t#LE#F(i):Beta(i,t,r)<=1)));

!Solapamiento de trabajos en una maquina;

@FOR(CLASES_MAQUINAS(r):@FOR(PERIODOS(t):@SUM(TRABAJOS(i):@SUM(PERIODOS(tt)|tt#GE#I(i)#And#tt#LE#F(i)#And#tt+d(i)#GT#t:Beta(i,tt,r)<=M(r)))));
END
```

Figura 4-7. Especificaciones del modelo 3 en Lingo.

En la segunda especificación tenemos una nueva expresion condicional, #GT#, la cual se traduce en el signo mayor estricto (>). [5,6] Por otra parte, indicar que, en el segundo sumatorio de esa misma especificación, el de los periodos y luego más adelante en la variable beta, el subíndice t aparece así (tt). Con esto se quiere indicar que es un t prima (t'), es decir un periodo diferente al periodo t. Se escribe así por cuestiones de aceptación del Lingo, el cual no admite la nomenclatura t'.

5 BATERÍA DE PROBLEMAS

5.1 Descripción

La batería de problemas de la que disponemos está compuesta por 1080 archivos de texto. Cada uno representa una clase de problema práctico con todos los datos necesarios para simular en Lingo los tres modelos en los que estamos trabajando.

Estos datos son los que se hubiesen introducido en la sección DATA del código Lingo visto en el capítulo anterior. Sin embargo, como ya explicamos, utilizaremos estos datos de forma externa, sin introducirlos en el código de cada modelo.

El nombre de cada archivo de texto que contiene cada problema es un código numérico del que se puede extraer la información del problema que tiene asignado. Un ejemplo del formato del nombre sería:

File_3_0.8_100_8_3_3_5

He resaltado cada número con un color para su mejor identificación:

- El primer número (ROJO), indica el máximo de la diferencia entre el instante de inicio tardío y el temprano del intervalo que existirá en los trabajos del problema en cuestión, es decir, indica la amplitud máxima que podrán tener las ventanas de comienzos de los trabajos. En el ejemplo es un 3, que marca un máximo de 20 unidades de tiempo. Podrán aparecer también un 1 (máximo de 5 unidades de tiempo), o un 2 (máximo de 10 unidades de tiempo).
- El segundo número (AZUL), se refiere al grado de solapamiento de trabajos en el problema. Dicho de otro modo, la carga de trabajos por unidad de tiempo. En el ejemplo es un 0.8, con lo que en ese problema no habrá apenas trabajos solapados en el tiempo. Las otras dos opciones serían un 1.4 (solapamiento medio) o un 2 (solapamiento alto).
- El tercer número (MORADO), hace referencia a la cantidad de trabajos a procesar en el problema. En el ejemplo existirían 100 trabajos, pero puede haber otros problemas con 25, 50, 200 o incluso 400 trabajos.
- El cuarto número (VERDE), representa el número de máquinas de las que se dispone en el problema. En el ejemplo hay 8 máquinas. En otros problemas podremos tener también 4 o 16 máquinas.
- El quinto número (AMARILLO), muestra cuántas clases diferentes de trabajos hay en el problema. En el ejemplo y en todos los problemas de la batería sólo habrá 3 tipos de trabajos.
- El sexto número (NARANJA) enseña la cantidad de clases de máquinas que aparecen en el problema. En el ejemplo existirán 3 clases de máquinas, aunque en otros problemas podrán tener o 2 o 4 clases.
- El séptimo número (GRIS) es un número de ordenación que marca el puesto del problema dentro de un

grupo de 10 problemas con las características referentes a los anteriores 6 números idénticas. Esos 10 problemas similares están numerados del 0 al 9. En el ejemplo el problema sería el quinto dentro de los de su misma categoría.

A continuación, veremos con otro ejemplo la información contenida dentro de cada problema, es decir, de cada archivo de texto.

```
 25  n trabajos
1 3 80 206 1 inicio_ventana fin_ventana duracion peso clase_trababjo
2 5 69 146 1
3 4 57 297 1
9 13 77 504 1
10 13 79 916 2
10 11 75 257 1
15 17 59 137 1
16 19 69 363 1
18 20 56 680 1
18 20 51 525 3
21 23 43 376 2
28 29 66 862 3
28 31 25 898 1
29 31 64 256 1
31 32 59 925 3
32 35 44 492 1
35 39 57 350 3
41 45 16 332 1
43 44 56 384 3
45 46 16 253 1
51 55 34 281 3
51 53 38 842 3
52 56 32 787 1
68 69 9 172 3
84 85 7 80 2

3 num_clases_trababjo
2 num_clases_maquinas
4 num_maquinas
2 2 n maquinas de cada clase
1  0
0  1
1  1

