Resumen

Este proyecto tiene por objetivo resolver un problema real de secuenciación de operaciones de una empresa del sector automoción, en un ambiente de taller mecánico con máquinas en paralelo, de forma que se cumpla con las fechas de entrega o, en su defecto, se minimice el retraso medio de los pedidos.

Como punto de partida se describe el problema, mostrando la situación inicial de la planta, explicando cómo se realiza la programación de la producción en la actualidad y destacando los inconvenientes que existen. En el análisis teórico posterior el problema se identifica como un caso de taller mecánico denominado *Flexible Job-Shop Scheduling Problem (FJSSP),* y se detalla un estado del arte de los métodos de resolución. Se comparan las diferentes alternativas disponibles hasta la fecha y se propone como procedimiento de resolución un método heurístico basado en la implementación informática de un algoritmo de *dispatching*.

A continuación se desarrolla el procedimiento de resolución y se muestra el funcionamiento de la aplicación creada. A partir de los datos del plan de entregas, el algoritmo propuesto genera como resultado la secuencia de operaciones necesaria para cumplir el plan y su temporización correspondiente, teniendo en cuenta las particularidades del caso real que se estudia y el objetivo que se pretende optimizar.

Por último se lleva a cabo un análisis de los resultados y los beneficios obtenidos, presentándose las conclusiones finales y posibles futuros desarrollos como continuación de este trabajo.

Indice

1. Prefacio

Un problema constante en las plantas industriales de producción es la búsqueda de la secuencia óptima de fabricación de los pedidos a través de las diferentes máquinas, con el fin de optimizar uno o varios objetivos. En casos como el del sector automoción, donde las penalizaciones por retrasos en la entrega de los pedidos son significativas, cobra especial importancia el cumplimiento de las fechas de entrega o, en su defecto, la minimización de los retrasos.

En las últimas décadas la secuenciación de operaciones ha sido uno de los campos más investigados, y lo continúa siendo en la actualidad, debido a la gran dificultad que supone obtener dicha secuencia y la importancia de conseguirla. La mayoría de los problemas de secuenciación que se presentan suelen ser muy complejos. En un ambiente de *Flexible Job -Shop* como el que se describe en este proyecto, nos encontramos con un problema de optimización combinatoria del tipo *NP-hard*, para el que no se conoce un algoritmo exacto que proporcione una solución óptima en un tiempo computacional razonable, salvo para ejemplares del problema de dimensiones muy reducidas.

Así, la imposibilidad práctica de encontrar una solución óptima mediante búsquedas exhaustivas ha favorecido, durante los últimos años, el desarrollo de abundantes aproximaciones y métodos heurísticos que propicien la obtención de soluciones no necesariamente óptimas, pero al menos sí factibles, con un índice de aceptabilidad elevado.

Por otro lado, las configuraciones reales de las plantas de fabricación son tan variadas y complejas que no permiten disponer de un modelo general de secuenciación que se pueda aplicar satisfactoriamente a cada caso concreto; de esta forma, la mayoría de las veces se deben realizar cuidadosas adaptaciones o bien diseñar un método específico para el problema en cuestión.

Actualmente se están realizando numerosos desarrollos e investigaciones en múltiples direcciones, con el fin de obtener técnicas de secuenciación cada vez más efectivas.

2. Introducción

En la planta de estampación y soldadura de componentes metálicos de una industria del sector automoción se plantea la necesidad de obtener un método para programar las operaciones productivas en las máquinas, de forma que la secuencia resultante minimice el retraso medio en la entrega de los pedidos. Los datos de partida reflejados en este trabajo son los de un caso real de ambiente de taller mecánico con máquinas en paralelo (*Flexible Job-Shop*), cuya configuración inicial es la siguiente:

- Llega el pedido del cliente que es un plan de entregas formado por *z* tipos de pieza. Cada una de estas piezas consta de una cantidad a producir y una fecha de entrega.

- La demanda es variable, siendo la demanda media de 28 tipos de piezas diferentes a fabricar.

- Cada pieza se compone de varias operaciones que han de realizarse siguiendo un orden de precedencia establecido.

- Hay 16 máquinas disponibles, de las cuales 10 son prensas y 6 son máquinas de soldadura.

- Cada máquina tiene asignado un tiempo de proceso y de preparación para cada operación de cada pieza.

- Los tiempos de preparación son constantes e independientes de la secuencia, es decir, su duración depende solamente de la operación actual a ser procesada. Se distinguen dos tipos de preparación diferentes: la correspondiente a las prensas y la de las máquinas de soldar. Las operaciones de prensas requieren un tiempo de preparación de 60 minutos y las de soldadura un tiempo de 20 minutos.

- Se ha de tener en cuenta que las máquinas son diferentes (de forma que algunas no están preparadas para realizar todos los tipos de piezas), y que puede existir una o varias unidades de cada tipo de máquina.

- Existen lotes de transferencia de piezas entre las máquinas de dos operaciones consecutivas. Esto implica que dado un pedido cualquiera de tamaño *q*, no es necesario esperar a que se procesen las *q* piezas de una operación para iniciar la operación siguiente, sino que una vez se ha fabricado una cierta cantidad de piezas, éstas se transfieren a la máquina siguiente para que trabaje en paralelo. El lote de transferencia para cualquier tipo de pieza siempre es de 15 unidades.

2.1. Objetivo

El objetivo de este proyecto es el siguiente:

- Resolver el problema de secuenciación que presenta una empresa real del sector automoción.

Para ello, se llevarán a cabo los siguientes puntos:

- Analizar los problemas de ambiente *Job-Shop*.
- Analizar los posibles métodos de resolución heurísticos y razonar la elección del método escogido.
- Describir los fundamentos del algoritmo de *dispatching*.
- Implementar el algoritmo de *dispatching* en una aplicación informática que proporcione una solución factible al problema de la secuenciación de los pedidos en las máquinas con el fin de minimizar el retraso medio en la entrega de los mismos, en un ambiente de *Flexible Job-Shop*.
- Comprobar la aptitud del algoritmo mediante la aplicación de una serie de pruebas y analizar los resultados obtenidos.
- Validar el algoritmo propuesto resolviendo casos reales y compararlo con las soluciones de la empresa.
- Plantear posibles futuros desarrollos.

2.2. Metodología

En primer lugar se ha definido detalladamente el problema a resolver. A continuación, se ha clasificado según su nivel de complejidad, puesto que en el ámbito de los problemas de secuenciación (y, en general, en los problemas de optimización combinatoria) dicha clasificación determina la metodología de resolución a utilizar. En este caso el problema planteado corresponde al tipo *NP-hard*, el más complejo, lo que aconseja descartar a priori los métodos exactos y seleccionar procedimientos aproximados, heurísticos, como posibles métodos de resolución.

A continuación se inicia una etapa de análisis preliminar en la que se estudian los problemas de ambiente *Job-Shop* y se describe un estado del arte en este campo,

indicando los distintos métodos de resolución posibles y exponiendo los trabajos más significativos recogidos en la literatura. Se realiza una valoración final y se explican los motivos de la elección del método de *dispatching* propuesto.

Una vez finalizado dicho análisis, en los apartados siguientes se explican los pasos necesarios para la implementación informática del procedimiento de *dispatching*. Como punto de partida se estudia el caso real de *Flexible Job-Shop* y se incluye una introducción al procedimiento de *dispatching*. A continuación se describe el funcionamiento del algoritmo de resolución. También se muestra un ejemplo práctico de aplicación del algoritmo para explicar su funcionamiento y se analiza el resultado obtenido.

Finalmente, se valida el procedimiento heurístico diseñado en diversos escenarios reales y se concluye mediante una valoración de los resultados obtenidos.

2.3. Alcance

El presente proyecto pretende llevar a cabo la implementación en lenguaje c# del algoritmo de *dispatching* desarrollado como procedimiento de resolución del problema de secuenciación de un caso real de *Flexible Job-Shop*. El alcance de este trabajo se extiende a cualquier proceso igual o muy semejante al descrito.

Se incluye el análisis de los diferentes métodos de resolución posibles y el razonamiento del método propuesto. A partir de los resultados obtenidos se realiza una valoración y se plantean posibles futuras líneas de investigación.

El programa muestra la solución propuesta en forma de tabla numérica; queda fuera del alcance de este proyecto el desarrollo de la representación gráfica mediante diagramas de Gantt. Además, el programa está diseñado para su utilización en PC como herramienta de ayuda a la toma de decisiones; no se contempla su implementación en las máquinas del taller.

3. Descripción del problema

3.1. La empresa

El problema planteado en este proyecto corresponde al caso de una industria real dedicada a la fabricación de componentes metálicos para el automóvil.

La empresa cuenta con un único centro productivo de estampación y soldadura, integrado por 23 personas. La cifra de negocios de 2008 ha sido de 5 Millones de euros.

Se trata de un tier-II o proveedor de segundo nivel, es decir, la planta suministra componentes a otra empresa que a su vez fabrica piezas para el cliente final.

La estructura organizacional consta de un departamento de Gerencia, situado en el mayor nivel jerárquico, del cual dependen los departamentos de Calidad, Producción y Administración, según el organigrama mostrado en al **Figura 3.1**:

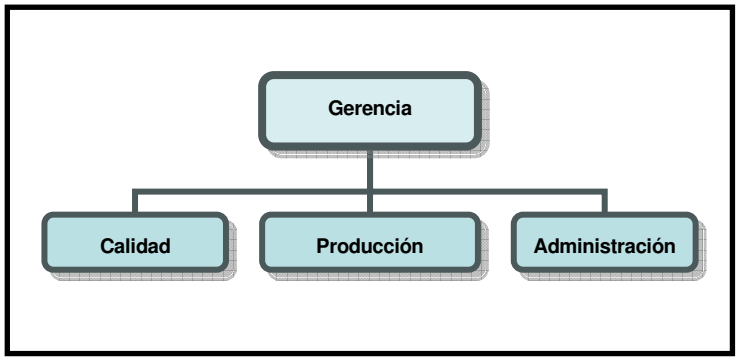


Fig. 3.1. Organigrama de la empresa

Las funciones relacionadas con RRHH, comercial y compras están integradas en Gerencia. Las funciones de logística son llevadas a cabo por el departamento de Producción.

La distribución en planta se muestra en la **Figura 3.2**. Se disponen de 16 máquinas, de las cuales 10 son prensas y el resto son equipos de soldadura. La planta está formada por dos secciones: una sección rectangular en la que se hallan las prensas más pequeñas y las máquinas de soldar, y una sección cuadrada donde se encuentran las prensas de mayor potencia. Existen 4 prensas en situación de parada permanente a causa de avería, las cuales no se han contabilizado como maquinaria disponible. No obstante, se ha previsto que la aplicación informática permita añadir estas máquinas una vez hayan sido reparadas.

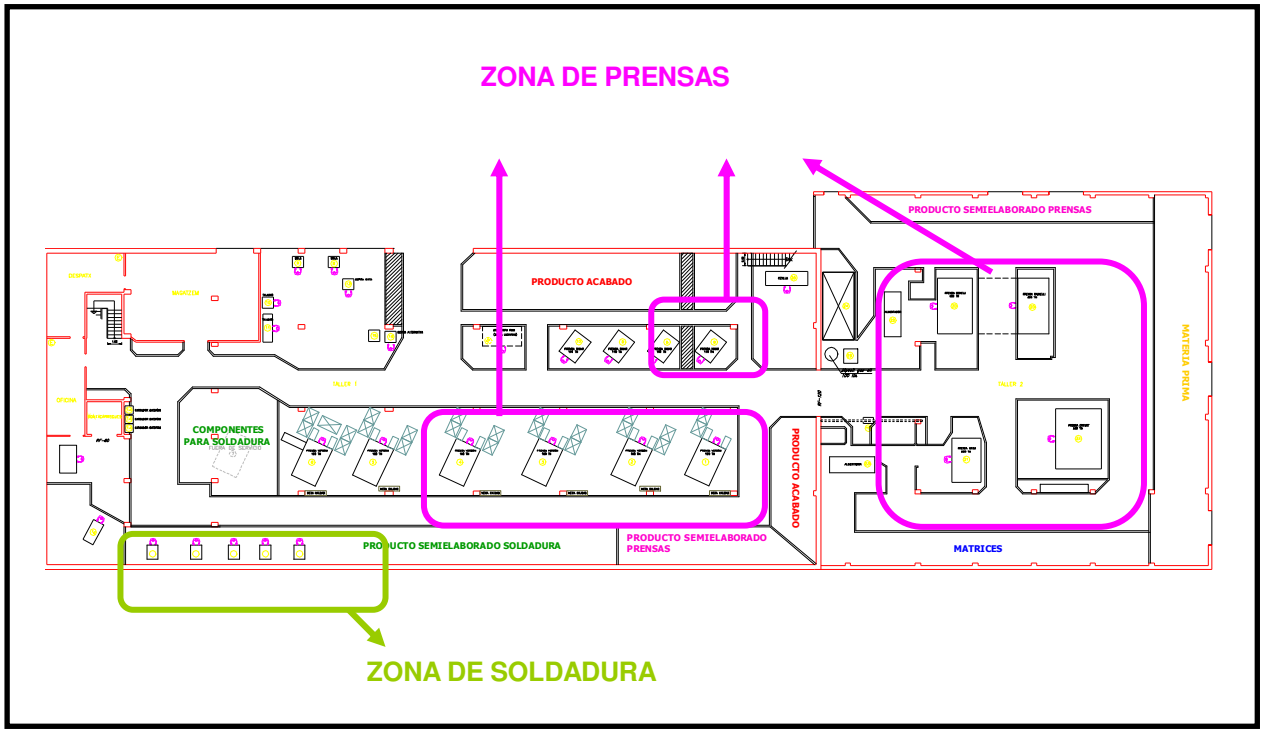


Fig. 3.2. Distribución en planta

Se dispone de espacios delimitados (mediante líneas pintadas en el suelo) para almacenar materia prima, matrices, productos semielaborados, componentes de soldadura y piezas acabadas. El resto corresponde a zonas de pasillo lo suficientemente amplias para permitir el paso de los toreros de una zona a otra. Además, existe una primera planta de idéntica sección que se utiliza como almacén de matrices, y un patio exterior parcialmente cubierto donde se almacena la materia prima y los residuos.

Semanalmente se fabrica una media de 28 tipos de piezas diferentes. El lunes de la semana actual la planta recibe el pedido del cliente. Las piezas se deben fabricar durante dicha semana y se entregan la semana siguiente en las fechas de entrega solicitadas. Se trabajan 16 horas diarias en dos turnos de producción, de lunes a sábado.

3.2. La programación de la producción

La programación de la produccion en la empresa se realiza de forma manual e intuitiva, no se dispone de un procedimiento implementado informáticamente como apoyo a la toma de decisiones.

El procedimiento utilizado en la planta para realizar la secuenciación de las operaciones y su asignación a las máquinas es el siguiente:

- Como paso previo, si existen retrasos de pedidos anteriores, las cantidades pendientes de entregar se suman al pedido actual.

- A continuación se selecciona el tipo de pieza a fabricar. Las piezas se clasifican de mayor a menor prioridad y se van fabricando según dicho orden. Una vez se selecciona una pieza se programan todas sus operaciones en las máquinas correspondientes, después se selecciona la siguiente pieza más prioritaria y se procede de igual modo. La selección de las piezas se realiza según el siguiente orden de prioridad:

    1) Mayor cantidad de unidades en retraso de pedidos anteriores.

    2) Fecha de entrega más urgente.

    3) Mayor duración de proceso.

    4) Mayor número de operaciones.

    5) Piezas con operación de soldadura.

    6) Mayor demanda mensual media.

- Una vez seleccionada una pieza, se programan todas sus operaciones. La asignación de las operaciones a las máquinas se realiza según el siguiente criterio:

    1) Máquina con disponibilidad más inmediata.

    2) Máquina con menor posibilidad de ser cuello de botella.

    3) Máquina más rápida capaz de realizar dicha operación.

La empresa no dispone de un *schedule* (programa) detallado con la programación prevista de las actividades productivas; la selección de operaciones y su asignación a las máquinas se improvisa en tiempo real, según las necesidades y la urgencia de la situación. Las dificultades que presenta la planta a la hora de realizar una programación adecuada provocan continuos retrasos en las entregas y la lógica insatisfacción del cliente.

3.3. La situación actual

La Gerencia de la empresa desea obtener un procedimiento de resolución práctico para resolver adecuadamente la programación de sus actividades productivas.

Según Gerencia, el departamento de Producción necesita una profunda revisión. En los dos últimos años la cartera de pedidos ha aumentado de forma importante, pero, sin embargo, no se ha realizado correctamente el análisis necesario para abordar esta nueva realidad y adaptar la compañía a las variaciones del entorno. Como resultado se ha llegado a una situación crítica, agravada especialmente por los continuos retrasos en las entregas, hasta el punto de comprometer la supervivencia de la empresa.

Entre los numerosos retos que se necesita resolver, y sobre los que se debe actuar, destacan los siguientes:

- ***Programación de la producción intuitiva, improvisada y manual.*** No se dispone de un procedimiento, implementado en un sistema informático, para realizar la programación de la producción. La asignación de las piezas a las máquinas se realiza de forma intuitiva, improvisada y manual, sin seguir ningún procedimiento formalizado. No existe una planificación semanal en la que se detalle las cantidades a producir de cada tipo de pieza y las máquinas donde deben realizarse; la programación se improvisa diariamente, según las urgencias y necesidades del momento, lo que comporta numerosos errores de previsión (al atender la urgencia se descuida lo importante). Como única fuente de información se dispone de una base de datos en Access en la que se registran los datos de producción (hojas de proceso, promedios de fabricación, listado de piezas, inventario de materias primas…etc.), pero no es fiable, puesto que la herramienta no se actualiza con la frecuencia necesaria, y por tanto, la información resulta parcial y obsoleta.

- ***Falta de control de inventario.*** El inventario de materia prima, productos y contenedores (los recipientes de carga para el almacenamiento y transporte de dichos materiales) no se actualiza con la frecuencia establecida (semanal). La falta de control y la dudosa fiabilidad de los datos de inventario hace que se produzcan situaciones como que se mantenga guardado en el almacén un stock de componentes obsoletos, o que se fabriquen más componentes (o menos) de los necesarios, con el consiguiente desperdicio de tiempo y recursos. Esta situación obliga a que cada vez que se consulta un dato se deba comprobar su veracidad.

- ***Falta de identificación de los productos.*** En ocasiones los contenedores no están etiquetados, por lo que se debe verificar su contenido, provocando pérdida de tiempo y de control.

- ***Distribución en planta no definida***. La única zona de la planta bien definida es la correspondiente a la zona de maquinaria. El espacio destinado al almacenamiento está delimitado pero no está organizado ni señalizado con carteles identificativos, por lo que se tiende al desorden y a la falta de control en la ubicación de los productos. Las deficiencias en este aspecto se manifiestan también en una frecuente falta de espacio y malgasto de tiempo y recursos: por ejemplo, se llega a invadir la zona de paso de carretilleros o a almacenar, al aire libre, chapa que debe ser protegida de las inclemencias del tiempo.

- ***Insatisfacción del cliente***. En la evaluación de proveedores realizada trimestralmente por su cliente, la empresa recibe repetidamente la calificación más baja, debido principalmente a los retrasos en las entregas. El problema es grave hasta el punto que el cliente se plantea la posibilidad de dejar de trabajar con la empresa.

3.4. El problema a resolver

De entre todos los retos descritos en el apartado anterior, el problema concreto a resolver consiste en programar adecuadamente los pedidos en las máquinas, de forma que la secuencia obtenida cumpla con las fechas de entrega o, en su defecto, se minimicen los retrasos.

A continuación se señalan los aspectos más importantes del problema. En el **Anexo A** se describen detalladamente los datos relativos a piezas, operaciones y máquinas.

3.4.1. Datos de los pedidos

El lunes de cada semana, la planta recibe de su cliente un pedido con las cantidades y las fechas de entrega de cada tipo de pieza. Se trata de un plan de entregas donde se especifica la demanda requerida de cada uno de los productos (piezas) y su fecha de entrega correspondiente. El horizonte de entrega es semanal. El lunes de la semana actual el cliente envía su pedido de la semana siguiente, de manera que la empresa fabrica durante la semana actual y entrega el pedido en la semana siguiente, tal como se muestra en la **Figura 3.3**.

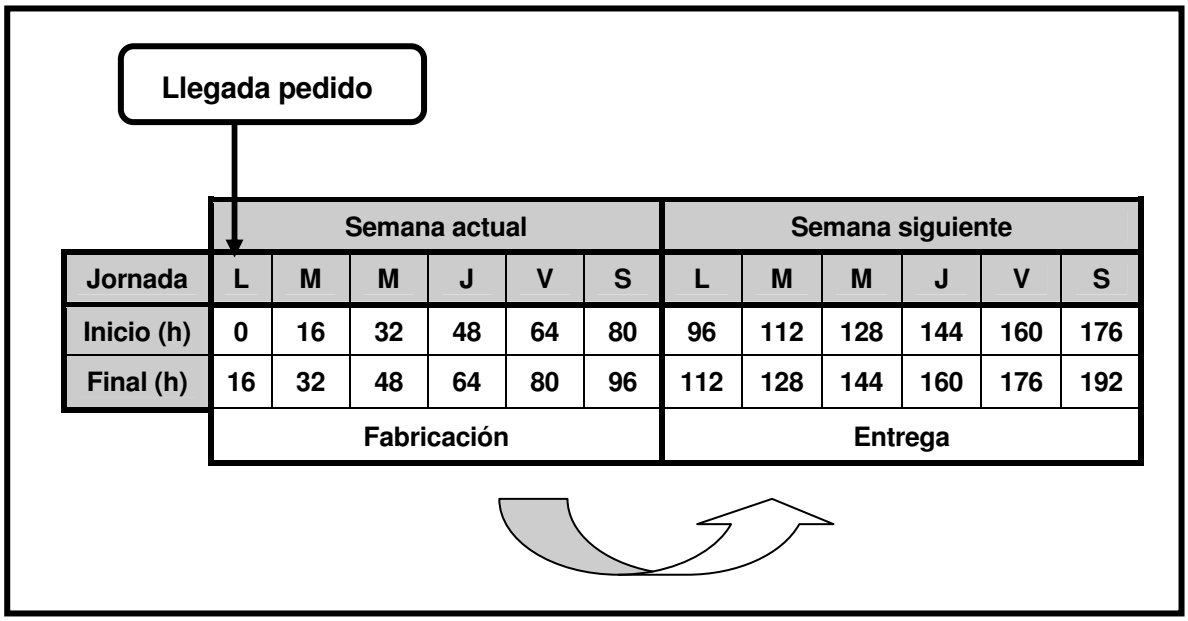


Fig. 3.3. Pedido del cliente

Si existen retrasos del pedido anterior, pueden darse dos situaciones, dependiendo de si la pieza en retraso se vuelve a pedir o no en el pedido siguiente:

a) ***Pieza con retraso en el pedido anterior con demanda en el pedido actual.***
En este caso las unidades en retraso se añaden a la nueva demanda y se entregan en el día solicitado del pedido actual.

b) ***Pieza con retraso en el pedido anterior sin demanda en el pedido actual.***
Si una pieza no se ha entregado a tiempo y en el pedido actual no hay nueva demanda, entonces se asigna como fecha de entrega el lunes de la semana actual.

En la **Tabla 3.4** se muestra un ejemplo de la demanda semanal.

Pieza	Demanda (Uds)	Fecha de entrega (día)
1	q_1	d_1
2	q_2	d_2
…	…	…
z	q_z	d_z

Tabla 3.4. Demanda semanal

En esta empresa se fabrica bajo pedido con una demanda variable. Algunas piezas suelen pedirse siempre. Otras, las correspondientes a recambios, tienen una demanda más espaciada. En cualquier caso, las piezas y las cantidades solicitadas pueden variar de forma significativa de una semana a otra. Se han considerado 28 tipos de piezas, que es la cantidad habitual solicitada por el cliente, pero el programa informático desarrollado permite añadir o eliminar piezas, según sea necesario.

En la **Tabla 3.5** se muestra la demanda media semanal de las piezas, ordenada de más a menor demanda media.

Numero pieza	Demanda media	Numero pieza	Demanda media
12	**25000**	27	**800**
7	**4500**	23	**750**
8	**3500**	2	**600**
6	**2500**	1	**500**
14	**2500**	3	**500**
19	**2300**	20	**500**
21	**2000**	4	**400**
5	**1800**	11	**400**
18	**1800**	24	**400**
15	**1500**	25	**350**
28	**1500**	22	**275**
16	**1000**	9	**200**
26	**1000**	10	**200**
17	**800**	13	**200**

Tabla 3.5. Demanda media semanal de las piezas

3.4.2. Datos de las máquinas

La planta de producción consta de 16 máquinas, entre las que se diferencian dos conjuntos básicos: prensa y soldadura. En el programa informático desarrollado se permite añadir más máquinas o eliminarlas.

Se dispone de flexibilidad parcial de máquinas, es decir, para algunas operaciones existe un subconjunto de máquinas idénticas capaz de procesarlas, pero ciertas operaciones sólo pueden realizarse en una sección de máquinas de la que existe una única unidad.

Por ejemplo, tal como se observa en la **Tabla 3.6**, existen 4 prensas idénticas del tipo S_5 (m_5, m_6, m_7 y m_8). Sin embargo, sólo hay una máquina del tipo S_3, la prensa m_3.

Tipo	Subconjunto	Uds	Máquina	Nombre/Código
Prensa	S_1	1	m_1	Arisa / A
	S_2	1	m_2	Benelli / B
	S_3	1	m_3	Radaelli / R
	S_4	1	m_4	Erfurt / E
	S_5	4	m_5	Verson / V1
			m_6	Verson / V2
			m_7	Verson / V3
			m_8	Verson / V4
	S_6	2	m_9	Inmar / I1
			m_{10}	Inmar / I2
Soldadura	S_7	2	m_{11}	Puntos / X1
			m_{12}	Puntos / X2
	S_8	2	m_{13}	Arco / Y1
			m_{14}	Arco / Y2
	S_9	2	m_{15}	Hilo / Z1
			m_{16}	Hilo / Z2

Tabla 3.6. Maquinaria disponible

3.4.3. Operaciones de las piezas

Cada pieza debe fabricarse siguiendo una serie de etapas llamadas operaciones o fases. Las piezas deben fabricarse respetando el orden de operaciones establecido.

En la **Tabla 3.7** se muestra las operaciones de la pieza número 1. El listado completo se recoge en el **Anexo A.1**.

Pieza	Operación	Descripción	Máquina	Golpes/hora
1	1	Cortar + Punzonar	A	1800
	2	Embutir	V1, V2,V3, V4	450

Tabla 3.7. Lista de operaciones

Según la tabla anterior, la secuencia para la pieza tipo 1 consta de dos operaciones. La primera operación solamente puede realizarse en la máquina cortadora, denominada Arisa (A), mientras que la segunda operación puede asignarse a cualquiera de las cuatro prensas Verson (V) en este caso, con el mismo tiempo de proceso, puesto que se trata de máquinas idénticas. También se indica los golpes por hora de las máquinas correspondientes a cada operación.

3.4.4. Tiempos de proceso unitarios

Se conoce el tiempo de proceso unitario p_{kij} de cada operación *k* de la pieza *i* en cada máquina *j* que puede procesarla. A modo de ejemplo se muestran los datos para la primera operación de las tres primeras piezas:

Operación k	Pieza i	Máquinas posibles j	Tiempo de proceso p_{kij}
1	1	m_1	p_{111}
1	2	m_1	p_{121}
1	3	m_5	p_{132}
		m_6	p_{133}
		m_7	p_{134}
		m_8	p_{135}

Tabla 3.8. Tiempos de proceso unitarios

Así, la primera operación de las piezas 1 y 2 sólo pueden hacerse en la máquina m_1. Sin embargo, la primera operación de la pieza 3 se puede realizar indistintamente en cualquiera de las máquinas m_5, m_6, m_7 y m_8. Nótese que el tiempo de proceso de una operación puede ser siempre el mismo, independientemente de la máquina que lo procesa, o bien puede ser distinto en cada máquina, dependiendo de cada caso concreto. El listado completo de los tiempos de proceso se muestra en el **Anexo A.2**.

3.4.5. Tiempos de preparación

Todas las operaciones conllevan tiempos de preparación puesto que todas ellas son diferentes entre sí y requieren matrices distintas.

Los tiempos de preparación se consideran todos constantes. Se distinguen dos tipos de preparación diferentes, la correspondiente a las prensas y la de los equipos de soldadura.

Prensas:

En las operaciones realizadas por prensas los tiempos de preparación, correspondientes al cambio de matriz, son todos *constantes* e iguales a 60 minutos: para cada operación, se debe invertir 30 minutos en quitar la matriz correspondiente a la operación anterior y otros 30 minutos en insertar la matriz de la operación actual. Por tanto, la disponibilidad de una máquina del tipo prensa no se tiene cuando se acaba de procesar una operación, sino 60 minutos después.

Soldadura:

En las operaciones de soldadura, los tiempos de preparación son todos constantes e iguales a 20 minutos. Análogamente, la disponibilidad de las máquinas de soldadura será efectiva 20 minutos después de procesar una operación.

Los tiempos de preparación se muestran en el **Anexo A.3**.

3.4.6. Lotes de transferencia de piezas

Para el avance en la fabricación de las piezas, se transfieren lotes entre máquinas que fabrican dos operaciones consecutivas. Esto se hace para permitir que la siguiente operación de un tipo de pieza pueda comenzarse a realizar en paralelo con la operación precedente.

En efecto, para que empiece una operación en un tipo de pieza no es necesario que haya terminado la anterior en todas las piezas de dicho tipo. Si, por ejemplo, se deben fabricar

500 unidades de un tipo de pieza, la operación 2 no tiene por qué esperar a que la operación 1 haya terminado en las 500 unidades, sino que basta con transferir un lote de una cierta cantidad de unidades a la segunda máquina para que empiece la operación 2. Esto sólo es posible si la duración de la primera operación es inferior o igual a la de la segunda. En caso contrario, para evitar que la segunda máquina se quede sin suministro, se debe retrasar el inicio de la segunda operación hasta el instante en que se pueda realizar la transferencia continua de la primera máquina a la siguiente.

El lote de transferencia entre máquinas utilizado en la empresa es de 15 unidades. Esto significa que una máquina podrá empezar a procesar una operación en cuanto se hayan completado 15 unidades de la operación anterior.

A efectos de la programación, el tamaño de lote sólo es determinante en la primera transferencia, que es la que determina el inicio de la fabricación de la siguiente operación en la segunda máquina. Por este motivo es adecuado un tamaño de lote pequeño, para adelantar lo antes posible el procesamiento de la operación siguiente.

Una vez se ha iniciado el tráfico de piezas entre máquinas, el tamaño del lote de transferencia pierde importancia, puesto que en general las máquinas siguientes son más lentas que sus predecesoras, por lo que es improbable que se queden paradas por falta de suministro, independientemente del tamaño de lote de transferencia. De todos modos, tal como se ha explicado anteriormente, también se ha previsto el caso contrario, retrasando el inicio de la segunda operación hasta que pueda realizarse el suministro continuo de la primera máquina a la siguiente.

Análogamente, se considera un tiempo medio de transferencia entre máquinas de 2 minutos. Se dispone de un torero que se dedica a transferir las cajas, las coloca al lado de la máquina y el operario va sacando las piezas de las cajas y las introduce (manualmente) en la máquina correspondiente. De esta manera, el tiempo mínimo para que la operación siguiente pueda empezar a fabricarse se refiere al tiempo que se tarda en procesar un lote de 15 piezas y desplazarlo de la primera a la siguiente máquina.