10 (coste de uso de cada tipo de maquina)
1  (coste de uso de cada tipo de maquina)
```

Figura 5-1. Interior de un archivo de texto de la batería

En la imagen anterior se muestra todo el contenido posible del problema. Primeramente, aparece el número de trabajos que tienen que realizarse (25), y por tanto existen 25 líneas de datos cada una referida a uno de esos trabajos.

Cada línea de datos de un trabajo consta de 5 números.

El primer y el segundo número corresponden a los instantes de inicio y fin de ventana, o, lo que es lo mismo, los instantes más temprano y más tardío para poder procesar un trabajo. Estos dos números definen los intervalos de inicio de cada trabajo.

El tercer número es la duración que supondría el procesamiento de ese trabajo.

El cuarto número es el peso o importancia de realización del trabajo. Recordamos que este dato es clave en el desarrollo de la función objetivo de los tres modelos que estamos estudiando. En la práctica se tendrían que priorizar los trabajos con los pesos más altos.

Y, el quinto número, marca la clase o tipo de cada trabajo. Según este dato, cada trabajo podrá ser procesado en

una u otra máquina.

Más allá de los datos de los trabajos, tenemos otros datos generales del problema como son el número de clases de trabajo, el cual estaba fijado para toda la batería en 3 clases exclusivamente.

También se muestran el número de máquinas disponibles, la cantidad de clases de máquinas que existen en el problema, y, a la vez, el número de máquinas pertenecientes a cada clase.

Por otra parte, tenemos la matriz de compatibilidad, de la cuál veiamos ejemplos en el segundo capítulo de este proyecto. Esta será clave para delucidar que clases de máquinas son adecuadas para qué tipos de trabajos. Siempre serán matrices de 3 filas por el hecho de que siempre tendremos 3 clases de trabajos únicamente. Sus columnas, sin embargo, variarán según cuántos tipos de máquinas haya en cada problema. En este ejemplo son 2, por lo que queda una matriz de compatibilidad 3x2.

Recordemos que cada 1 de la matriz significaba la existencia de compatibilidad máquina-trabajo, y cada 0, lo contrario.

Por último, aparecen los costes que supone el uso de cada máquina disponible cada vez que procese un trabajo. Este último dato está pensando para modelos que se quieran resolver siguiendo un objetivo táctico. En estos casos se partía de la premisa de que había que procesar todos los trabajos pendientes utilizando el menor número de máquinas posibles para ello.

Sin embargo, como para nuestros tres modelos estamos siguendo un objetivo operacional distinto para cada uno, en los cuales mantenemos un número de máquinas fijo, este último dato de coste de uso de las máquinas no es rrelevante y no lo tendremos en cuenta en la posterior simulación de los modelos.

5.2 Obtención

Para poder conseguir el formato adecuado para introducir el resto de problemas de la batería en los modelos de Lingo, hemos desarrollado programa mediante anaconda jupyter, el cual se vale del lenguaje python para su programación.

El objetivo principal del programa es dejar preparado con el formato adecuado para Lingo la parte en la que se rellenan las cinco principales características de los trabajos (II, F, d, B, C) con todos los datos procedentes del archivo de texto que contiene a cada problema. Si no se hiciese esto, sería prácticamente interminable pasar manualmente todos los datos para los problemas con 100, 200 o 400 trabajos.

El resto de datos que aparecen en cada archivo de texto, como pueden ser el número total de trabajos, la matriz de compatibilidad o el número de clases de trabajos o máquinas, simplemente se copiarán igual, ya que no supone mucho esfuerzo ir cambiándolos a la hora de la posterior simulación en Lingo.

El algorítmo en cuestión, está dividido en 3 celdas, cada una con una función determinada:

- Celda de lectura. Desde aquí vamos leyendo los distintos ficheros que corresponderán a cada problema de la batería que queramos convertir. Sólo tendremos que cambiar el nombre del fichero cada vez. Con la función readlines se consigue que se lea cada línea del fichero por separado. Como en la primera línea del archivo de texto se encuentra el número de problemas total del problema, inicializamos un entero "n" con ese valor que ya se ha leído. Con esto conseguimos que el algoritmo sepa hasta cuantas líneas tiene que leer para completar los vectores de las cinco características que comentábamos al principio de este subapartado.

  Dividimos ahora el fichero en dos partes, una para esas cinco características especificamente (Array) y otra para el resto de datos externos (Array2). Array llega hasta el ultimo trabajo "n" más 1 debido a que hay un espacio entre el primer número del fichero y la primera fila de 5 datos. A partir de ahí empieza a rellenarse Array 2 con el resto de datos externos hasta la última línea que haya leído la función readlines.

  Acto seguido, creamos cinco listas, una para cada característica. Luego mediante un bucle for vamos introduciendo el primer número (x) de cada línea en la primera lista (II), el segundo número en la segunda lista y así sucesivamente con cada línea.

Por último, recorremos en otro bucle for el resto de números (y) del fichero y eliminamos los saltos de línea.

```
f = open("Baterias(1)/File_3_2_400_16_3_4_4.txt", "r")
Lines = f.readlines()
n = int(Lines[0]);
Array = Lines[1:n]
Array2 = Lines[n+1:len(Lines)]
II = []
F = []
d = []
B = []
C = []
Lines.pop(0)
for x in Array:
    II.append(x.split(' ')[0])
    F.append(x.split(' ')[1])
    d.append(x.split(' ')[2])
    B.append(x.split(' ')[3])
    C.append(x.split(' ')[4])

for y in Array2:
    y.replace('\n', '')
```

Figura 5-2. Celda de lectura del algoritmo

- Celda de creación. En esta parte crearemos una cadena (string) para cada característica, e iremos introduciendo los valores que teníamos en sus respectivas listas, pero ahora irán separados por comas, algo necesario para su introducción en Lingo. Esto lo conseguimos mediante las funciones join. Es necesario también que dentro de cada función join se realize otro bucle para que la coma se introduzca entre todos los números de la cadena representante de una característica.

  Para la cadena de la carácteristica clase (c) hay que realizar un preproceso antes de introducir las comas al ser el ultimo elemento del Array. De nuevo hay que eliminar los saltos de línea ya que esta característica es la última de cada línea en el fichero inicial y también lo llevaban.

```
stringII = ','.join([str(item) for item in II])
stringF = ','.join([str(item) for item in F])
stringd = ','.join([str(item) for item in d])
stringB = ','.join([str(item) for item in B])
listC = []
for c in C:
    c.replace('\n', '')
    listC.append(c.replace('\n',''))
    #print(c)
stringC = ','.join([str(item) for item in listC])
```

Figura 5-3. Celda de creación del algoritmo

- Celda de escritura. Aquí generamos un nuevo fichero con el mismo nombre que tenga el problema que estemos convirtiendo y sobre el que vamos a escribir mediante la función write. En primer lugar, convertimos el entero "n" a un string mediante la función "str". Posteriormente escribimos el número de trabajos (n) seguido de un salto de línea. A continuación, escribimos las cinco cadenas creadas en la celda anterior, precedidas de su respectivo símbolo y un igual, y delimitadas por un ";". Así conseguimos que en cada fichero que se vaya creando se mantenga la estructura de datos apropiada para Lingo.

  De nuevo, también escribiremos el resto de números almacenados en Array2, sustituyendo los saltos de

línea por espacios en blanco.