Existen tres casos posibles de transferencia entre máquinas, los cuales determinan el instante de disponibilidad de la siguiente operación, tal como se explica a continuación.

a) ***Transferencia en una misma máquina.***

Si en una máquina se está procesando una operación, y la siguiente operación programada debe asignarse a esa misma máquina, entonces, lógicamente, la segunda operación sólo podrá iniciarse cuando termine la primera.

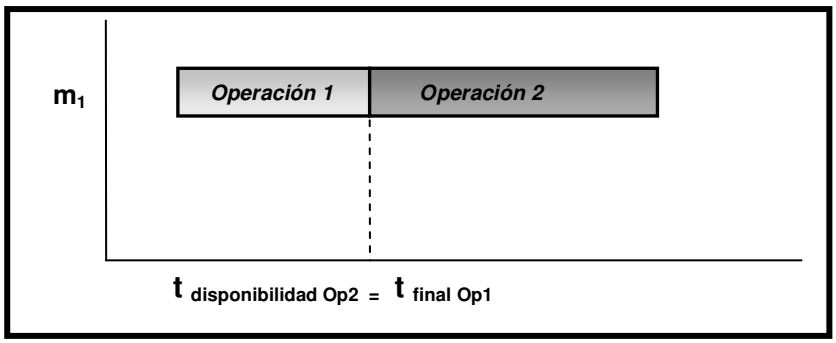


Fig. 3.9. Caso operaciones en la misma máquina

b) ***Transferencia en máquinas distintas (tiempo de proceso de la segunda operación mayor que el tiempo de proceso de la primera)***

En este caso la segunda máquina empieza a trabajar cuando se ha fabricado y transferido un lote de 15 piezas desde la primera máquina hacia la segunda.

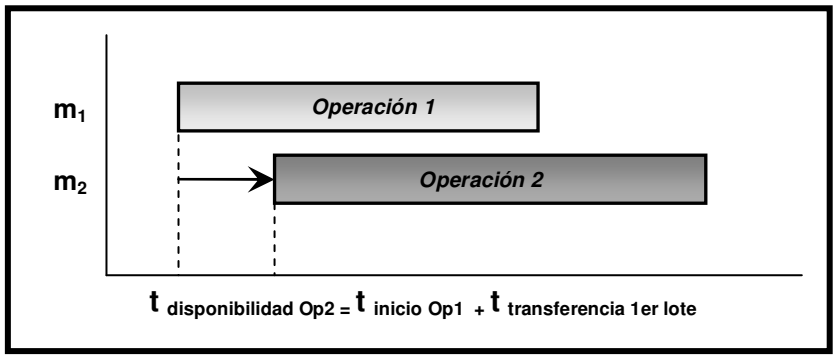


Fig. 3.10. Caso máquinas distintas, $t_{proceso\ Op2} \geq t_{procesoOp1}$

c) ***Transferencia en máquinas distintas (tiempo de proceso de la segunda operación menor que el tiempo de proceso de la primera)***

En este caso la segunda máquina, al ser más rápida, ha de esperar a que la primera vaya procesando (le da una cierta ventaja), para luego empezar a trabajar en paralelo con ésta. Para calcular el momento en el que debe ponerse en marcha la segunda

máquina, se toma como referencia el tiempo final de proceso de la primera operación, y a ese tiempo se le resta el tiempo de proceso de la segunda operación. A continuación, se suma el tiempo necesario para fabricar y trasladar un lote de 15 piezas entre las máquinas. El resultado es el tiempo de inicio de la segunda operación. De esta manera se evita que la segunda máquina, al ser más rápida que la primera, se quede parada por falta de suministro.

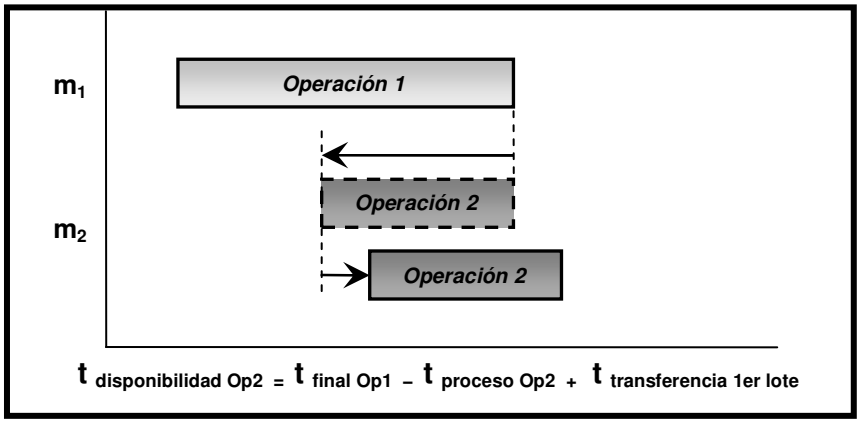


Fig. 3.11. Caso máquinas distintas, $t_{proceso\ Op2} < t_{proceso\ Op1}$

3.4.7. Particularidades del problema

Las consideraciones particulares a tener en cuenta sobre el proceso de fabricación son:

- ***Notación:*** Se entiende por operación o fase la tarea realizada por una máquina (ya sea una prensa o una máquina de soldar). Aunque una matriz permita agrupar varias operaciones (doblar, conformar, punzonar…), sigue considerándose como una sola operación, puesto que se realiza en el mismo golpe.

- ***Tipos de piezas:*** El número de tipos de piezas que se debe fabricar cada semana es aproximadamente de 28.

- ***Etapas del proceso:*** El proceso de fabricación generalmente consiste en una etapa de corte inicial seguida de una o varias etapas de estampación y una sola etapa final de soldadura. Algunas piezas no tienen etapa de soldadura.

- ***Máquinas polivalentes***: Las operaciones que se pueden fabricar en las prensas pequeñas (Verson, Inmar), también se pueden realizar en las grandes (Benelli, Radaelli

y Erfurt). Sin embargo, en la práctica no es factible puesto que las prensas grandes son cuellos de botella y según la Gerencia de la empresa resulta más conveniente asignar las tareas a las prensas más pequeñas, de las que se suele tener mayor disponibilidad, por lo que no se contempla la posibilidad de utilizar las prensas grandes para realizar este tipo de operaciones.

Existen operaciones (las de mayor espesor de chapa, sobretodo en embutición) que únicamente pueden realizarse en la prensa Erfurt.

- ***Flexibilidad parcial de máquinas:*** Se tienen varias unidades del mismo tipo de prensa (4 prensas Verson, 2 prensas Inmar). Del resto de prensas sólo se dispone 1 unidad de cada tipo. En soldadura se tienen 2 máquinas de soldadura por puntos, 2 de arco y 2 de hilo.

- ***Tiempos de proceso***: Se conocen los tiempos de proceso de cada operación de cada pieza en cada máquina, en piezas/hora y en segundos. Se pueden consultar en el **Anexo A.2**.

- ***Interrupción de secuencia / reprogramación***: En caso puntual de exigencia del cliente, por pedido urgente e inesperado, se puede interrumpir la secuencia de fabricación de las piezas y reprogramar las operaciones para dar paso a esta nueva prioridad. Este caso se resuelve utilizando la herramienta para reprogramar, añadiendo el nuevo pedido a la lista de operaciones y ejecutándolo de nuevo, de forma que se obtiene la nueva programación.

- ***Combinación de operaciones:*** Debido a su mayor potencia y a las mayores dimensiones de su plato, la prensa Erfurt es la única prensa disponible que permite combinar dos matrices en el mismo plato, por lo que es posible combinar dos operaciones consecutivas de una misma pieza. No se pueden combinar operaciones de piezas diferentes.

  Las operaciones combinables se combinan de la siguiente manera: una matriz se carga al lado de otra, con un tiempo de preparación de 60 minutos. En una combinación de dos operaciones, se necesitan dos golpes para obtener la primera pieza (en el primer golpe, la pieza está en el lado de la matriz 1, la matriz 2 trabaja en vacío. En el segundo golpe, se coloca una nueva pieza en la matriz 1, y la matriz 2 completa la segunda

operación de la pieza inicial). A partir de la segunda pieza producida, a cada golpe de prensa sale una pieza con las dos operaciones completadas.

En el caso real esta situación se da en ocasiones muy puntuales y, de acuerdo con la empresa, no se considerará en el procedimiento de resolución.

- ***Disponibilidad total de operarios***: No hay problemas de asignación de operarios a máquinas ni a realizar las transferencias de lotes entre ellas.

- ***Horario de trabajo:*** Se trabaja de lunes a sábado en dos turnos diarios, un primer turno de 6:00h a 14:00h y un segundo turno de 14:00h a 22:00h. Además, no se permite realizar horas extra.

- ***Averías***: En el proceso de fabricación se pueden producir diferentes tipos de averías, tanto en las máquinas de prensas como de soldadura. En el caso de las prensas, los cilindros se pueden quedar clavados por falta de lubricación o por excesiva fuerza del golpe en la bajada del plato. Si se produce una rotura, la reparación lleva más tiempo en función de la seriedad de la avería: leve (rotura de un pasador), moderada (se rompe una guía), grave (se parte parcialmente la matriz) o muy grave (se parte totalmente). De acuerdo con la empresa, los paros por avería no se contemplan en la programación desarrollada, aunque sí se han introducido reglas de robustez para disminuir el impacto de incidencias inesperadas en la reprogramación de las operaciones.

- ***Tiempos de mantenimiento***: Los tiempos de mantenimiento de las máquinas no influyen en la secuenciación, puesto que el mantenimiento se realiza fuera de los turnos de producción.

- ***Ventana horaria de entregas:*** Se establece que la entrega es de lunes a sábado al final de la jornada, es decir, hay una única entrega diaria a las 22h, con una cobertura de ±1h. Se permiten hacer varios viajes en caso de que no haya suficiente capacidad de camiones.

4. Secuenciación de operaciones para plantas tipo *Flexible Job-Shop*: estado del arte

En este capítulo se presenta un estado del arte de uno de los problemas más populares de la optimización combinatoria y la secuenciación de operaciones: el *Flexible Job-Shop Scheduling Problem* (FJSSP o problema del taller mecánico híbrido o flexible).

El capítulo se estructura de la siguiente forma: la sección 1 muestra, a modo de introducción, una visión general de la relevancia de la planificación de la producción en las plantas industriales y presenta el modelo del taller mecánico y en particular el FJSSP. A continuación, en la sección 2 se define de forma general el problema del taller mecánico, sus variables, restricciones y objetivos; además se indican las variantes más conocidas que existen. La sección 3 describe el estado del arte en el campo de la secuenciación, destacando los principales métodos de resolución utilizados desde sus orígenes, hace más de cuarenta años, entre los cuales figuran múltiples algoritmos y heurísticas del campo de la inteligencia artificial; de forma más concreta, se exponen los enfoques de solución más importantes que se han publicado para el caso del FJSSP. El análisis continúa en la sección 4 con la valoración de los métodos expuestos anteriormente y una tabla donde se evalúa la idoneidad de los mismos para la resolución del caso real que nos ocupa. Por último, en el apartado final se concluye con una exposición de las razones sobre la elección del procedimiento escogido.

4.1. Entorno de la secuenciación de operaciones

La planificación de la producción es un proceso de toma de decisiones habitual en el día a día de la industria. Se trata de uno de los procesos de decisión más importantes en gestión de la producción y aporta enormes ventajas. En efecto, si se realiza un análisis de las necesidades a largo plazo, y se conoce, a intervalos de tiempo regulares, la cantidad de producción que es demandada (y por lo tanto, a producir), entonces es posible desarrollar programas para la fabricación que permitan a la compañía obtener un pleno rendimiento, que no sea necesario mantener niveles de stock excesivos, que aumente la eficacia, disminuyan los costes de producción y en general, aumente el beneficio.

Para abordar el proceso de planificación se debe tener en cuenta muchos factores y considerar diversos objetivos, por lo que es necesario establecer distintos niveles de planificación según su nivel de detalle y plazo temporal para realizarlos. La secuenciación de operaciones pertenece al nivel de planificación a corto plazo, y es un problema de optimización combinatoria para el que se han desarrollado numerosos métodos a partir de

técnicas matemáticas y procedimientos heurísticos, con el objetivo de definir el orden o secuencia óptima en que se ejecutan las operaciones asignadas a los recursos productivos.

Las técnicas de resolución desarrolladas se pueden dividir en dos categorías: los ***métodos exactos***, que proporcionan una solución óptima, pero pueden requerir un tiempo de computación muy alto, y los ***métodos heurísticos***, que proporcionan una solución razonablemente buena en un tiempo aceptable. Además, se han desarrollado múltiples modelos teóricos para representar los diferentes tipos de configuraciones de planta.

Entre los modelos más habituales en la teoría de secuenciación destaca el problema de *Job-Shop Problem* (JSSP), considerado como una buena aproximación a la configuración real de las plantas industriales actuales. Como extensión de este modelo posteriormente se estableció el *Flexible Job-Shop Scheduling Problem* (FJSSP), que incorpora la posibilidad de considerar un conjunto de máquinas para realizar una operación determinada. Se trata de un modelo más complejo que el *Job-Shop*, puesto que añade el problema de escoger la asignación de cada una de las operaciones a una de las máquinas del centro de trabajo.

La complejidad de este tipo de problemas plantea dificultades en su resolución. En primer lugar, el gran número de soluciones (programas) que ofrece la combinatoria hace inviable una exploración exhaustiva de las mismas. Por otro lado, al ser *NP-hard* hay serias sospechas de que no existen métodos para resolverlos de forma óptima en tiempo polinómico. Finalmente, otra dificultad añadida es la gran variedad de tipos de problemas que se presentan, siendo necesaria, en muchas ocasiones, la creación de nuevos procedimientos adaptados a la particularidad de cada caso.

Estos hechos han motivado la investigación y desarrollo de métodos útiles para la secuenciación en el FJSSP. En el **apartado 4.3** se muestran las contribuciones más importantes que se han realizado en este campo desde sus orígenes hasta la actualidad.

4.2. El problema del taller mecánico *(Job-Shop Problem)*

Los problemas de secuenciación se basan en el denominado ***problema del taller mecánico*** (*Job-Shop Problem*). El enunciado básico general del problema es el siguiente (Companys, 2003, pág.171) **[6]**.

> ***n piezas*** *(lotes de piezas, pedidos u órdenes de trabajo) deben realizarse en* ***m máquinas*** *(recursos, secciones, puestos de trabajo). La realización de cada pieza implica la ejecución, en cierto orden establecido, de una serie de operaciones prefijadas donde cada operación está asignada a una de las m máquinas y tiene una duración (tiempo de proceso) determinada y conocida; debe establecerse un programa, es decir, la secuencia de operaciones en cada máquina y el intervalo temporal de ejecución de las operaciones, con el objetivo de optimizar un índice determinado que mide la eficiencia del programa.*

4.2.1. Problemas estáticos, semidinámicos y dinámicos

Los problemas de taller mecánico pueden ser de tres tipos:

- Problemas estáticos, en los que las piezas y las máquinas son un número finito y están todas disponibles en el mismo instante cero relativo inicial. En el instante de realizar la programación se conoce la ruta de la pieza, sus operaciones, en qué máquinas se debe realizar cada operación y la duración correspondiente. La finalidad de estos programas es optimizar un índice de eficiencia establecido.

- Los problemas semidinámicos tan sólo se diferencian respecto a los problemas estáticos en que los instantes de disponibilidad de las piezas y/o máquinas no son los mismos, aunque sí conocidos.

- Por último, en los problemas dinámicos el horizonte del funcionamiento del taller, así como el número de piezas, aunque no el de las máquinas, es ilimitado hacia el futuro. Aunque en cualquier instante el conocimiento de la situación es muy parecido al caso semidinámico, no basta con un solo programa, puesto que con el transcurrir el tiempo se producen nuevas llegadas de piezas y es necesario realizar un procedimiento de ciclos de programación, al cual deben asociarse los índices de eficiencia requeridos.

El problema a resolver, descrito en este proyecto, corresponde a un problema semidinámico, pues existen distintos instantes de disponibilidad iniciales de máquinas.