```
with open('Modificaciones/File_3_2_400_16_3_4_4.txt', 'w') as f:
    f.write(str(n))
    f.write('\n')
    f.write('II = ' + stringII + ';')
    f.write('\n')
    f.write('F = ' + stringF + ';')
    f.write('\n')
    f.write('d = ' + stringd + ';')
    f.write('\n')
    f.write('B = ' + stringB + ';')
    f.write('\n')
    f.write('c = ' + stringC + ';')
    f.write('\n')
    for y in Array2:
        y.replace('\n', '')
        f.write(y)
```

Figura 5-4. Celda de escritura del algoritmo

6 SIMULACIÓN Y ANÁLISIS DE LOS RESULTADOS

En este capítulo realizaremos primeramente una simulación manual de los 3 modelos, tras la que comentaremos y explicaremos los resultados obtenidos. Acto seguido, recogereremos los resultados de una serie de distintos casos de la batería de problemas a través del código generador de problemas que programamos al final del capítulo anterior.

6.1 Simulación manual

El motivo de esta simulación previa es introducir a mano los datos de uno de los problemas de la batería al azar en la sección DATA de los modelos en Lingo, la cual, recordemos, dejamos vacía en el capítulo 4, y en menor medida en la sección SETS, para mostrar cómo quedaría el código completo en Lingo para cada modelo.

Otro motivo es el de explicar el cuadro de resultados que ofrece Lingo una vez se corre el modelo en el programa, y empezar a comentarlos. Así, en el momento de adentrarnos en la siguiente simulación, en la que corremos grupos de problemas más grandes gracias al código generador de archivos Lingo, podremos centrarnos en el objetivo final de este trabajo, que no es otro que el de estudiar los tiempos de simulación de cada modelo para ver cual de ellos es el más potente, es decir, el que menos tiempo necesita para encontrar el resultado óptimo.

Para mayor facilidad, hemos escogido un problema de la batería de tan solo 25 trabajos. El nombre del archivo que lo contiene es el "File_1_0, 8_25_4_3_2_2.txt". Del capítulo 5 sabemos que será un problema de intervalos de inicio para cada trabajo no superiores a 5 unidades de tiempo, con un nivel bajo de solapamiento entre ellos. Contaremos con cuatro máquinas que podrán ser de dos clases distintas y, como siempre, los trabajos podrán ser de tres clases.

Los datos que contiene son los que aparecen a continuación:

```
 25
1 4 77 130 1
1 3 71 303 2
2 4 19 977 2
7 9 9 234 3
9 13 12 179 3
9 11 76 21 1
10 14 70 354 1
11 13 45 756 3
13 16 60 624 1
15 19 34 106 2
18 21 29 854 1
19 20 78 33 2
22 23 56 332 3
23 26 36 677 3
29 31 53 451 3
29 32 51 856 2
30 34 48 365 2
31 32 53 479 3
31 32 68 338 3
44 47 14 564 1
46 49 27 34 1
46 50 29 457 3
50 53 4 242 2
61 62 27 140 3
65 69 8 933 3

3
2
4
2  2
1  0
0  1
1  1

6
9
```

Figura 6-1. Datos de un problema de la batería

Con ellos ya podremos completar las secciones DATA y SETS de los modelos. Para los modelos 1 y 2, la forma de introducir esos datos será idéntica, por lo que solo mostraremos el modelo 1. Respecto a la sección SETS, los cambios están en los números que marcan el tamaño de los conjuntos simples, los cuales sustituyen a las constantes que los representaban en los capítulos anteriores con los modelos incompletos.

Por ejemplo, sabemos que tenemos 25 trabajos, con lo que la constante que marcaba el tamaño del conjunto TRABAJOS será igual a 25. Lo mismo ocurre con los tamaños de los conjuntos CLASES_TRABAJOS y CLASES_MAQUINAS, de los que ahora conocemos sus tamaños. Como hay tres clases de trabajos, CT valdrá 3 y, como hay dos clases de máquinas, CM será igual a 2.

Por ultimo, el tamaño del conjunto PERIODOS lo obtenemos fijándonos en el posible instante de inicio más tardío del ultimo trabajo a procesar (el trabajo 25), el instante 69. Esto se debe a que, si cada periodo t es el momento en el que finalmente se realiza cada trabajo, el instante t=69 será el ultimo posible ya que el resto de trabajos se realizan antes.

En la sección DATA, vamos introduciendo los 25 valores de cada parámetro asociados cada uno a uno de los 25 trabajos disponibles. Con ello completamos todos los inicios tempranos (I) y tardíos (F) de programación, duraciones (d), pesos (B), clases (C) y compatibilidades (A) posibles.

Por otro lado, para el parámetro D, que indicaba la clase a la que pertenecía cada máquina disponible, como sabemos que hay cuatro máquinas en total en este problema y que hay 2 de un tipo y dos de otro, rellenamos sus 4 valores con dos 1 y dos 2.

```
SETS:
TRABAJOS/1..25/:II,F,d,C,x,B;
MAQUINAS/1..4/:DD;
ASIGNAR(TRABAJOS,MAQUINAS):Alfa;
CLASES_TRABAJOS/1..3/;
CLASES_MAQUINAS/1..2/;
COMPATIBILIDAD(CLASES_TRABAJOS,CLASES_MAQUINAS):A;
AUXILIAR(TRABAJOS,TRABAJOS,MAQUINAS):pi;
AUXILIAR2(TRABAJOS,TRABAJOS):Sigma;
ENDSETS


DATA:
II=1,1,2,7,9,9,10,11,13,15,18,19,22,23,29,29,30,31,31,44,46,46,50,61,65;
F=4,3,4,9,13,11,14,13,16,19,21,20,23,26,31,32,34,32,32,47,49,50,53,62,69;
d=77,71,19,9,12,76,70,45,60,34,29,78,56,36,53,51,48,53,68,14,27,29,4,27,8;
B=130,303,977,234,179,21,354,756,624,106,854,33,332,677,451,856,365,479,338,564,34,457,242,140,933;
C=1,2,2,3,3,1,1,3,1,2,1,2,3,3,3,2,2,3,3,1,1,3,2,3,3;
DD=1,1,2,2;
A=1 0
  0 1
  1 1;

ENDDATA
```

Figura 6-2. DATA y SETS del modelo 1

Respecto al modelo 3, tenemos una sección SETS con los mismos tamaños que los dos modelos anteriores y una sección DATA también idéntica, aunque hay que añadir los valores del parámetro M, que indicaba cuantas máquinas se contabilizaban en cada clase. Como hay dos clases de máquinas y dos máquinas de cada clase, sus valores serán 2 y 2.

```
DATA:
II=1,1,2,7,9,9,10,11,13,15,18,19,22,23,29,29,30,31,31,44,46,46,50,61,65;
F=4,3,4,9,13,11,14,13,16,19,21,20,23,26,31,32,34,32,32,47,49,50,53,62,69;
d=77,71,19,9,12,76,70,45,60,34,29,78,56,36,53,51,48,53,68,14,27,29,4,27,8;
B=130,303,977,234,179,21,354,756,624,106,854,33,332,677,451,856,365,479,338,564,34,457,242,140,933;
C=1,2,2,3,3,1,1,3,1,2,1,2,3,3,3,2,2,3,3,1,1,3,2,3,3;
M=2,2;
A=1  0
  0  1
  1  1;

ENDDATA
```

Figura 6-3. DATA del modelo 3

En el momento en el que ya están introducidos todos los datos necesarios y el resto del código del modelo en Lingo no contiene errores, ya podemos proceder con la simulación.

Una vez ejecutamos el Solver en cada modelo, aparecerán cuadros de resultados de la simulación como el que vemos a continuación:

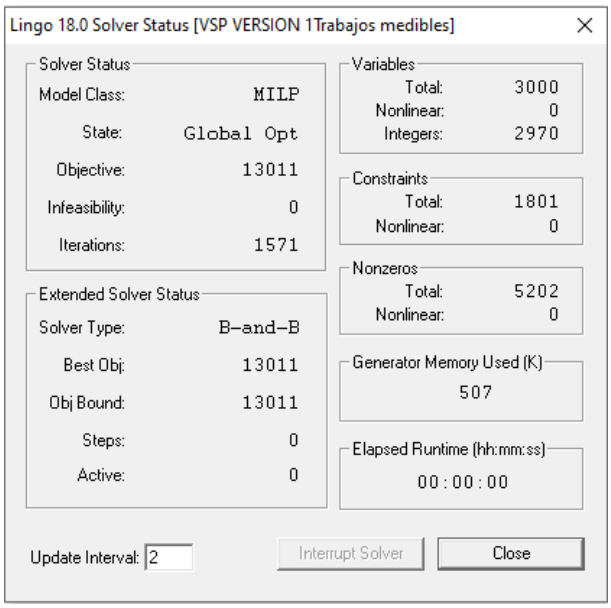

Figura 6-4. Lingo. Solver Status. Versión 1

Este en concreto pertenece al primero de los tres modelos. Está dividido en varios bloques:

- En el bloque variables se encuentra el total de variables con las que cuenta el modelo (3000). Dentro de esas variables, indica cuales son no lineales (0) o enteras (2970). Al no existir variables no lineales en el modelo (aquellas que cuentan con alguna relación no lineal en una restricción determinada) en proporción al total de variables, se consigue resolver el modelo de forma más rápida. [6]

- Existe un bloque referido a las restricciones (constraints), en el que aparece la cantidad final de las mismas, junto al número de restricciones no lineales de entre ellas. [6]

- El bloque nonzeros enseña cuantos coeficientes no ceros contabiliza el modelo y cuantos de ellos son no lineales. [6]

- También se indica la capacidad de memoria que usa el programa Lingo para resolver el modelo (507 K) en el bloque Generator Memory Used. [6]

- En el bloque Solver Status aparecen la clase de modelo que se resuelve y el tipo de óptimo que se calcula, en este caso un óptimo global (aquel que es la mejor solución posible) junto a su valor numérico (13011). Otros datos presentes en este bloque son la cantidad por la cual todas las restricciones han sido violadas (infeasibility), y las iteraciones (iterations), que señalan cuántas veces ha recorrido el software Lingo el modelo (0) para su resolución. [6]

- El bloque Extended Solver Status contiene la forma en la que se ha resuelto el modelo. Para esta ocasión ha optado por el método de eliminación Branch and Bound (B-and-B). Por otro lado, obtenemos de aquí el mejor objetivo alcanzado hasta el momento (Best Obj) y el número que representa la distancia entre el objetivo en el que se encuentra el procesador Lingo en determinado momento y el mejor objetivo que se haya encontrado hasta ese instante (Obj Bound). Ambos números son idénticos en este caso (13011), debido a que el modelo ya ha sido completamente resuelto en la simulación. [6]

- Por último y más importante, contamos con el bloque Elapsed Runtime, el cual indica la duración que ha necesitado el software para simular el modelo en horas, minutos y segundos. No tiene que marcar el

tiempo total, ya que mientras se esté simulando, existe la posibilidad de parar la simulación, lo que llevará a que este bloque marque el tiempo justo en el que se para. En este caso en particular, la simulación ha sido inmediata. [6]

Además de este cuadro de simulación, al resolver el modelo también aparece otra pantalla extra:

Variable	Value	Reduced Cost
E	9.000000	0.000000
II(1)	1.000000	0.000000
II(2)	1.000000	0.000000
II(3)	2.000000	0.000000
II(4)	7.000000	0.000000
II(5)	9.000000	0.000000
II(6)	9.000000	0.000000
II(7)	10.00000	0.000000
II(8)	11.00000	0.000000
II(9)	13.00000	0.000000
II(10)	15.00000	0.000000
II(11)	18.00000	0.000000
II(12)	19.00000	0.000000
II(13)	22.00000	0.000000
II(14)	23.00000	0.000000
II(15)	29.00000	0.000000
II(16)	29.00000	0.000000
II(17)	30.00000	0.000000
II(18)	31.00000	0.000000
II(19)	31.00000	0.000000
II(20)	44.00000	0.000000
II(21)	46.00000	0.000000
II(22)	46.00000	0.000000
II(23)	50.00000	0.000000
II(24)	61.00000	0.000000
II(25)	65.00000	0.000000
F(1)	4.000000	0.000000
F(2)	3.000000	0.000000
F(3)	4.000000	0.000000
F(4)	9.000000	0.000000
F(5)	13.00000	0.000000
F(6)	11.00000	0.000000
F(7)	14.00000	0.000000
F(8)	13.00000	0.000000
F(9)	16.00000	0.000000

Figura 6-5. Valores de los datos y variables

En ella vienen recogidos todos los valores de los datos del problema y de las variables que se han ido activando o no, en la columna valor (Value). [6]

La última columna, denominada coste reducido (Reduced cost), se refiere a la cantidad por la que el coeficiente objetivo de cada variable debería de mejorar para que empezara a aportar un valor positivo a la solución óptima. [6]