4.2.2. Tipos de flujo

Existen múltiples tipos de flujo de las piezas a través del taller, aunque generalmente suelen presentarse los siguientes casos:

- *Flow-Shop o Flujo Regular*: Todas las piezas siguen la misma ruta a través de las máquinas. Algunas piezas pueden no tener operaciones en alguna de las máquinas. Las piezas no tienen más de una operación por máquina.

- *Permutation Flow-Shop*: Es un caso particular del anterior en el que se establece la condición de que la secuencia de piezas en las máquinas ha de ser siempre la misma, por lo que el programa se obtiene mediante una permutación de las *n* piezas que optimice la medida de eficiencia. Una vez definido el orden de paso de las piezas en la primera máquina, este orden se conserva para las máquinas restantes.

- *Job-Shop o Flujo General*: En este caso se permite cualquier orden de piezas a través de las máquinas, pero cada pieza debe fabricarse según un orden establecido. Una misma pieza puede tener varias operaciones en la misma máquina.

- *Open Shop*: Cada operación debe realizarse en su máquina asignada, pero el orden de las operaciones de las piezas es indiferente.

En la empresa de estudio el tipo de flujo corresponde a un caso de *Job-Shop* con máquinas en paralelo, conocido como *Flexible Job-Shop*, debido a una mayor disponibilidad de las máquinas (una operación puede ser realizada por más de una máquina).

4.2.3. Medidas de eficiencia

Los programas se evalúan y se comparan entre sí utilizando medidas de eficiencia. En la práctica, el objetivo general es a menudo una combinación de varios objetivos básicos. A continuación se señalan algunos de los objetivos básicos más habituales:

- ***C_{max} (Makespan):*** Instante en el que la última pieza abandona el taller.

$$C_{\max} = \max\{C_i\}, \qquad \text{(Ec. 4.1)}$$

donde C_i es el tiempo de finalización de la pieza ***i***.

- ***L_{max} (Lateness):*** Valor máximo del huelgo de una pieza.

$$L_{\max} = \max\{L_i\}, \qquad \text{(Ec. 4.2)}$$

donde L_i es el huelgo de la pieza i, es decir, la diferencia entre C_i (tiempo de terminación de la pieza o completion time) y d_i (fecha de entrega o *due date*).

- ***T_{med} (Average Tardiness):*** Retraso medio de las piezas.

$$T_{med} = \frac{\sum_{i=1}^{n} \max(0, C_i - d_i)}{n}, \qquad \text{(Ec. 4.3)}$$

donde n es el número de piezas, C_i es el tiempo de finalización de la pieza i y d_i su fecha de entrega.

- ***T_{pond} (Weighted Tardiness):*** Retraso ponderado de las piezas.

$$T_{pond} = \sum_{i=1}^{n} \{ w_i \cdot \max(0, C_i - d_i) \}, \qquad \text{(Ec. 4.4)}$$

donde w_i es el peso de prioridad asociado a la pieza i.

4.2.4. Clasificación

Un problema de taller mecánico suele describirse mediante la sencilla notación propuesta por Conwall, Maxwell y Miller **[12]** formada por cuatro símbolos A / B / C / D:

- **A** indica el número de piezas. Si es un número arbitrario se designa por *n*; en nuestro caso *n*=28. En los problemas dinámicos puede referirse a la ley de probabilidad de llegada de las mismas.

- **B** indica el número de máquinas en el taller. Si es un número arbitrario se designa por *m*. En nuestro caso disponemos de 16 máquinas.

- **C** señala el tipo de flujo de las piezas en el taller. Generalmente se designa por uno de los símbolos siguientes:

    - F (*Flow-Shop o Flujo regular*)
    - P (*Permutation Flow-Shop*)

- R (*Randomly Routed Job-Shop o Rutas Aleatorias*)
- G (*General Job o Job-Shop*)

En nuestro problema el flujo corresponde al flujo de tipo general, *Job-Shop* o G. Se trata de una situación donde una pieza puede tener dos operaciones consecutivas en las máquinas j y j', mientras que otra pieza puede tener dos operaciones consecutivas en j' y j.

- **D** corresponde a la medida de eficiencia escogida para evaluar la calidad de los programas. En nuestro caso a resolver es T_{med}.

Según se desprende de lo expuesto hasta ahora se concluye que nuestro caso de estudio corresponde al problema semidinámico n / m / G / T_{med}, donde *n* = 28 y *m* = 16, flujo general y el retraso medio como medida de eficiciencia.

Cabe destacar que el procedimiento de resolución propuesto contemplará tanto el caso semidinámico como el estático (incorporando la posibilidad de fijar instantes de disponibilidad de máquinas dsitintos o iguales) y que los valores de *n* y *m* no son siempre fijos. Se han tomado 28 productos y 16 máquinas por corresponder a la situación más habitual en la planta, pero en la aplicación informática se tendrá en cuenta la posibilidad de agregar o eliminar máquinas y productos.

4.2.5. Hipótesis generalmente aceptadas en los problemas estáticos

Conway, Maxwell y Miller **[12]** enumeran las hipótesis habitualmente aceptadas en los problemas estáticos del Taller Mecánico:

1. Cada máquina está disponible continuamente desde un instante, f ≥ 0, hasta T, con T arbitrariamente grande. No existen intervalos de indisponibilidad bien sea por averías o por mantenimiento programado,

2. En las rutas de las piezas no se producen convergencias (montajes) ni divergencias (partición en lotes). Para cada operación existe una sola operación precedente inmediata (exceptuada la primera operación de cada pieza) y una sola operación siguiente inmediata (exceptuada la última operación de cada pieza),

3. Cada operación puede ejecutarse en un solo tipo de máquina de taller,

4. Sólo hay una máquina de cada tipo en el taller,

5. Cuando una operación ha comenzado debe terminarse antes de ejecutar otra en la misma máquina, no se admiten interrupciones,

6. No pueden solaparse dos operaciones de la misma pieza (en la misma máquina o en máquinas distintas),

7. Cada máquina puede ejecutar una sola operación a la vez,

8. La única restricción activa en el taller es la relativa a las máquinas, no existen restricciones relativas a la disponibilidad de mano de obra, utensilios, materiales, etc.

Estas hipótesis definen un problema teórico parecido al modelo considerado en este trabajo, a excepción de los puntos 1 (ya que las máquinas pueden no estar disponibles hasta un cierto instante), 3 (una operación puede realizarse en un solo tipo o en varios tipos de máquinas) y 4 (existen varias unidades de algunos tipos de máquina). Tal y como se explicó anteriormente, de acuerdo con la empresa se asumen las hipótesis 6 y 7 como ciertas.

4.2.6. El taller mecánico flexible *(Flexible Job-Shop)*

En este proyecto se plantea un problema de secuenciación en un entorno denominado taller mecánico con máquinas en paralelo *(Parallel Machine Scheduling)*, que también es conocido como taller mecánico híbrido o flexible *(Flexible Job-Shop Scheduling Problem, FJSSP).* Dicho problema es una variante del taller mecánico clásico y se caracteriza por poseer una mayor disponibilidad de las máquinas para realizar las operaciones.

En esta configuración, una máquina puede realizar más de un tipo de operación. Para cualquier operación dada, existe un subconjunto no vacío de máquinas capaz de realizarla. Se pueden considerar dos tipos de flexibilidad de máquinas para describir el funcionamiento del FJSSP:

- ***Flexibilidad total:*** todas las operaciones son realizables en todas las máquinas disponibles.

- ***Flexibilidad parcial:*** algunas operaciones solamente se pueden realizar en algunas de las máquinas disponibles.

En nuestro caso se tiene flexibilidad parcial de máquinas: programación de *n* piezas (productos) en *m* máquinas no homogéneas en paralelo. “No homogéneo” significa que no todas las operaciones pueden ser procesadas en cualquier máquina, sino sólo en un conjunto predeterminado.

4.3. Estado del arte

La investigación en el campo de secuenciación (*scheduling*) sigue en continuo desarrollo desde sus orígenes, a finales de los años cincuenta, y ha sido objeto de gran cantidad de literatura para su resolución, desde técnicas basadas en reglas de despacho *(dispatching)* hasta algoritmos de ramificación y poda *(branch&bound)* altamente sofisticados, heurísticas basadas en "cuellos de botella" *(bottleneck based heuristics)* o metaheurísticas como, por ejemplo, los algoritmos genéticos paralelos. Dichas técnicas han sido formuladas por un gran número de investigadores en el ámbito de la industria, desde científicos de gestión hasta expertos en producción. No obstante, con la aparición de nuevas metodologías, como redes neuronales y computación evolutiva, investigadores relacionados con la biología, genética y neurofisiología han realizado también contribuciones regulares a la teoría de secuenciación, poniendo de manifiesto la naturaleza multidisciplinar de la misma.

Existen diversas opiniones con respecto al origen del problema del taller mecánico. Roy y Sussman **[30]** en 1964 fueron los primeros en proponer la representación mediante el grafo disyuntivo. Balas en 1969 **[5]** aplicó por primera vez un procedimiento enumerativo basado en este grafo. Sin embargo existen trabajos anteriores: Giffer y Thompson **[16]** propusieron en 1960 un algoritmo de reglas de *dispatching* de prioridad, Jackson **[19]** en 1956 generalizó el algoritmo del *flow-shop* de Johnson **[20]** de 1954 al algoritmo del *job-shop* y, en 1955, Akers y Friedman **[3]** aplicaron un modelo de álgebra booleana para representar secuencias de procesamiento. Aunque no está claro a quién se le debe atribuir el primer planteamiento del *Job-Shop Problem*, se acepta que el libro "*Industrial Scheduling*" editado en 1963 por Muth y Thompson **[26]**, constituye la base para la mayoría de las investigaciones que siguieron.

Más recientemente, se han desarrollado métodos exactos para la resolución del *job-shop*, como los publicados por Applegate y Cook (1991) **[4]**, Brucker et al. (1994) **[8]**, Williamson et al. (1997) **[32]**, quienes han resuelto con éxito pequeños ejemplares, incluyendo el ejemplar 10x10 de Fisher y Thompson propuesto en 1963 y resuelto por primera vez veinte años después. En la actualidad, problemas de dimensión 15x15 siguen considerándose aún intratables mediante métodos exactos. Para resolver tales problemas se han desarrollado numerosos métodos heurísticos, entre los que cabe mencionar las contribuciones en algoritmos genéticos de Oliveira (2000) **[27]**, la creación de un método GRASP por Binato et al. (2002) **[7]**, el desarrollo de un método GRASP con recirculación de Aiex et al. (2003) **[1]**, y el método de optimización híbrida de Wang y Zheng (2001) **[31]**.

En el entorno de *job-shop* con máquinas en paralelo Cheng et al. (1990) **[11]** publicaron un estado del arte que describe las investigaciones realizadas en los problemas de

secuenciación con máquinas en paralelo. Posteriormente, Kim et al. (2002) **[22]** propusieron un método de resolución para dichos problemas basado en el algoritmo de recocido simulado. Más recientemente, Ravetti et al. (2007) **[29]** desarrollaron un modelo matemático basado en la metaheurística GRASP para máquinas en paralelo y tiempos de preparación dependientes de la secuencia. Finalmente, un informe reciente en esta misma área es el publicado por Allahverdi et al. (2008) **[2]**, en el que se menciona las diferentes clases de problemas de *job-shop* flexible y las técnicas empleadas para su resolución.

Los métodos desarrollados en el campo de la secuenciación de operaciones se pueden dividir en dos categorías, los métodos *exactos*, que proporcionan una solución globalmente óptima, pero pueden requerir un tiempo de computación muy alto, y los métodos aproximados (*heurísticos*), que habitualmente proporcionan una solución razonablemente buena en un tiempo aceptable. Como mejora a los métodos heurísticos se generaron los métodos *metaheurísticos*. A continuación en los **apartados 4.3.1 a 4.3.3** se muestra una selección, sin ánimo de ser exhaustiva, de los métodos más destacados.

En el **apartado 4.3.4** se explica el concepto de *robustez* y las reglas que pueden utilizarse para hacer un programa (solución) más robusto frente a las perturbaciones aleatorias que pueden ocurrir durante su ejecución.

4.3.1. Métodos exactos

Los métodos de resolución exactos son aquellos que garantizan el valor óptimo de la solución, pero normalmente a costa de un consumo computacional muy elevado, por lo que resultan adecuados para instancias pequeñas del problema. Según la clasificación establecida por Pinedo **[28]** se pueden distinguir los siguientes métodos:

a) Programación lineal

Este procedimiento, introducido en la Segunda Guerra mundial, resuelve un problema indeterminado, formulado a través de ecuaciones lineales, optimizando la función objetivo, también lineal, de tal forma que las variables de dicha función, estén sujetas a una serie de restricciones expresadas mediante un sistema de inecuaciones lineales. Un programa lineal entero es un programa lineal con la condición adicional de que las variables han de tomar valores enteros. Si entre las variables hay enteros y reales, se trata de un programa lineal mixto. Los programas lineales pueden resolverse en tiempo polinomial. Sin embargo, esto no sucede con los enteros ni los mixtos.

b) Ramificación y poda (*Branch&Bound*)

La técnica de *Branch&Bound* fue creada por Land y Doig **[24]** en 1960. Se suele interpretar como un árbol de soluciones, donde cada rama lleva a una posible solución posterior a la actual. La característica de esta técnica con respecto a otras anteriores es que el algoritmo se encarga de detectar en qué ramificación las soluciones dadas ya no son óptimas, para "podar" esa rama del árbol y no continuar desaprovechando recursos en casos que se alejan de la solución óptima. La búsqueda comienza en el nodo raíz y continúa hasta llegar a un nodo hoja. Desde un nodo no seleccionado, la operación de ramificación determina el siguiente conjunto de posibles nodos a partir del cual puede progresar la búsqueda. El procedimiento de poda selecciona la operación con la que continuar la búsqueda y se basa en una estimación de una cota inferior y la mejor cota superior alcanzada hasta el momento.

Habitualmente estos algoritmos toman como solución inicial una obtenida como resultado de algún método heurístico, lo que permite ahorrar un tiempo de computación considerable.

c) Programación dinámica

En 1953 Bellman **[6]** ideó un método recursivo al que denominó Programación Dinámica. Este método plantea la solución como una sucesión de decisiones que intenta, a partir de un enfoque divide y vencerás, reducir el tiempo computacional necesario. Se trata de resolver una serie de etapas (subdivisiones del problema) en cada una de las cuales se toma una decisión simple hasta hallar una solución al problema original. Se determina la solución óptima para cada iteración, en base a la información obtenida en todas las etapas anteriores, y su contribución a la función objetivo.

4.3.2. Métodos heurísticos

Los métodos heurísticos se utilizan para hallar una solución aunque no óptima, al menos factible, razonablemente buena, con un consumo de tiempo limitado. La *heurística* comprende los métodos o algoritmos de exploración en los que las soluciones se hallan mediante la evaluación del progreso logrado en la búsqueda de un resultado final. Se califica de heurístico a un procedimiento que encuentra soluciones con un coste computacional razonable, aunque no se garantice su optimalidad.

Según indica Martí **[25],** los algoritmos heurísticos suelen ser métodos de resolución muy útiles cuando se cumplen una o varias de las siguientes situaciones:

- La solución puede ser aproximada.
- El problema es de una naturaleza tal que no se conoce ningún método exacto para su resolución.
- Aunque existe un método exacto para resolver el problema, su uso es computacionalmente muy costoso o inviable.
- Como paso intermedio hacia otro procedimiento de resolución, normalmente un algoritmo metaheurístico.
- Cuando existen limitaciones de tiempo, espacio para almacenaje de datos, presupuesto, etc, que priorizan una solución rápida a costa de pérdida de precisión.

Al resolver un problema de forma heurística se debe de medir la calidad de los resultados puesto que, como ya se ha mencionado, la optimalidad no está garantizada. Se debe, pues, medir la calidad y eficiencia de un algoritmo para poder determinar su valía frente a otros. Para tal fin existen diversos procedimientos, entre los que se encuentran los siguientes **[25]**:

- Comparación con la solución óptima, si se conoce, o con la mejor solución disponible.
- Comparación con una cota.
- Comparación con un método exacto truncado.
- Comparación con los resultados de otras heurísticas.
- Análisis del peor caso.