Finalmente, incluiremos además los cuadros de simulación de los otros dos modelos para este problema:

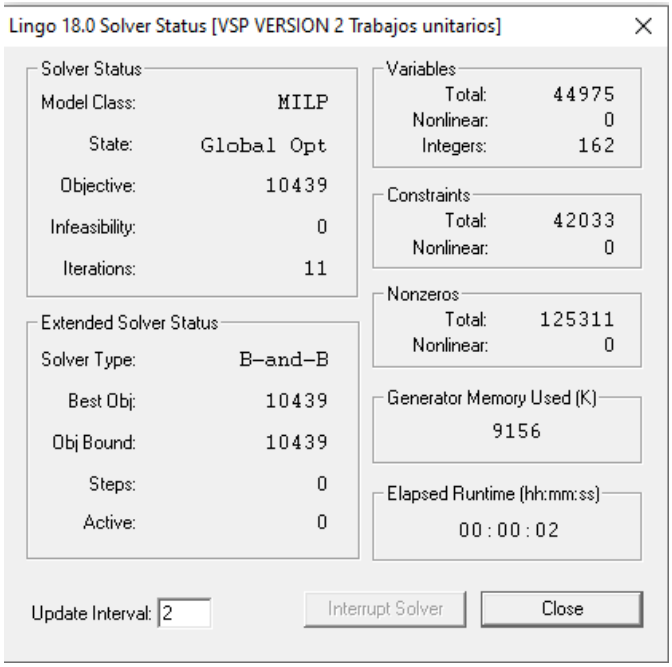

Figura 6-6. Lingo. Solver Status. Versión 2

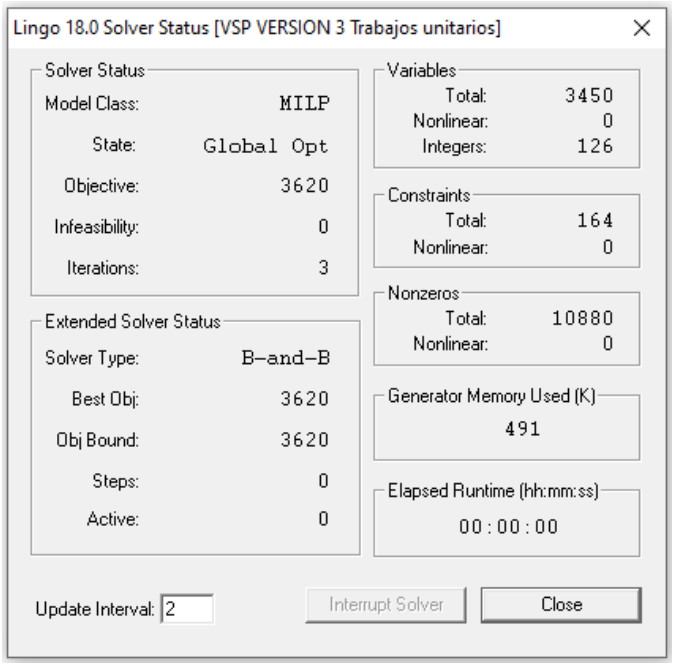

Figura 6-7. Lingo. Solver Status. Versión 3

Cabe resaltar que, para este problema, los tres modelos poseen un tiempo de resolución ínfimo. Esto es debido a que el problema simulado consta sólo de 25 trabajos, algo que si se hiciese a mano resultaría tedioso, pero para un software como Lingo es muy sencillo de resolver. De los tres modelos, el tercero se realiza prácticamente de forma inmediata, gracias a que tiene una estructura mucho menos compleja que los otros dos. Más adelante en cuanto resolvamos problemas que tengan una mayor cantidad de trabajos a procesar veremos como los tiempos de simulación aumentarán.

6.2 Simulación de la batería de problemas

Una vez realizada la primera simulación a mano, procederemos a simular un grupo concreto de problemas de la batería, en concreto 135 problemas. Se ha intentado que en su mayoría sean problemas con características principales distintas, para explorar la mayor cantidad de combinaciones diferentes posibles, y que cada tamaño de problema según el número de trabajos (25, 50, 100, 200, 400) tenga la misma representación.

En la tabla que veremos en este subcapítulo, están recogidos los números que marcan algunas de las principales características de cada uno de los problemas simulados, junto a los tiempos de simulación que supone procesarlos en cada uno de los modelos estudiados. Añadir también, que solo incluiremos en este capítulo una pequeña parte de la tabla por su extensión, con el objetivo de explicar todos sus componentes. La tabla aparece completa en el anexo de este trabajo para su visualización.

Como características diferenciales de los problemas tenemos: en la primera columna, la amplitud de los intervalos, en la segunda, el solapamiento de los trabajos, en la tercera, el número de trabajos a procesar, en la cuarta, el número de máquinas disponibles y, en la quinta, el número de clases de máquinas que hay. Hemos obviado el número de clases de trabajos, ya que siempre será 3, y el número que diferenciaba a cada problema de los de su mismo subgrupo, porque se ha intentado simular en su mayoría problemas de diferentes subgrupos.

Por otro lado, en las últimas tres columnas, aparecen los tiempos de simulación del primero modelo (Tiempo V1), del segundo (Tiempo V2) y del tercero (Tiempo V3) para cada problema. Los tres se medirán en segundos.

Amplitud	Solapamiento	Nº Trabajos	Nº Máq	Clases máq	Tiempo V1	Tiempo V2	Tiempo V3
1	0,8	25	4	2	0	1	0
1	0,8	25	4	2	1	2	1
1	0,8	25	4	2	2	1	1
1	0,8	50	4	2	0	3	1
1	0,8	50	4	2	1	4	1
1	0,8	50	8	3	3	10	3
1	0,8	100	4	2	8	21	2
1	0,8	100	8	3	16	33	5
1	0,8	100	16	4	14	55	5
1	0,8	200	4	2	266	76	5
1	0,8	200	8	3	*	123	10
1	0,8	200	16	4	40	240	11
1	0,8	400	4	2	523	*	11
1	0,8	400	8	3	*	*	16
1	0,8	400	16	4	*	*	22
1	1,4	25	4	2	0	1	0
1	1,4	25	4	2	1	2	2
1	1,4	25	4	2	0	3	1
1	1,4	50	4	2	3	5	2
1	1,4	50	4	2	5	6	1
1	1,4	50	8	3	16	10	2

Tabla 6-1. Muestra de la batería de problemas

Como hemos explicado antes, este solo es el inicio de la tabla. A continuación, comentaremos los resultados obtenidos en la totalidad de la tabla incluida en el anexo.

Antes de nada, señalar que durante las simualciones se ha establecido un límite de 900 segundos (15 minutos). En el momento en el que alguno de los problemas de la batería tardara más de ese tiempo en ser resuelto por Lingo, se paraba automáticamente la simulación marcando un asterisco (*) en la tabla de resultados. Esta limitación se ha impuesto debido a que, en muchos casos, la mayor parte problemas de 200 o 400 trabajos, los tiempos de simulación se dilataban varias horas, lo que impedía tener una cierta continuidad, necesaria para simular manualmente tal cantidad de problemas (135 problemas en cada uno de los tres modelos).

Recalcar que, aunque los problemas que tengan 900 segundos de duración, no sea realmente su duración exacta, por lo menos, al momento de cortar la simulación, en el cuadro de simulación obtenido aparecía un estado factible, es decir, que había encontrado en ese tiempo una solución que respetase todas las restricciones del modelo donde se simulase. Para el resto de problemas cuya duración es menor al límite de 900 segundos, el estado en el cuadro de simulación marcaba óptimo global, o, lo que es lo mismo, la certeza de que se ha encontrado la mejor solución posible para cada uno de ellos.

Una vez planteadas las condiciones que hemos elegido para realizar la simulación podemos pasar a comparar los resultados obtenidos entre los 3 modelos.

A primera vista, se puede apreciar fácilmente que el tercer modelo es el más veloz en simular la totalidad de los problemas. En ningún caso se ve superado por alguno de los otros dos modelos, como mucho igualado en los problemas de menor tamaño. De hecho, todas sus simulaciones han llegado a su óptimo global correspondiente. En ningún momento se supera el límite que hemos impuesto. Al igual que los otros dos modelos, este tercer modelo va aumentando su duración conforme los problemas van teniendo mayor número de trabajos y máquinas, aunque tiene como mayor tiempo total tan solo 26 segundos, nada comparado a sus dos modelos competidores que veremos a continuación.

El primer modelo y el segundo poseen un comportamiento y unos resultados muy similares. Ambos consiguen resolver de forma casi inmediata los problemas con 25 trabajos y con duraciones muy pequeñas los de 50 trabajos. Sin embargo, los resultados obtenidos por el modelo 1 son algo mejores que los generados por el modelo 2 para estos dos tamaños de problema. En cuanto a los problemas de 100 trabajos, se puede observar que en algunos de ellos se empieza a superar el límite de tiempo impuesto, en especial en el modelo 1 y en menor medida en el segundo. A pesar de estos todavía hay una cantidad aceptable de caso que sí encuentrar su óptimo global. Para los problemas de mayor tamaño, los de 200 y 400 trabajos, apreciamos que en su mayoría incumplen el límite. Esto nos deja la certeza de que ambos modelos (1 y 2) no serían eficientes a la hora de resolver problemas tan extensos.

Asimismo, a raíz de la tabla de resultados principal del anexo hemos generado una nueva tabla que adjuntamos a continuación.

MEDIA		Tiempo V1	Tiempo V2	Tiempo V3
Amplitud	1	57	44	6
	2	19	66	6
	3	74	59	7
Solapamiento	0,8	46	40	6
	1,4	79	59	7
	2	29	67	6
Nº Trabajos	25	0	4	0
	50	8	20	2
	100	35	82	4
	200	189	206	9
	400	211	*	18
Nº Máquinas	4	28	18	3
	8	85	59	8
	16	117	221	14

Tabla 6-2. Promedios de los tiempos de simulación

En ella hemos recogido los promedios de los tiempos de simulación de cada modelo en función de cada una de las características principales, con el propósito de compararlos de una forma más concreta y precisa.

No está incluida la característica que indica el número de clases de máquinas, ya que coincidiría con las medias de los tiempos respecto al número de máquinas. Tampoco hemos tenido en cuenta todos los problemas que han sobrepasado el límite de los 15 minutos. De ahí que aparezca otro (*) en el promedio de los problemas de 400 trabajos para el modelo 2, ya que ninguno en ninguno de los problemas ejecutados de ese tipo hemos llegado a encontrar el óptimo global.

Si clasificamos los problemas únicamente según su amplitud, podemos ver que el modelo 3 se comporta de forma uniforme y con medias relativamente bajas para cualquier tipo de amplitud de los intervalos asignados a los trabajos. Recordemos que los intervalos de los problemas con amplitud “1” tenían como máximo 5 unidades temporales, los de amplitud “2”, 10 unidades y los de amplitud “3”, 20 unidades. Sería lógico pensar antes de ver los resultados, que en los casos en los que tengamos intervalos más extensos será mas sencillo programar los trabajos al existir más posibilidades de inicio. Sin embargo, tanto en el modelo 1 como en el 2 la duración media de los trabajos de amplitud “3” no es la mas baja.

El solapamiento se medía a partir de tres niveles de carga (0,8;1,4;2). En orden creciente, señalaban la probabilidad de que coincidiesen los trabajos en el tiempo al procesarse. En teoría, el pensamiento antes de simular tendría que ser que cuanto más grado de solapamiento, más debería durar la simulación, al ser más complicado encontrar espacio suficiente para cada trabajo. Observando ahora los resultados generados, nos damos cuenta de que para el modelo 2 la suposición se verifica. Conforme mayor es el grado de solapamiento, más altas son las medias ($40<59<67$). En el modelo 1, por el contrario, la duración promedio para los problemas con más grado de solapamiento es la menor (29). Por otra parte, en el modelo 3 los resultados se mantienen estables para todos los grados, aunque, de nuevo, los tiempos del modelo 3 siguen siendo ínfimos en comparación a los otros.