A continuación se muestra una clasificación **[25],** no excluyente, de los métodos heurísticos más conocidos. Esta clasificación pretende establecer una enumeración de dichas técnicas heurísticas, aunque algunas se podrían incluir en más de una categoría:

a) Heurísticas constructivas

Consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración. Un ejemplo lo constituyen las heurísticas basadas en reglas de prioridad referidas en el **apartado f).**

b) Heurísticas de descomposición

Consisten en dividir el problema en subproblemas más pequeños, siendo la salida de uno la entrada de otro, de forma que al resolver todos los subproblemas se obtenga una solución para el problema global.

c) Heurísticas de reducción

Modifican las estructuras del modelo creando una versión más pequeña con el fin de hacerlo más sencillo de resolver, obteniendo, a partir de la solución del problema modificado, la solución del caso completo.

d) Heurísticas inductivas

Consiste en identificar alguna característica que deba poseer la solución óptima (por ejemplo, la detección de alguna variable con ciertos valores o correlación) e introducirla como restricciones del problema para reducir el espacio de soluciones, simplificando el problema. El riesgo obvio es dejar fuera la solución óptima del problema original.

e) Heurísticas de búsqueda local

A diferencia de los métodos anteriores, los procedimientos de búsqueda o mejora local, comienzan con una solución del problema y la mejoran progresivamente. El método finaliza cuando no existe ninguna solución accesible que mejore la anterior.

f) Heurísticas basadas en reglas de prioridad (*Priority Dispatch Rules*)

Las reglas de prioridad son probablemente las aplicadas con mayor frecuencia para resolver problemas de *scheduling*, esto se debe a su fácil implementación y a su aceptable complejidad de tiempo. Una regla de despacho es una regla que prioriza los trabajos en espera de ser procesados por una máquina. El criterio de prioridad debe tener en cuenta las características de las máquinas y de las operaciones, así como los instantes de tiempo. Cada vez que una máquina queda libre, la regla de despacho selecciona el trabajo con mayor prioridad entre los trabajos candidatos en espera. La investigación en este campo ha sido ampliamente desarrollada durante décadas, y numerosas reglas han sido diseñadas desde entonces. Los algoritmos de Giffer y Thompson **[16]** se pueden considerar como la base de la mayoría de heurísticas basadas en reglas de prioridad.

g) Heurísticas basadas en cuellos de botella (*Bottleneck Based Heuristics*)

Las serias dificultades que se han encontrado en la resolución óptima de ejemplares de tipo *Job-Shop* de dimensiones industriales propiciaron el diseño de muchos métodos heurísticos; entre los más exitosos destaca el *Shifting bottleneck procedure (SBP).*

Un SBP se caracteriza por las siguientes etapas: identificación del subproblema, selección del cuello de botella, solución del subproblema y reoptimización del *scheduling*. La estrategia implica relajar el problema de *scheduling n* piezas x *m* máquinas, en *m* problemas de una máquina y resolver cada subproblema de forma iterativa usando la aproximación de Carlier **[9].** Cada una de las soluciones se compara con las demás y se ordenan las máquinas según su solución. La máquina no secuenciada que tiene la solución mayor se identifica como la máquina que provoca el cuello de botella. SPB secuencia dicha máquina basándose en las que ya han sido calculadas, ignorando el resto de máquinas no secuenciadas. Se secuencia la máquina cuello de botella puesto que el *scheduling* en etapas posteriores podría deteriorar el tiempo de realización del programa (*makespan*). Cada vez que una máquina se identifica como cuello de botella, todas las máquinas que ya han sido secuenciadas, y que son susceptibles de mejoras, se reoptimizan localmente resolviendo el problema de una máquina de nuevo. La principal contribución de esta aproximación es la forma en que se usa la relajación de una máquina para decidir el orden en el que las máquinas deben ser secuenciadas.

En los últimos años han aparecido una serie de métodos bajo el nombre de *Metaheurísticas* con el propósito de obtener mejores resultados que los obtenidos por los métodos heurísticos tradicionales. La base de los procedimientos metaheurísticos la constituyen los métodos constructivos y los de búsqueda local.

4.3.3. Métodos metaheurísticos

Los algoritmos metaheurísticos son algoritmos aproximados de optimización y búsqueda de propósito general. Parten de una solución factible inicial (habitualmente obtenida con otra heurística) y mediante alteraciones de la solución, van iterando a otras factibles de su entorno, almacenado la mejor solución encontrada hasta que se cumpla un determinado criterio de parada.

El sufijo "meta" significa "más allá", a un nivel superior, las metaheurísticas son estrategias para diseñar o mejorar los procedimientos heurísticos con miras a obtener un alto rendimiento. Se aplican generalmente a los problemas para los cuales no hay algoritmo específico satisfactorio o heurístico para su resolución; o cuando no es práctico utilizar tal método en ejecución.

Los métodos metaheurísticos poseen características interrelacionadas que dificultan elestablecimiento de una única clasificación. A continuación se describen algunas de las formas más habituales de clasificación recogidas en la literatura **[14]:**

- ***Atendiendo a la inspiración:***
  Natural: Algoritmos que se basan en un símil real, ya sea biológico, social, etc.
  Sin inspiración: Algoritmos que se obtienen directamente de sus propiedades matemáticas.

- ***Atendiendo al número de soluciones:***
  Poblacionales: buscan el óptimo a través de un conjunto de soluciones.
  Trayectoriales: trabajan con una sola solución que mejoran iterativamente.

- ***Atendiendo a la función objetivo:***
  Estáticos: no hacen ninguna modificación sobre la función objetivo del problema.
  Dinámicos: modifican la función objetivo durante la búsqueda.

- ***Atendiendo a la vecindad:***
  Una vecindad: durante la búsqueda utilizan una sola estructura de vecindad.
  Varias vecindades: durante la búsqueda modifican la estructura de la vecindad.

- ***Atendiendo al uso de memoria:***
  Sin memoria: se basan exclusivamente en el estado anterior.
  Con memoria: utilizan la estructura de memoria para recordar la historia pasada.

Según el criterio de clasificación basado en el número de soluciones, cabe destacar los siguientes métodos metaheurísticos:

a) Metaheurísticas basadas en trayectorias

Parten de una solución inicial vacía y van actualizando la solución actual mediante la exploración del vecindario, formando un vecindario. La búsqueda finaliza cuando se alcanza un número máximo de iteraciones, se encuentra una solución con una calidad aceptable o se detecta un estancamiento del proceso.

Un ejemplo de metaheurística constructiva es la ***GRASP (Greedy Randomized Adaptative Search Procedures)***, un procedimiento de búsqueda introducido en 1995 por Feo y Resende **[15],** basado en funciones voraces aleatorizadas que se adaptan. Es uno de los métodos más populares debido a su sencillez y facilidad de implementación.

El método GRASP es un método iterativo donde cada paso consiste en una fase de construcción y otra de mejora. En la fase de construcción se aplica una heurística constructiva para obtener una solución inicial y en la segunda fase dicha solución se mejora mediante un algoritmo de búsqueda local. En cada iteración la elección del próximo elemento a ser añadido a la solución parcial está determinada por una función greedy (voraz o golosa), la cual elige el elemento que da mejor resultado inmediato sin tener en cuenta una perspectiva más amplia. Se dice que se adapta porque en cada iteración se actualizan los beneficios obtenidos al añadir el elemento seleccionado a la solución parcial. Se denomina aleatorizado porque no selecciona al mejor candidato sino que, con el objeto de diversificar y no repetir soluciones, se construye una lista de los mejores candidatos, de entre los cuales, en función de su bondad, se elige uno al azar. En la fase de mejora se realiza un proceso de búsqueda local a partir de la solución construida hasta que no se pueda mejorar más.

Los algoritmos de **Recocido Simulado *(Simulated Annealing)*** fueron introducidos por Cerny **[10]** y Kirkpatrick **[23]** para la optimización de problemas combinatorios con mínimos locales. Utilizan técnicas de optimización no determinista: no buscan la mejor solución en el entorno de la solución actual sino que generan aleatoriamente una solución cercana y la aceptan como la mejor si tiene menor coste, o en caso contrario con una cierta probabilidad p; esta probabilidad de aceptación irá disminuyendo con el número de iteraciones y está relacionada con el empeoramiento del coste.

El término de recocido simulado proviene de la estrecha analogía que guarda con el proceso del recocido tal y como se usa en metalurgia. Éste consiste en el enfriamiento progresivo de un metal de manera que sus moléculas van adoptando poco a poco una

configuración de mínima energía. A medida que la temperatura disminuye se va ralentizando el movimiento de las moléculas y éstas tienden a adoptar paulatinamente las configuraciones de menor energía.

El espacio de configuraciones (posiciones de las moléculas) en el caso del *Job-Shop* viene determinado por los valores de una variable de interés, normalmente la secuencia de operaciones que se desean procesar, mientras que el papel de la energía lo asume la función que se intenta minimizar.

La **Búsqueda Tabú** ***(Tabu Search)*** aumenta el rendimiento del método de búsqueda local mediante el uso de estructuras de memoria: una vez que una solución es abandonada, se la marca como "tabú" de modo que el algoritmo no vuelva a visitar esa posible solución en un número determinado de iteraciones. Este método se debe a Glover **[17]**, quien lo presentó en 1986.

Consiste en un procedimiento que restringe la búsqueda y evita los mínimos locales, almacenando la historia de búsqueda en memoria. Se prohíben movimientos entre vecinos que cumplan ciertas propiedades, con objeto de guiar el proceso de búsqueda para no duplicar soluciones previamente obtenidas.

b) Metaheurísticas basadas en poblaciones

Son métodos que van construyendo un conjunto de individuos que representan el conjunto de soluciones. El procedimiento consiste en generar, seleccionar, combinar y reemplazar dicho conjunto. Su eficiencia y resultado depende fundamentalmente de la forma con la que se manipula la población en cada iteración.

Un ejemplo lo constituyen los Algoritmos Genéticos.

Los ***Algoritmos Genéticos*** surgieron en 1975 de la mano de Holland **[18]**, e imitan el procedimiento de la selección natural sobre el espacio de soluciones del problema considerado. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Otra metaheurística basada en poblaciones es la **Optimización basada en Colonias de Hormigas** ***(Ant Colony Optimization)***. Se trata de algoritmos inspirados en el comportamiento que muestran las hormigas para encontrar las trayectorias desde la

colonia hasta el alimento. En la naturaleza, las hormigas vagan aleatoriamente en su búsqueda de alimento, y a lo largo de su trayectoria de regreso a la colonia depositan una hormona denominada feromona. Si otras hormigas encuentran este rastro, lo más probable es que sigan esta trayectoria para dejar el alimento en la colonia. Con el tiempo el rastro de la feromona comienza a evaporarse y se reduce su fuerza atractiva, pero las hormigas que siguen el rastro la refuerzan y logran que perdure por más tiempo. Cuantas más hormigas recorren una trayectoria, más intenso es el olor del rastro, lo que estimula a más hormigas a seguir esa trayectoria. Desde el punto de vista algorítmico, la evaporación de la feromona tiene la ventaja de evitar la convergencia a una solución localmente óptima. Si no hubiera feromonas, todas las trayectorias serían igualmente atractivas para las hormigas, lo que haría que la exploración del espacio de la solución fuese demasiado amplia. Este comportamiento es la base para el diseño del algoritmo, donde las "hormigas simuladas" caminan alrededor del gráfico que representa el problema a solucionar.

En 1995 Kennedy y Eberhart **[21]** desarrollaron una técnica de optimización basada en poblaciones llamada ***Particle Swarm Optimization, PSO*** **(Optimización mediante cúmulo de partículas).** Este tipo de algoritmos se inspiraron en el comportamiento social de las bandadas de pájaros y los bancos de peces. En estas asociaciones existe un líder que condiciona el desplazamiento de la manada. Además, cada individuo se guía basándose en su propia experiencia previa; en concreto, la toma de decisión por parte de cada individuo o partícula se realiza teniendo en cuenta un componente social y un componente individual, mediante las que se determina el movimiento de esta partícula para alcanzar una nueva posición.

4.3.4. Robustez

Una vez presentados los procedimientos de resolución heurísticos y metaheurísticos conviene destacar aquí la importancia del concepto de *robustez*, ya que dichos procedimientos no sólo deben proporcionar una solución factible, sino también robusta frente a las variaciones externas que puedan afectar a su rendimiento.

El concepto de robustez de una solución o programa se refiere generalmente a su habilidad para afrontar perturbaciones aleatorias que ocurran en el tiempo de ejecución y permanecer aceptable (con un deterioro mínimo en su función objetivo). En la práctica, debido a las condiciones de trabajo dinámicas e inciertas, los planes difícilmente se corresponden con las previsiones y deben modificarse a menudo a causa de sucesos aleatorios (por ejemplo, una avería en una máquina o un trabajo urgente que debe insertarse inmediatamente). De este modo, cuanto más robusto es un plan o una

secuenciación de operaciones, más sencillo es reprogramarlo, por lo que resulta conveniente incorporar reglas de robustez en el propio desarrollo del método de resolución.

En concreto, se pueden seguir varias reglas para crear programas más robustos, por ejemplo:

- ***Insertar tiempos de máquina parada.***

  La primera regla sugiere insertar periodos de paro de máquina en ciertos instantes de tiempo. Es equivalente a programar las máquinas por debajo de su capacidad. Las duraciones de los periodos ociosos, así como la fijación de los tiempos en el programa, dependerán de la naturaleza de las perturbaciones que pueden ocurrir. Se podría argumentar que los periodos ociosos al principio del programa deben ser más cortos que los periodos ociosos hacia el final del programa, puesto que la probabilidad de que ocurra un suceso inesperado al principio suele ser menor que más adelante.

  En la práctica, algunos programadores siguen la regla según la cual en la semana actual las máquinas funcionan al 90% de su capacidad, en la semana siguiente al 80%, y en la siguiente al 70%. Sin embargo, una razón para mantener los periodos ociosos de igual longitud al principio que al final del programa es la siguiente: aún cuando la probabilidad de una perturbación sea pequeña, su impacto relativo es más grave que una perturbación que ocurra hacia el final del proceso.

- ***Programar los trabajos menos flexibles primero.***

  Esta regla indica que los trabajos menos flexibles deben tener mayor prioridad que los más flexibles. Si ocurre un suceso inesperado, entonces los trabajos pendientes de procesar corresponderán a los más flexibles. La flexibilidad de un trabajo está determinada, por ejemplo, por el número de máquinas que pueden realizarlo, o bien por los tiempos de preparación que requiere. Algunos trabajos requieren una preparación que no depende de la secuencia, mientras otros precisan una preparación que dependerá fuertemente de la secuencia, es decir, los tiempos de preparación serán cortos sólo cuando siguen a ciertos trabajos, de lo contrario pueden ser mucho más largos. Dichos trabajos son claramente menos flexibles.

- ***No posponer innecesariamente el procesamiento de ninguna operación.***

  Desde el punto de vista de costes de inventario y penalizaciones por adelantos, es deseable empezar las operaciones tan tarde como sea posible. Sin embargo, desde un punto de vista de robustez, puede ser deseable empezar las operaciones tan

pronto como sea posible. Así, en cada caso se debe buscar el equilibrio entre robustez y costes por inventario, y ponderar qué es lo que más interesa.

- ***Mantener siempre algún trabajo en la cola de las máquinas más utilizadas.***

  Esta regla tiene el propósito de asegurar que una máquina que sea cuello de botella nunca deje de estar alimentada a causa de un suceso aleatorio que ocurra en alguna etapa anterior. El razonamiento es el siguiente: si no se mantiene un inventario frente a la máquina cuello de botella y la máquina que alimenta el cuello de botella de repente se avería, entonces el cuello de botella debe permanecer ocioso y puede no ser capaz de recuperar el tiempo perdido más adelante.