Acto seguido, nos adentramos en la división de los problemas según su número de trabajos a procesar total. Las medias obtenidas según esta clasificación vienen a representar lo mismo que comentábamos en la tabla de resultados individuales de cada trabajo. Los problemas pequeños de 25 trabajos son simulados inmediatamente en los modelos 1 y 3, seguidos por el modelo 2 que suele tardar unos pocos segundos más. Para los problemas de 50 trabajos se mantiene el mismo orden de eficiencia, aunque el modelo 3 empieza a destacar sobre el 1. Luego vemos que para los de 100 trabajos, el modelo 1 es mucho más eficiente que el 2 ($35<82$). El modelo 3 sigue en su línea de duraciones muy pequeñas. Con respecto a los problemas de 200 y 400 se confirma el hecho de que que para los dos primeros modelos se incumpla tantas veces el límite establecido El modelo 3 por su parte sigue siendo muy capaz de simularlos en tiempo record.

El hecho de que los tiempos en los tres modelos aumenten según la cantidad de trabajos es razonable. En una situación real, como en un centro de logística, cuantas más cargas de trabajos a realizar haya, mas difícil será organizarlo todo para que entren todos ellos a tiempo dentro de sus intervalos y sin que se sobrecarguen los recursos que los procesen. Fijandonos también en el propio programa Lingo, en él se producirán muchas más variables que generar en proporción al número de trabajos, lo que aumentará el tiempo total.

Y si nos centramos en el número de máquinas y, por consiguiente, en el número de clases de máquinas, tenemos una clara diferencia en los tres modelos. Para los problemas con menos máquinas (4) todos los tiempos serán reducidos e irán siendo más elevados para los de 8 y 16 máquinas. Cabría pensar de antemano que con más maquinas a nuestra disposición sería mas fácil procesar todos los trabajos, pero también hay que tener en cuenta que se volverán a generar muchas más variables en Lingo en los casos con 16 máquinas. Es por esto que los tiempos van aumentando. Con respecto a los modelos, el modelo tres vuelve a ser óptimo en esta categoría. El modelo 2 se coloca mejor que el 1 si contamos con 4 u 8 máquinas y el modelo 1 será superior al 2 en problemas con 16 máquinas.

7 CONCLUSIONES

Para concluir con este proyecto, en este último capítulo señalaré las ideas fundamentales extraíbles del mismo.

El interés por este trabajo surge cursando la asignatura Métodos Cuantitativos de Gestión, impartida por el tutor de mi TFG, D. José Manuel García Sánchez, en el grado de Ingeniería de Organización Industrial. Los contenidos vistos en ella me han servido de guía para la explicación de los tres modelos propuestos en el tercer capítulo de este trabajo, en el cual aparecen descritos y formulados. Asímismo, me proporcionó una base teórica de Lingo para su posterior implementación en el cuarto capítulo.

El objetivo del trabajo ha sido desarrollar teóricamente la programación de trabajos en intervalos y los modelos VSP, implentar en Lingo y valorar los resultados obtenidos de la resolución de estos tres modelos a partir de los datos de los problemas de la batería, para acabar hallando el que se adaptase mejor en términos de rapidez y eficiencia.

En la parte teórica se parte de tres modelos matemáticos, los cuales mantienen un objetivo operacional, basado en maximizar el peso que supone la realización de cada trabajo. He expuesto los elementos, las actividades de decisión, las especificaciones y las funciones objetivo de cada modelo.

En lo referente a la valoración de las librerías de optimización Lingo, puedo decir que han sido capaces de interaccionar correctamente con los modelos basados en la variante VSP de la programación operacional de trabajos en intervalos, teniendo en cuenta que los problemas ejecutados de la batería para cada modelo se han resuelto, en su mayoría, en un tiempo razonablemente bajo. He de destacar también la sencilla adaptación de los modelos mátematicos a su versión en Lingo.

Respecto a los problemas que no se han resuelto dentro del límite de tiempo marcado, debo insistir en que esto se ha producido por su gran cantidad de datos, junto al hecho de que han sido ejecutados en modelos muy extensos, con muchas especificaciones y variables, como sucede en los modelos 1 y 2. Por ello, no recomendaría usar Lingo en casos en los que se den simultáneamente ambas características.

Tengo que resaltar también la importancia del algoritmo en lenguaje Python, necesario para implementar los datos procedentes de los problemas de la batería en Lingo. Sin él, el proceso de implementación y posterior ejecución de los problemas hubiese sido de una duración excesiva.

En resumen, he realizado un estudio en esta memoria sobre tres modelos distintos de tipo VSP y su comportamiento frente a un número considerable de problemas, y podemos confirmar, tras evaluar los resultados

obtenidos en las simulaciones, que el modelo 3 es el más eficiente y rápido en resolver todo tipo de problemas. Su diferencia en potencia respecto a los otros dos es bastante amplia y es el que elegiríamos para encontrar las soluciones óptimas en problemas de programación de intervalos variables. El hecho de que el modelo 3 haya sido capaz de resolver todos los problemas en poco tiempo se debe a su sencilla estructura en su versión matemática, donde se vale de unas pocas especificaciones y variables para formular un problema muy similar al que proponen sus dos modelos alternativos.

Por otro lado, nos quedaríamos con el modelo 2 en el caso de que contemos con pocas máquinas para tratar los trabajos, y con un grado de solapamiento bajo entre esos trabajos. Para el resto de casos escogeríamos siempre el modelo 1, el cual es superior al 2 con respecto a la mayoría de características o parámetros.

.

REFERENCIAS

[1] García, J.M., Programación de operaciones en sistemas productivos y logísticos. Ingeniería de organización. modelos y aplicaciones. Ed. DIAZ DE SANTOS. (ISBN: 978-84-7978-847-6) pp 29- 76 (2008)

[2] Mercedes Morales Blázquez, Análisis táctico en Gestión de Proyectos mediante Programación de Trabajos en Intervalos. Trabajo de fin de grado.

[3] María Eugenia Morell González, Nuevos escenarios de aplicación de la programación de trabajos en intervalos. Proyecto fin de carrera Ingeniero Industrial. Universidad de Sevilla

[4] Jose Carlos Cauto, R*esolución exacta y heurística de problemas de programación táctica de trabajos en intervalos*, 2014. Proyecto fin de carrera. Ingeniería de organización industrial. Universidad de Sevilla

[5] Jose Manuel García Sanchez, 2021. "*Modelling in Mathematical Programming*," International Series in Operations Research and Management Science, Springer, number 978-3-030-57250-1

[6] Erica Canizo, Paola Lucero, Investigación operativa 2002 Software Para Programación Lineal - LINGO/LINDO-. Universidad tecnológica nacional, Mendoza, Argentina.

ANEXO

Tabla de resultados de la simulación de la batería de problemas

Amplitud	Solapamiento	Nº Trabajos	Nº Máq	Clases máq	Tiempo V1	Tiempo V2	Tiempo V3
1	0,8	25	4	2	0	1	0
1	0,8	25	4	2	1	2	1
1	0,8	25	4	2	2	1	1
1	0,8	50	4	2	0	3	1
1	0,8	50	4	2	1	4	1
1	0,8	50	8	3	3	10	3
1	0,8	100	4	2	8	21	2
1	0,8	100	8	3	16	33	5
1	0,8	100	16	4	14	55	5
1	0,8	200	4	2	266	76	5
1	0,8	200	8	3	*	123	10
1	0,8	200	16	4	40	240	11
1	0,8	400	4	2	523	*	11
1	0,8	400	8	3	*	*	16
1	0,8	400	16	4	*	*	22
1	1,4	25	4	2	0	1	0
1	1,4	25	4	2	1	2	2
1	1,4	25	4	2	0	3	1
1	1,4	50	4	2	3	5	2
1	1,4	50	4	2	5	6	1
1	1,4	50	8	3	16	10	2
1	1,4	100	4	2	16	19	3
1	1,4	100	8	3	*	24	4
1	1,4	100	16	4	42	66	6
1	1,4	200	4	2	*	76	5
1	1,4	200	8	3	*	93	8
1	1,4	200	16	4	415	223	11
1	1,4	400	4	2	*	*	11
1	1,4	400	8	3	*	*	17
1	1,4	400	16	4	*	*	22
1	2	25	4	2	0	1	0
1	2	25	4	2	0	1	1
1	2	25	4	2	0	1	0
1	2	50	4	2	6	4	1
1	2	50	4	2	6	4	1
1	2	50	8	3	5	10	2
1	2	100	4	2	40	21	2

1	2	100	8	3	*	22	4
1	2	100	16	4	*	46	5
1	2	200	4	2	*	78	6
1	2	200	8	3	*	91	8
1	2	200	16	4	231	203	11
1	2	400	4	2	*	*	11
1	2	400	8	3	*	*	16
1	2	400	16	4	*	*	22
2	0,8	25	4	2	1	2	0
2	0,8	25	4	2	0	3	0
2	0,8	25	4	2	0	4	0
2	0,8	50	4	2	2	12	1
2	0,8	50	4	2	4	9	1
2	0,8	50	8	3	3	24	3
2	0,8	100	4	2	6	35	2
2	0,8	100	8	3	23	39	4
2	0,8	100	16	4	12	95	5
2	0,8	200	4	2	84	*	6
2	0,8	200	8	3	*	*	9
2	0,8	200	16	4	17	*	11
2	0,8	400	4	2	*	*	11
2	0,8	400	8	3	*	*	17
2	0,8	400	16	4	62	*	23
2	1,4	25	4	2	0	3	0
2	1,4	25	4	2	0	4	0
2	1,4	25	4	2	0	2	0
2	1,4	50	4	2	4	9	1
2	1,4	50	4	2	1	8	1
2	1,4	50	8	3	11	20	3
2	1,4	100	4	2	16	38	2
2	1,4	100	8	3	*	46	4
2	1,4	100	16	4	*	234	6
2	1,4	200	4	2	*	*	6
2	1,4	200	8	3	*	*	9
2	1,4	200	16	4	162	515	11
2	1,4	400	4	2	*	*	12
2	1,4	400	8	3	*	*	17
2	1,4	400	16	4	*	*	24
2	2	25	4	2	0	2	0
2	2	25	4	2	0	2	0
2	2	25	4	2	1	3	0
2	2	50	4	2	5	10	1
2	2	50	4	2	5	8	1
2	2	50	8	3	31	23	2
2	2	100	4	2	60	35	2

2	2	100	8	3	*	43	4
2	2	100	16	4	*	133	6
2	2	200	4	2	*	*	6
2	2	200	8	3	*	*	9
2	2	200	16	4	*	547	10
2	2	400	4	2	*	*	12
2	2	400	8	3	*	*	17
2	2	400	16	4	*	*	23
3	0,8	25	4	2	0	7	1
3	0,8	25	4	2	0	8	1
3	0,8	25	4	2	2	9	1
3	0,8	50	4	2	2	31	1
3	0,8	50	4	2	3	27	2
3	0,8	50	8	3	5	49	2
3	0,8	100	4	2	5	80	3
3	0,8	100	8	3	7	98	5
3	0,8	100	16	4	22	93	6
3	0,8	200	4	2	15	*	6
3	0,8	200	8	3	8	*	9
3	0,8	200	16	4	17	*	12
3	0,8	400	4	2	64	*	9
3	0,8	400	8	3	525	*	19
3	0,8	400	16	4	30	*	25
3	1,4	25	4	2	1	11	1
3	1,4	25	4	2	0	10	1
3	1,4	25	4	2	0	8	0
3	1,4	50	4	2	1	28	1
3	1,4	50	4	2	1	30	1
3	1,4	50	8	3	82	82	3
3	1,4	100	4	2	40	79	3
3	1,4	100	8	3	*	111	4
3	1,4	100	16	4	*	*	6
3	1,4	200	4	2	221	*	7
3	1,4	200	8	3	526	*	9
3	1,4	200	16	4	453	*	13
3	1,4	400	4	2	231	*	14
3	1,4	400	8	3	*	*	19
3	1,4	400	16	4	45	*	26
3	2	25	4	2	0	6	0
3	2	25	4	2	0	5	0
3	2	25	4	2	0	5	0
3	2	50	4	2	4	27	1
3	2	50	4	2	4	25	1
3	2	50	8	3	12	51	2
3	2	100	4	2	71	110	2

3	2	100	8	3	*	130	5
3	2	100	16	4	197	424	7
3	2	200	4	2	*	*	7
3	2	200	8	3	*	*	10
3	2	200	16	4	*	*	13
3	2	400	4	2	*	*	14
3	2	400	8	3	*	*	18
3	2	400	16	4	*	*	25