En este apartado se ha analizado la influencia de la aparición de sucesos aleatorios inesperados en la programación de operaciones, y se ha destacado la importancia de las reglas de robustez como una poderosa ayuda a la optimización bajo incertidumbre. Como se explicará en el capítulo siguiente, se han adoptado dichas reglas introduciéndolas en el propio procedimiento de resolución.

4.4. Análisis comparativo de los métodos de resolución

A continuación se muestran los criterios de selección que han motivado la elección del método de resolución para el caso de *Job-Shop* flexible descrito en este trabajo. Se realiza una comparación entre los métodos disponibles referidos en los apartados anteriores, valorando los pros y contras de cada uno de ellos y analizando su adecuación al problema a resolver.

La tabla siguiente resume las principales ventajas e inconvenientes de los tipos de métodos aplicables, y sirve como base para establecer un criterio de selección:

Métodos	Ventajas	Inconvenientes	Idoneidad
Exactos	Solución óptima Exploración exhaustiva de las soluciones Precisos	Tiempo computacional elevado No adecuados para problemas *NP-hard* Limitados a *Job-Shop* reducidos Dificultad de implementación	*Descartado*
Heurísticos	Tiempo computacional razonable Adecuados en problemas *NP-hard* Proporcionan varias soluciones Sencillez de implementación	No siempre garantizan el óptimo Exploración de soluciones no exhaustiva	*Adecuado*
Meta heurísticos	Tiempo computacional razonable Adecuados para problemas *NP-hard* Proporcionan varias soluciones Sencillez de implementación Mejoran los resultados heurísticos	No siempre garantizan el óptimo Exploración de soluciones no exhaustiva	*Adecuado*

Fig. 4.1. Cuadro comparativo de los posibles métodos de resolución

Como criterio de selección se consideran los siguientes aspectos fundamentales que debe cumplir el método de resolución:

- ***Simple.*** El método debe estar basado en un principio sencillo y claro; fácil de comprender.

- ***Rápido.*** Se ha de proporcionar una respuesta rápida, a ser posible, unos pocos minutos.

- ***Solución aproximada.*** No es imprescindible obtener la secuencia óptima, pero sí una secuencia factible que mejore el retraso medio respecto a la solución actual.

- ***Realista.*** El programa debe respetar las particularidades del problema (prioridades, precedencias, tiempos de preparación, etc).

- ***Robusto.*** El comportamiento del método debe ser poco sensible a pequeñas alteraciones del contexto de aplicación.

- ***Interactivo.*** Se debe permitir que el usuario pueda aplicar sus conocimientos para mejorar el rendimiento del procedimiento.

- ***Múltiple***. Debe suministrar diferentes soluciones alternativas de alta calidad entre las que el usuario pueda elegir.

4.5. Elección del método de resolución

Como resultado de los criterios anteriores se descartan directamente los métodos exactos de resolución, puesto que demuestran ser intratables para ejemplares de las dimensiones del problema de estudio, consumen un tiempo computacional muy elevado y son difíciles de implementar.

Se opta por utilizar una heurística. La elección de este de método se basa en varios motivos. En primer lugar, la solución puede ser aproximada, siempre que satisfaga de forma razonable el objetivo perseguido. En segundo lugar, aunque siempre que esté disponible y se pueda costear es preferible un método exacto frente a uno heurístico, en nuestro caso no se puede aplicar un método exacto dada la naturaleza *hard* del problema, puesto que dicho método o no existe o dispara el consumo de tiempo computacional. La empresa dispone de serias limitaciones económicas y de tiempo, por lo que prevalece la obtención de un método rápido que mejore la planificación actual, aunque sea menos preciso. Por último, una ventaja importante es que su complejidad es reducida en comparación con los métodos exactos, por lo que suelen ser más fáciles de entender (por parte de los directivos de las empresas y gente no experta), sin mencionar que generalmente ofrecen más de una solución, permitiendo así ampliar las posibilidades de elección.

En este punto se plantea la cuestión ¿Qué tipo de heurística aplicar? La literatura recoge que uno de los métodos más exitosos utilizados para este tipo de problema corresponde a la utilización de un método híbrido, consistente en aplicar primero una heurística (por ejemplo un algoritmo de tipo constructivo) que proporcione una solución inicial, y a continuación, en una segunda etapa, utilizar una metaheurística de búsqueda local que intente mejorar el resultado obtenido.

Así, como heurística inicial se ha elegido un método constructivo directo basado en un ***Algoritmo de Dispatching***, cuyo funcionamiento se explica en el **capítulo 5.3**.

La utilización de un procedimiento de búsqueda local en el caso de estudio presenta dificultades debido a la complejidad de la definición del vecindario. El algoritmo de *dispatching* que proporciona la solución inicial está fuertemente priorizado y existen numerosas restricciones de precedencia, por lo que cabe esperar que las posibilidades de alterar operaciones o máquinas para generar un programa vecino son reducidas. Además, experimentalmente se ha observado que estos tipos de vecindario para el problema de *Job-Shop* son demasiado simples para ser efectivos: el número de programas vecinos que son mejores que el programa actual tiende a ser muy limitado.

Por esta razón, se ha descartado la búsqueda local y, como alternativa, se ha optado por desarrollar un procedimiento aleatorio en la segunda etapa del procedimiento de resolución. Es decir, en una primera etapa se obtiene una única solución inicial resultante de la aplicación del procedimiento directo del algoritmo de *dispatching*, en el que las reglas de prioridad siguen un orden fijo predeterminado. En una segunda etapa, con el fin de obtener un mayor número de soluciones y tratar de encontrar entre ellas alguna que mejore la solución inicial, se tiene la opción de utilizar el procedimiento aleatorio, consistente en relajar los órdenes de las reglas prioridad mediante la asignación de porcentajes de probabilidad de aplicación de las mismas, obteniendo tantas soluciones como número de repeticiones haya indicado el usuario.

La ponderación probabilística de las reglas de prioridad está motivada por el siguiente razonamiento: El enfoque de solución propuesto, basado en un método de *dispatching,* es un procedimiento directo, es decir, las decisiones se van tomando según un orden de prioridad de selección de operaciones fijo que conduce a una única secuencia solución. Sin embargo, a pesar de que este orden se ha intentado establecer para conseguir un buen valor del retraso medio, lo cierto es que no es un orden absoluto y tampoco tiene por qué ser siempre el mejor. En este contexto, se plantea la necesidad de relajar el orden de las reglas de prioridad mediante la asignación de pesos de probabilidad, para observar si entre las distintas combinaciones aleatorias se produce alguna mejora respecto a la solución inicial.

El funcionamiento detallado de los procedimientos directo y aleatorio se explica, respectivamente, en los **apartados 5.3.6 y 5.3.7**.

Finalmente, con el fin de tratar de obtener una solución resistente frente a los incidentes aleatorios, se han utilizado distintas reglas de robustez en el diseño del algoritmo, tal como se indica en el **apartado 5.3.5**.

5. Procedimiento de resolución

5.1. Clasificación del problema

Como se ha comentado, el problema de secuenciación planteado en este proyecto corresponde al tipo de problemas *NP-hard*, los de mayor complejidad, para los que no existen algoritmos capaces de conducir a soluciones óptimas en un tiempo computacional aceptable. Para poder solucionarlo se ha desarrollado un procedimiento heurístico que se presenta en este apartado.

5.2. Formalización del problema

El caso expuesto corresponde al problema de taller mecánico flexible n/m/G/T_{med} con máquinas no homogéneas en paralelo *(Flexible Job-Shop Scheduling Problem,* FJSSP*).* Esto es, para una operación dada, puede existir más de una máquina capaz de realizarla, pero no todas las operaciones pueden ser procesadas en cualquier máquina.

Se pretende encontrar la secuencia de operaciones en las máquinas para la fabricación de un pedido de *z* tipos de pieza, con demandas y fechas de entrega diferentes, de manera que se minimice el retraso medio en la entrega de las piezas.

5.3. El Algoritmo de Dispatching

Para resolver el problema de secuenciación de la empresa se diseña una heurística consistente en un *Algoritmo de Dispatching*. Se trata de un método heurístico *constructivo* (se parte de una solución inicial vacía y se van añadiendo elementos siguiendo ciertos criterios para obtener la solución final) y *directo* (una vez se programa una operación no se reconsidera ni modifica en pasos siguientes).

En un momento dado del proceso, el conjunto **E** (operaciones elegibles) está constituido por las operaciones con sus precedentes en el subconjunto **P** (operaciones ya programadas):

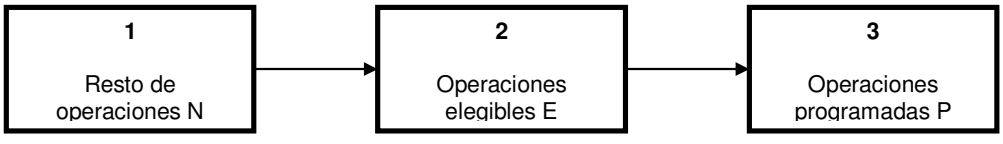


Fig. 5.1. Conjuntos de operaciones del algoritmo de Dispatching

Al programar una operación de una pieza, la operación pasa de **E** a **P** automáticamente y la operación siguiente de dicha pieza pasa de **N** a **E** (salvo si la operación es la última de la pieza).

En un problema *n*/*m* inicialmente se sitúan en **E**, las *n* primeras operaciones de las piezas sin operación precedente, y en **N** las operaciones restantes; **P** está vacío.

En un instante *t*, los subconjuntos **N**, **E** y **P** se hallan en un estado determinado. Cuando el algoritmo termina, todas las operaciones están en **P**.

El conjunto **E** se subdivide en subconjuntos, caracterizados por las operaciones que se han de procesar en la máquina o máquinas. Así, $\mathbf{E_j}$ ($E = E_1 \cap E_2 \cap \ldots \cap E_m$) es el subconjunto de operaciones de **E** a realizar en la máquina ***j***, también llamado cola de operaciones en la máquina *j* (y que puede estar vacío durante la aplicación).

En cada iteración, la programación se realiza en dos fases. En primer lugar se elige la máquina o máquinas con menor instante de disponibilidad para iniciar una nueva operación, y a continuación se escoge la operación a programar entre las operaciones del subconjunto $\mathbf{E_j}$ correspondiente, según las reglas de prioridad establecidas.

5.3.1. Nomenclatura

- **n:** número de piezas.
- **m:** número de máquinas.
- **G(i):** número de operaciones de la pieza i (i = 1…n)
- **H(j):** número de operaciones en la cola de la máquina j (j =1 …m).
- $\mathbf{f_j}$**:** instante en que está disponible la máquina j para una nueva operación (inicialmente coincide con la disponibilidad inicial).
- $\mathbf{p_{k,i,j}}$**:** tiempo de proceso de la operación k de la pieza i en la máquina j.
- $\mathbf{r_{k,i}}$: instante de disponibilidad de la operación (k, i). Es el instante en que la operación llega al sistema, la fecha más temprana en la que la operación k de la pieza i puede ser procesada.
- $\mathbf{Rp_{k,i,j}}$ **:** (*release date* o *ready date*) instante más temprano para empezar la operación (k, i) en la máquina j:

$$rp_{k,i,j} = \max \{ r_{k,i}, f_j \} \qquad \text{para todo } (k,i) \in E_j$$

- **fp_j:** instante más temprano para empezar una nueva operación en la máquina j (si no tiene cola, se asigna valor infinito):

$$fp_j = \min_{(k,i) \in Ej} \{ rp_{k,i,j} \} \qquad \text{si } E_j \text{ no es vacío}$$

$$fp_j = \infty \qquad \text{si } E_j \text{ es vacío}$$

- **$t_{inicio\ (k,\ i)}$:** instante de inicio de fabricación la operación programada.

$$t_{inicio\ (k,\ i)} = rp_{k,i,\ j}$$

- **$t_{final\ (k,\ i)}$:** instante final de fabricación de la operación programada.

$$t_{final\ (k,\ i)} = t_{inicio\ (k,\ i)} + D \cdot p_{k,i,j}$$

5.3.2. Elección de máquina

En el caso que nos ocupa, se elegirá la máquina según las reglas de prioridad siguientes:

- ***Regla 1***: Seleccionar la máquina j que esté disponible más pronto. Es decir, aquella con menor fp_j: $fp_{min} = \min \{ fp_j \}$.

  Si hay empate entre varias máquinas, ir a la *Regla 2*.

- ***Regla 2***: Utilizar las reglas del criterio de elección de operaciones y elegir la máquina cuya operación sea la más prioritaria.

  Si la operación elegida puede realizarse en varias máquinas, ir a *Regla 3*.

- ***Regla 3***: Elegir la máquina más rápida para realizar la operación de la *Regla 2.*

  En caso de empate entre varias máquinas, ir a Regla 4.

- ***Regla 4***: Seleccionar cualquiera.

La *Regla 1* hace que las operaciones empiecen lo antes posible (en cuanto queda libre una máquina se elige y se asigna la operación correspondiente). Su aplicación contribuye a la robustez del programa, puesto que las penalizaciones por los retrasos y los bajos costes de inventario hacen deseable no posponer innecesariamente el procesamiento de las piezas, para prevenir así el impacto de sucesos inesperados que afectan al ritmo de producción.

5.3.3. Elección de operación

Una vez elegida la máquina, es preciso aplicar una regla para seleccionar la operación. Este paso tiene una gran importancia, puesto que determina la forma en la que se ordenan las operaciones en la secuencia solución. Por este motivo es conveniente establecer un orden de prioridad adaptado al objetivo de nuestro caso real: minimizar en lo posible el retraso medio de las entregas. El orden de prioridad elegido, ordenado de mayor a menor importancia, es el siguiente:

- ***Regla 1***: ***Prioridad a la pieza de mayor % de retraso de entrega acumulado en el pedido anterior.*** Selecciona la operación correspondiente a la pieza con mayor número de unidades en retraso. El porcentaje de unidades en retraso se obtiene calculando la diferencia, dividida por 100, entre la cantidad de unidades de la pieza demandadas en el pedido anterior y el número de unidades en retraso de dicho pedido.

  Conviene recordar que la entrega es semanal; cada pedido debe entregarse en un día concreto de la semana actual. Si en la semana actual no se ha podido servir las piezas, entonces se entregan en el pedido de la semana siguiente, en el día que corresponda. Si resulta que en la semana siguiente no hay pedido de esa pieza, se fabrica durante la semana actual y se asigna como fecha de entrega el lunes de esa semana.

- ***Regla 2***: ***Prioridad a la pieza con más días de retraso acumulados desde el pedido anterior.*** Prioriza la operación de la pieza que lleva más días en retraso, entendiéndose como días de retraso la diferencia entre la fecha actual y la fecha en la que debería haberse realizado la entrega en el pedido anterior.

- ***Regla 3***: ***Prioridad a la pieza con fecha de entrega más urgente***. Prioriza la operación de la pieza con fecha de entrega más inmediata en el pedido de la semana actual.

- ***Regla 4: Prioridad a la pieza de mayor duración pendiente.*** Da prioridad a la operación de la pieza cuyas operaciones siguientes comportan más tiempo de proceso. Se calcula sumando los tiempos de proceso de las operaciones pendientes de realizar. Si una operación determinada tiene varios tiempos de proceso posibles (al poderse fabricar en diversas máquinas), se toma el tiempo de proceso mayor.

  Esta regla se ha introducido en el procedimiento como *medida de robustez***.** Indica que los trabajos menos flexibles deben tener mayor prioridad que los más flexibles. En este caso, los trabajos menos flexibles son los de mayor duración de proceso, puesto que corresponden o bien a los de mayor número de operaciones (más tiempo de preparación y más tiempo de máquina), o bien a operaciones de soldadura, cuyas

máquinas correspondientes tienden a convertirse en cuellos de botella. Así, si ocurre un suceso inesperado, los trabajos pendientes de procesar corresponderán a los más flexibles y el impacto en la reprogramación será menor.

- ***Regla 5: Prioridad a la pieza de mayor demanda mensual media.*** Esta regla prioriza la fabricación de las piezas de las que se sabe que tienen una mayor demanda en promedio.

En resumen, se ha elegido un orden de prioridades basado en la necesidad de reducir los retrasos en las entregas, por lo que se da preferencia a la operación de una pieza determinada cuanto más crítico sea su nivel de retraso, medido en porcentaje de retraso acumulado en el pedido anterior (*Regla 1*), cantidad de días de retraso (*Regla 2*), entrega más inmediata (*Regla 3*), más tiempo de proceso pendiente (*Regla 4*) y mayor cantidad media demandada (*Regla 5*).

5.3.4. Esquema del Algoritmo de Dispatching

- **Paso 0 (inicialización).**
  - Situar en el subconjunto de operaciones candidatas las primeras operaciones de las piezas, con sus valores $r_{1,i}$ correspondientes.
  - Para cada máquina j, determinar el valor fp_j.
  - Determinar fp_{min} (y la máquina asociada).
- **Paso 1 (elección de la máquina).**
  - Si $fp_{min} = \infty$, se han programado todas las operaciones.
  - En caso contrario, elegir la máquina según fp_{min} (*Regla 1*). En caso de empate entre varias máquinas elegir máquina mediante las *Reglas 2, 3 y 4* de elección de operaciones.
  - Una vez elegida la máquina, generar el subconjunto de operaciones candidatas, formado por las operaciones elegibles que es capaz de realizar dicha máquina.
- **Paso 2 (elección de la operación).**
  - Si existe una sola operación candidato tomarla.
  - En caso contrario, aplicar las reglas de prioridad de selección de operaciones para elegir la operación a programar.

- **Paso 3 (actualización).**

  - Programar la operación (k, i) elegida fijando sus instantes de inicio ($t_{inicio\ (k,\ i)}$) y de fin ($t_{final\ (k,\ i)}$):

$$t_{inicio\ (k,\ i)} = rp_{k,i,\ j}$$

$$t_{final\ (k,\ i)} = t_{inicio\ (k,\ i)} + D \cdot p_{k,i,j}\ ,$$

    donde D es la demanda de la pieza correspondiente a la operación programada.

  - Pasar la operación elegida a Programada, guardarla en la secuencia con sus instantes de inicio y fin.

  - Si no es la última operación de la pieza *i*, pasar la operación siguiente de *i* de N a E.

  - Sea j' la máquina asociada a esta operación siguiente.

    - **Si j' = j (la misma máquina realiza las dos operaciones consecutivas)**

      En este caso, la segunda operación no puede empezar hasta que se haya procesado la primera, por lo que la operación siguiente (k+1, i) se sitúa en el conjunto de operaciones candidatas con fecha de disponibilidad $r_{(k+1,\ i)}$ igual a la fecha de terminación de la operación anterior $t_{final(k,\ i)}$.

$$\mathbf{r_{(k+1,\ i)} = t_{final\ (k,\ i)}}$$

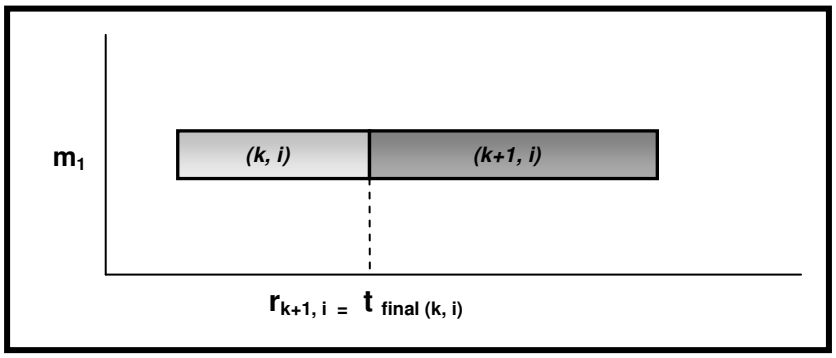


Fig. 5.2. Caso máquinas iguales

- **Si j' ≠ j (las máquinas que procesan las operaciones son distintas)**

  - **Si $p_{(k+1,\ i,\ j')} \geq p_{(k\ ,i\ ,j)}$**

    En este caso el tiempo de proceso de la segunda operación es mayor o igual que el de la primera. La disponibilidad en horas de la operación siguiente $r_{(k+1,\ i)}$ será igual a la fecha de inicio de la operación anterior (k, i) más el tiempo necesario para fabricar y trasladar un lote de transferencia de 15 piezas de la máquina j a la máquina j':

$$r_{(k+1,i)} = t_{inicio(k,\ i)} + (2 + 15p_{k,i,j})/60$$

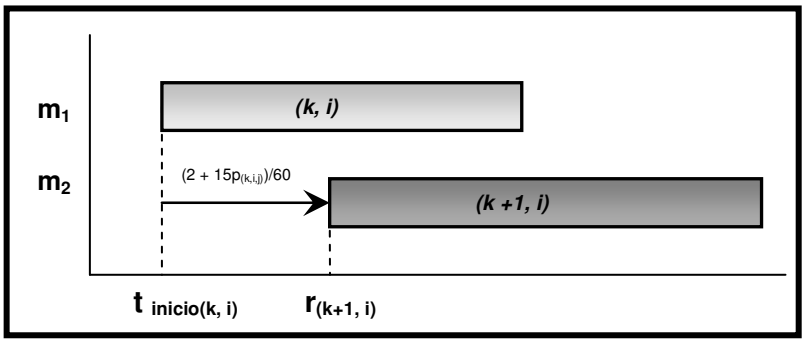


Fig. 5.3. Caso máquinas distintas, $p_{(k+1,\ i,j')} \geq p_{(k,\ i,j)}$

- **Si $p_{(k+1,i,j')} < p_{(k,i,j)}$**

Si el la segunda operación tiene menor tiempo de proceso que la primera, entonces la fecha de disponibilidad en horas de la segunda operación (k+1, i) será igual a la fecha final de la primera operación (k, i) menos el tiempo de proceso de (k+1, i) en la máquina j' más el tiempo necesario para fabricar y trasladar un lote de transferencia de 15 piezas de la máquina j a la j':

$$r_{(k+1,i)} = t_{final(k,\ i)} - D \cdot p_{(k,\ i,\ j')} + (2 + 15p_{k,\ i,\ j})/60$$

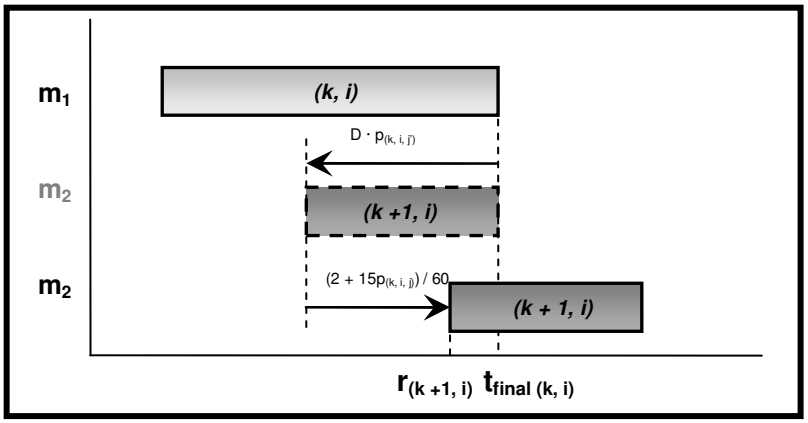


Fig. 5.4. Caso máquinas distintas, $p_{(k+1,\ i,\ j')} < p_{(k,\ i,\ j)}$

- Actualizar los valores fj' de la máquina j' de la siguiente manera:

  Si la máquina j' todavía no se ha utilizado, $f_j = 0$.

  Si se ha programado alguna operación en la máquina j', entonces $f_j = t_{final\ (k,\ i)}$, es decir, la máquina j' tendrá un tiempo de disponibilidad $f_{j'}$ igual al tiempo de terminación de la última operación programada en la máquina j'.

- Calcular $rp_{k,i,\ j'} = \max(r_{k,\ i},\ f_{j'})$

- Determinar $fp_{min} = \min(rp_{k+1,\ i})$

- Volver al paso 1.

- **Paso 4 (función objetivo).**

  - Calcular el retraso medio según la fórmula:

$$\boldsymbol{T_{med}} = \frac{\sum_{i=1}^{n} \max(0, \boldsymbol{C_i} - \boldsymbol{d_i})}{\boldsymbol{n}}, \qquad \text{(Ec. 5.1)}$$

  Donde $\boldsymbol{C_i}$ es el instante de finalización o *completion time* de la pieza ***i***, y $\boldsymbol{d_i}$ es la fecha de entrega o *due date* de la pieza ***i***.

5.3.5. Reglas de robustez

En este apartado se exponen las reglas de robustez que se han integrado en el diseño del algoritmo. El propósito de estas reglas es contribuir a crear un programa más robusto, es decir, más eficiente pese a los sucesos aleatorios que pueden ocurrir durante su ejecución.

Las reglas de robustez empleadas en el programa propuesto son las siguientes:

a) ***No posponer innecesariamente el procesamiento de las operaciones.***
Esta regla de robustez se ha implementado a través de la *Regla 1* de elección de máquina: se prioriza la máquina que está disponible más pronto, de manera que las operaciones empiezan lo antes posible, reduciendo así el impacto de posibles futuros retrasos debidos a algún contratiempo inesperado que altere el ritmo de producción.

b) ***Mantener alimentadas las máquinas cuello de botella.***

Este criterio tiene el propósito de intentar que una máquina nunca deje de estar alimentada a causa de un suceso aleatorio que ocurra en alguna etapa anterior, por lo que supone la aplicación de una *medida de robustez*. El razonamiento es el siguiente: si no se mantiene un inventario frente a una máquina que resulta ser un cuello de botella y la máquina que alimenta el cuello de botella de repente se avería, entonces el cuello de botella debe permanecer ocioso y puede que posteriormente no sea capaz de recuperar el tiempo perdido.

Para ello, se mantiene las máquinas alimentadas con un cierto inventario de piezas. El lote de transferencia entre máquinas es de 15 unidades. Esto significa que la segunda máquina dispondrá, como mínimo, de un suministro de 15 piezas mientras trabaja en paralelo con la primera máquina. En la práctica, es improbable que una máquina se quede sin suministro, puesto que normalmente una máquina cualquiera es más lenta que la máquina que le suministra (la que realiza la operación precedente). Esto se debe a que la operación siguiente de una pieza suele ser más compleja y comporta más tiempo de proceso.

c) ***Priorizar los trabajos menos flexibles.***

Esta regla corresponde a la cuarta regla de elección de operación. Se programan primero los trabajos menos flexibles (los de mayor tiempo de proceso pendiente). De este modo, si ocurre un contratiempo, entonces los trabajos que faltan por entregar son los más flexibles, por lo que resultará más sencillo afrontan la reprogramación de los mismos.

5.3.6. Procedimiento directo

Por defecto el programa se ejecuta, en una primera etapa, según el orden de prioridad de elección de operaciones jerarquizado y fijo proporcionado anteriormente. Es decir, a la hora de seleccionar qué operación se debe fabricar, el algoritmo va ejecutando ordenadamente las reglas, de más a menos prioridad, hasta que obtiene la selección de la operación. Esto es, en primer lugar se ejecuta la *Regla 1*. Si al aplicar esta regla se produce un empate entre varias operaciones, se descarta el resto de operaciones y se va a la *Regla 2* con el grupo de operaciones empatadas, y así hasta llegar a alguna regla que produzca el desempate y proporcione la operación a seleccionar. Si se llega a la última regla con un empate entre varias operaciones, entonces se elige una operación entre cualquiera de ellas.

5.3.7. Procedimiento aleatorio

Una vez se ha obtenido una secuencia inicial mediante el procedimiento directo, se ofrece la posibilidad, en una segunda etapa, de explorar un mayor número de soluciones mediante un procedimiento aleatorio, con el fin de establecer una comparación respecto a la solución inicial y elegir la opción más adecuada.

Este procedimiento se diferencia del directo en que las reglas de prioridad de selección de operaciones no siguen un orden jerarquizado, sino que se ejecutan aleatoriamente según unos pesos (porcentajes) de probabilidad que el usuario asigna a cada regla. De esta manera se consigue una relajación en el orden de prioridad que permite obtener un mayor número de secuencias solución y por tanto, al ejecutarlo un mayor número de veces, aumenta las posibilidades de hallar una solución mejor que la obtenida inicialmente mediante el método directo.

Por defecto se tomarán los siguientes porcentajes de probabilidad para la aplicación de las reglas de prioridad de elección de operaciones: 60, 20, 10, 5 y 5% para las reglas de prioridad 1 a 5 de selección de operaciones.

La selección de la operación a fabricar funciona del siguiente modo:

Se genera un número aleatorio entre 0 y 99; por ejemplo, el 74. Según la **Tabla 5.5**, dicho número pertenece al intervalo de aplicación de la *Regla 2*, por lo que se elige la operación según esta regla. Si, según esta regla, hubiese un empate entre varias operaciones, entonces se continúa visitando las siguientes reglas de forma ordenada (de la 3 a la 5) hasta que alguna de ellas proporcione la operación a elegir. Si se diera un empate final, entonces el programa iría a la *Regla 6* y elegiría al azar una operación cualquiera entre las finalistas.

El usuario podrá ponderar los porcentajes de probabilidad de aplicación de las reglas de prioridad según las necesidades de la planta. Las reglas con mayores porcentajes tendrán una mayor aplicación.

Regla	Descripción	% Probabilidad	Intervalo aplicación
1	Mayor % de unidades en retraso	60	0 a 59
2	Mayor número de días de retraso	20	60 a 79
3	Fecha de entrega más urgente	10	80 a 89
4	Mayor duración pendiente	5	90 a 94
5	Mayor demanda mensual	5	95 a 99

Tabla 5.5. Pesos de probabilidad predeterminados

5.4. Ejemplo práctico

A continuación se resuelve un ejemplo práctico, con el propósito de aclarar el funcionamiento del algoritmo de *dispatching* propuesto.

5.4.1. Datos iniciales

Sea el ejemplar 26 / 16 / G / T_{med}, es decir, un caso del problema del taller mecánico con 26 piezas, 16 máquinas, flujo general y el retraso como medida de eficiencia.

Se deben fabricar 26 piezas (es decir, de las 28 posibles piezas hay 2 piezas con demanda 0). En la **Tabla 5.6** se muestran los datos de la pieza 5, y en el **Anexo A.2** se adjunta la lista completa de piezas con su tiempo de proceso unitario correspondiente.

Pieza	Op.	Tiempo Preparación (min)	Descripción	Máquina	Tiempo proceso unitario (min)	Uds
5	1	60	Cortar formato	A	0,05000	2000
	2	60	Embutir + Cortar	B, R	0,30000	2000
	3	60	Embutir + Cortar	V1, V2,V3, V4	0,30000	2000
	4	60	Embutir + Cortar	V1, V2,V3, V4	0,30000	2000
	5	60	Conformar + Cortar	V1, V2,V3, V4	0,30000	2000

Tabla. 5.6. Datos de la pieza nº5

En la **Tabla 5.7** se muestra el calendario de entregas solicitado para el período del 16 al 20 de marzo de 2009. El lunes de la semana actual, es decir, el día 9, la planta recibe el calendario de pedidos, el cual se fabrica durante dicha semana y se entrega la semana siguiente en las fechas solicitadas en el calendario recibido. Si hubiera retrasos acumulados de pedidos anteriores, éstos se añaden al pedido actual. Obsérvese que en el calendario de pedidos la cantidad solicitada de las piezas número 1 y 4 es cero, por lo que esa semana en particular se fabrican 26 piezas de las 28 posibles. Además, las piezas número 13 y número 23 no tienen demanda pero sí retraso acumulado del pedido anterior, por lo que la cantidad de unidades en retraso del pedido anterior se añaden al pedido actual.

Pieza	Calendario de entregas pedido actual							
	16/03/2009	17/03/2009	18/03/2009	19/03/2009	20/03/2009	Días de retraso	Uds Retrasos	Uds Total Pedido
1						1		0
2					400	1	200	600
3		300					165	465
4								0
5		2000						2000
6	2275					2	650	2925
7					5000			5000
8				3600				3600
9		200						200
10		200						200
11				250		1	176	426
12					24400			24400
13						3	250	250
14			2800					2800
15				1400		1	350	1750
16			750					750
17	900							900
18					2000			2000
19			2100			1	190	2290
20		640						640
21				2300		2	40	2340
22	300							300
23						2	600	600
24				400				400
25	300							300
26			1250					1250
27			1000					1000
28				1820				1820

Tabla 5.7. Calendario de entregas del 16/03/09 al 20/03/09

5.4.2. Reglas de prioridad de máquinas

La selección de las máquinas se realiza según el proceso explicado en el **apartado 5.3.2**. En cada iteración, se siguen las 4 reglas de prioridad en el orden descrito (de la primera a la cuarta) y se selecciona la máquina que realizará la operación.

5.4.3. Reglas de prioridad de operaciones

Una vez elegida la máquina, se debe seleccionar la operación a fabricar. En este ejemplo concreto se tienen los siguientes datos necesarios para la aplicación de estas reglas:

Regla 1) Elegir la pieza con mayor porcentaje de unidades en retraso.

Se da preferencia a la operación correspondiente a la pieza con mayor porcentaje de unidades en retraso en el pedido anterior. Se obtiene calculando la diferencia entre la cantidad de piezas pedidas y el número de piezas en retraso y dividiendo el resultado entre 100:

REGLA 1: Prioridad por porcentaje de unidades en retraso				
Orden prioridad	**Numero pieza**	**Nº piezas Retraso**	**Nº piezas pedido anterior**	**% retraso**
1	13	250	250	**100**
1	23	600	600	**100**
2	2	200	500	**40**
3	11	176	500	**35,2**
4	6	650	2000	**32,5**
5	3	165	600	**27,5**
6	15	350	1800	**19,44**
7	19	190	2170	**8,75**
8	21	40	1000	**4**
9	1	0	0	**0**
9	4	0	0	**0**
9	5	0	0	**0**
9	7	0	0	**0**
9	8	0	0	**0**
9	9	0	0	**0**
9	10	0	0	**0**
9	12	0	0	**0**
9	14	0	0	**0**
9	16	0	0	**0**
9	17	0	0	**0**
9	18	0	0	**0**
9	20	0	0	**0**
9	22	0	0	**0**
9	24	0	0	**0**
9	25	0	0	**0**
9	26	0	0	**0**
9	27	0	0	**0**
9	28	0	0	**0**

Tabla 5.8. Prioridad según % retraso

Regla 2) Elegir la pieza con más días de retraso

Prioriza la pieza con mayor tiempo de retraso. La tabla siguiente muestra los valores de nuestro caso correspondientes a los días de retraso acumulados por las piezas en el pedido anterior:

REGLA 2: Prioridad por días de retraso		
Orden prioridad	**Numero pieza**	**Días Retraso**
1	13	**3**
2	23	**2**
3	6	**2**
4	21	**2**
5	2	**1**
6	11	**1**
7	3	**1**
8	15	**1**
9	19	**1**
10	1	**0**
10	4	**0**
10	5	**0**
10	7	**0**
10	8	**0**
10	9	**0**
10	10	**0**
10	12	**0**
10	14	**0**
10	16	**0**
10	17	**0**
10	18	**0**
10	20	**0**
10	22	**0**
10	24	**0**
10	25	**0**
10	26	**0**
10	27	**0**
10	28	**0**

Tabla 5.9. Prioridad según número de días de retraso

Regla 3) Elegir la pieza con fecha de entrega más urgente.

Da preferencia a la pieza que se debe entregar antes. A continuación se muestra el orden de prioridad de las piezas según las fechas de entrega pactadas en este caso. Como ya se explicó anteriormente, algunas piezas no tienen fecha de entrega. La pieza 4, por ejemplo, no se ha pedido esa semana, por lo que no se tiene en cuenta. Las piezas 13 y 23 son piezas en retraso por lo que se les dará carácter de urgencia y por defecto se les asignará el lunes como fecha de entrega.

REGLA3: Prioridad por fecha de entrega más urgente		
Orden prioridad	**Numero pieza**	**Fecha de entrega**
1	13	**En retraso (lunes)**
1	23	**En retraso (lunes)**
1	6	**lunes**
1	17	**lunes**
1	22	**lunes**
1	25	**lunes**
2	3	**martes**
2	5	**martes**
2	9	**martes**
2	10	**martes**
2	20	**martes**
3	14	**miércoles**
3	16	**miércoles**
3	19	**miércoles**
3	26	**miércoles**
3	27	**miércoles**
4	8	**jueves**
4	11	**jueves**
4	15	**jueves**
4	21	**jueves**
4	24	**jueves**
4	28	**jueves**
5	2	**viernes**
5	7	**viernes**
5	12	**viernes**
5	18	**viernes**
6	1	**No hay pedido**
6	4	**No hay pedido**

Tabla 5.10. Prioridad según fecha de entrega

Regla 4) Elegir la pieza con mayor duración pendiente.

Da prioridad a la pieza con operaciones siguientes que comportan más tiempo de proceso pendiente. Se trata de una regla dinámica, puesto que la duración pendiente de una pieza varía a medida que avanza la programación y va progresando su nivel de procesamiento. En un instante de tiempo determinado, la duración pendiente de una pieza corresponde al sumatorio de los tiempos de proceso de sus operaciones que faltan por realizar. En caso de existir varios tiempos de proceso posibles (al disponer de varias máquinas distintas para realizar la operación), se toman los tiempos de proceso más largos.

Regla 5) Elegir la pieza de mayor demanda mensual media.

Prioriza las piezas más demandadas en promedio. En nuestro caso, el orden de prioridades es:

Regla 5: Prioridad por demanda		
Orden prioridad	**Numero pieza**	**Demanda media**
1	12	**25000**
2	7	**4500**
3	8	**3500**
4	6	**2500**
4	14	**2500**
5	19	**2300**
6	21	**2000**
7	18	**1800**
7	5	**1800**
8	28	**1500**
8	15	**1500**
9	26	**1000**
9	16	**1000**
10	27	**800**
10	17	**800**
11	23	**750**
12	2	**600**
13	20	**500**
13	1	**500**
13	3	**500**
14	4	**400**
14	11	**400**
14	24	**400**
15	25	**350**
16	22	**275**
17	9	**200**
17	10	**200**
17	13	**200**

Tabla 5.11. Prioridad según demanda media

5.4.4. Simulación

El programa se obtiene ejecutando el algoritmo de *dispatching* descrito en el **apartado 5.3.4**. A modo de ejemplo se explica una iteración escogida al azar:

Iteración	Elegibles	p_{kij} (h)	Máquinas posibles	$r_{k,i}$	f_j	$rp_{k,i}$	Empieza (h)	Termina (h)	fp_{min}
58	(1, 20)	8,333	Y1	0	33,583	33,583	33,583	41,916	**30,348**
		8,333	Y2	0	40,566	40,566	40,566	48,899	
	(1, 24)	1,615	B	0	40,062	40,062	40,062	41,677	
		1,615	R	0	40,140	40,140	40,140	41,755	
	(1, 27)	5,166	B	0	40,0623	40,062	40,062	45,229	
		5,166	R	0	40,140	40,140	40,140	45,306	
	(1, 28)	19,2	B	0	40,062	40,062	40,062	59,262	
		19,2	R	0	40,140	40,140	40,140	59,340	
	(5, 15)	9,083	X1	16,401	44,702	44,702	44,702	53,786	
		9,083	**X2**	**16,401**	**30,348**	**30,348**	**30,348**	**39,431**	
	(2, 16)	6	B	25,923	40,062	40,062	40,062	46,062	
		6	R	25,923	40,140	40,140	40,140	46,140	
		4,75	E	25,923	33,955	33,955	33,955	38,705	
	(6, 7)	25,333	X1	26,432	44,702	44,702	44,702	70,036	
		25,333	X2	26,432	30,348	30,348	30,348	55,681	
	(5, 25)	2,333	X1	28,401	44,702	44,702	44,702	47,036	
		2,333	X2	28,401	30,348	30,348	30,348	32,681	
	(5, 5)	11	V1	29,279	40,170	40,170	40,170	51,170	
		11	V2	29,279	40,279	40,279	40,279	51,279	
		11	V3	29,279	33,221	33,221	33,221	44,221	
		11	V4	29,279	39,861	39,861	39,861	50,861	
	(3, 14)	5,666	B	34,556	40,062	40,062	40,062	45,729	
		5,666	R	34,556	40,140	40,140	40,140	45,806	
	(3, 26)	4,571	B	29,906	40,062	40,062	40,062	44,633	
		4,571	R	29,906	40,140	40,140	40,140	44,711	
		4,125	E	29,906	33,955	33,955	33,955	38,08	

Tabla 5.12. Ejemplo de cálculo del algoritmo

El algoritmo de *dispatching* contiene tantas iteraciones (decisiones) como operaciones se deben programar. En nuestro caso se tienen 85 operaciones y, por tanto, 85 iteraciones a realizar.

Veamos, como ejemplo, el cálculo de la iteración número 58. Por columnas, se tiene (todos los tiempos expresados en horas):

- ***Elegibles*** Lista de operaciones de las piezas candidatas a ser programadas.
- ***Máquinas*** Lista de máquinas capaces de realizar la operación.
- $p_{k,i,j}$ Tiempo de proceso total de la operación en la máquina.
- $r_{k,i}$ Instante más temprano en que está disponible la operación.
- f_j Instante más temprano en que queda disponible la máquina j.
- $rp_{k,i,j}$ Instante más temprano en el que se puede procesar la operación.
- ***Empieza*** Instante más temprano en el que se podría procesar la operación.
- ***Termina*** Instante más temprano en el que se podría procesar la operación.
- fp_{min} El valor mínimo de los instantes de disponibilidad de las máquinas.

En el algoritmo la asignación y la secuenciación se realizan conjuntamente en cada iteración. Se calcula el valor de $\mathbf{fp_{min}}$ y se elige la máquina correspondiente a este valor, puesto que se trata de la máquina que puede empezar más pronto; en este caso es la máquina X2. Si hay varias máquinas con el mismo valor, entonces se elige la máquina correspondiente a la operación más prioritaria. Hay tres operaciones en la cola de X2: la (5, 15), la (6, 7) y la (5, 25). Se utiliza el criterio de prioridad de selección de operaciones y la primera regla proporciona la operación más prioritaria: la (5, 15).

De este modo, se elige la máquina X2 para procesar la quinta operación de la pieza número 15, y se asignan los valores de inicio y de fin de proceso.

En la iteración siguiente se actualiza el valor de disponibilidad de la máquina X2 (tomando el valor de terminación de la operación asignada) y se añade la operación siguiente de la pieza (si la tiene) a la lista de elegibles. En ese momento se repite el proceso explicado para seleccionar la siguiente operación a asignar.

5.4.5. Resultados

Se obtiene la siguiente secuencia de fabricación de las operaciones de las piezas en las máquinas, con sus instantes de principio y fin.

Orden	Operación	Pieza	Máquina	Empieza(h)	Termina(h)
1	1	13	Y1	0	6,58
2	1	23	I1	0	1,75
3	1	2	A	0	1,33
4	1	11	Y2	0	10,98
5	1	3	V1	0	1,93
6	1	15	R	0	6,83
7	1	19	E	0	5,58
8	1	17	B	0	5,50
9	2	2	V2	0,04	2,37
10	2	23	Z1	0,05	6,38
11	2	3	V3	0,06	1,99
12	1	6	A	1,33	4,77
13	2	6	V4	1,38	10,74
14	3	6	V1	2,50	10,81
15	4	6	V2	4,03	10,88
16	5	6	V3	4,10	19,72
17	1	5	A	4,77	7,44
18	2	19	B	5,50	11,59
19	2	15	E	5,58	10,95
20	3	23	Y1	6,58	12,92
21	3	19	R	6,83	12,92
22	1	14	A	7,44	11,94
23	6	6	V4	10,74	26,36
24	3	15	E	10,95	16,33
25	1	21	Y2	10,98	40,56
26	4	19	B	11,59	17,68
27	5	19	X1	11,65	44,70
28	1	8	A	11,94	13,94
29	2	8	I1	11,97	14,61
30	3	8	X2	12,01	30,35
31	1	10	Y1	12,92	23,25
32	2	17	R	12,92	19,92
33	1	7	A	13,94	23,27
34	2	7	V1	13,99	27,50
35	3	7	V2	14,07	27,57
36	4	15	E	16,33	21,70
37	1	22	B	17,68	19,14
38	3	17	B	19,14	26,14
39	4	7	V3	19,72	33,22
40	2	22	R	19,92	21,38
41	3	22	Z1	19,98	30,31
42	4	17	R	21,38	26,88
43	1	26	E	21,70	25,83
44	1	9	Y1	23,25	33,58
45	1	12	A	23,27	27,65
46	1	16	E	25,83	29,83
47	1	25	B	26,14	27,60
48	5	7	V4	26,36	39,86
49	2	25	R	26,88	28,34

50	3	25	B	27,61	29,06
51	1	18	A	27,64	28,88
52	4	25	R	28,35	29,81
53	2	5	B	29,06	40,06
54	3	5	V1	29,17	40,17
55	4	5	V2	29,28	40,28
56	2	14	R	29,81	40,14
57	2	26	E	29,83	33,95
58	5	15	X2	30,35	39,43
59	5	5	V3	33,22	44,22
60	1	20	Y1	33,58	41,92
61	2	16	E	33,95	38,70
62	3	26	E	38,70	42,83
63	5	25	X2	39,43	41,76
64	1	27	B	40,06	45,23
65	3	14	R	40,14	45,81
66	4	26	X2	41,76	50,43
67	2	27	E	42,83	48,83
68	6	7	X1	44,70	70,04
69	4	14	B	45,23	52,45
70	5	14	R	45,81	53,03
71	3	27	E	48,83	54,83
72	4	27	Z1	48,94	82,60
73	6	14	X2	50,43	78,76
74	3	16	B	52,45	58,45
75	1	28	R	53,03	72,23
76	2	28	B	58,45	77,65
77	4	16	X1	70,04	72,87
78	3	28	R	72,23	91,43
79	4	28	E	72,41	91,61
80	1	24	B	77,65	79,27
81	2	24	B	79,27	80,88
82	3	24	B	80,88	82,49
83	4	24	B	82,49	84,11
84	5	24	X1	82,55	84,22
85	5	28	E	91,61	110,81

Tabla 5.13. Secuencia resultante de la simulación del ejemplo práctico

El valor de la función objetivo (retraso medio), la relación de piezas que terminan tarde y su tiempo de retraso correspondiente, son los mostrados en la **Tabla 5.14**.

Pieza	Máquina	Empieza(h)	Termina(h)	Fecha entrega(h)	Retraso
2	V2	0,04	2,37	0	2,37
3	V3	0,06	1,99	0	1,99
5	V3	33,22	44,22	128	0
6	V4	10,74	26,36	0	26,36
7	X1	44,70	70,04	176	0
8	X2	12,01	30,35	160	0
9	Y1	23,25	33,58	128	0
10	Y1	12,92	23,25	128	0
11	Y2	0	10,98	0	10,98
12	A	23,27	27,65	176	0
13	Y1	0	6,58	0	6,58
14	X2	50,43	78,76	144	0
15	X2	30,35	39,43	0	39,43
16	X1	70,04	72,87	144	0
17	R	21,38	26,88	112	0
18	A	27,65	28,88	176	0
19	X1	11,65	44,70	0	44,70
20	Y1	33,58	41,92	128	0
21	Y2	10,98	40,57	0	40,57
22	Z1	19,98	30,31	112	0
23	Y1	6,58	12,92	0	12,92
24	X1	82,55	84,22	160	0
25	X2	39,43	41,76	112	0
26	X2	41,76	50,43	144	0
27	Z1	48,94	82,60	144	0
28	E	91,61	123,01	160	0
				Retraso medio (h):	**7,15**

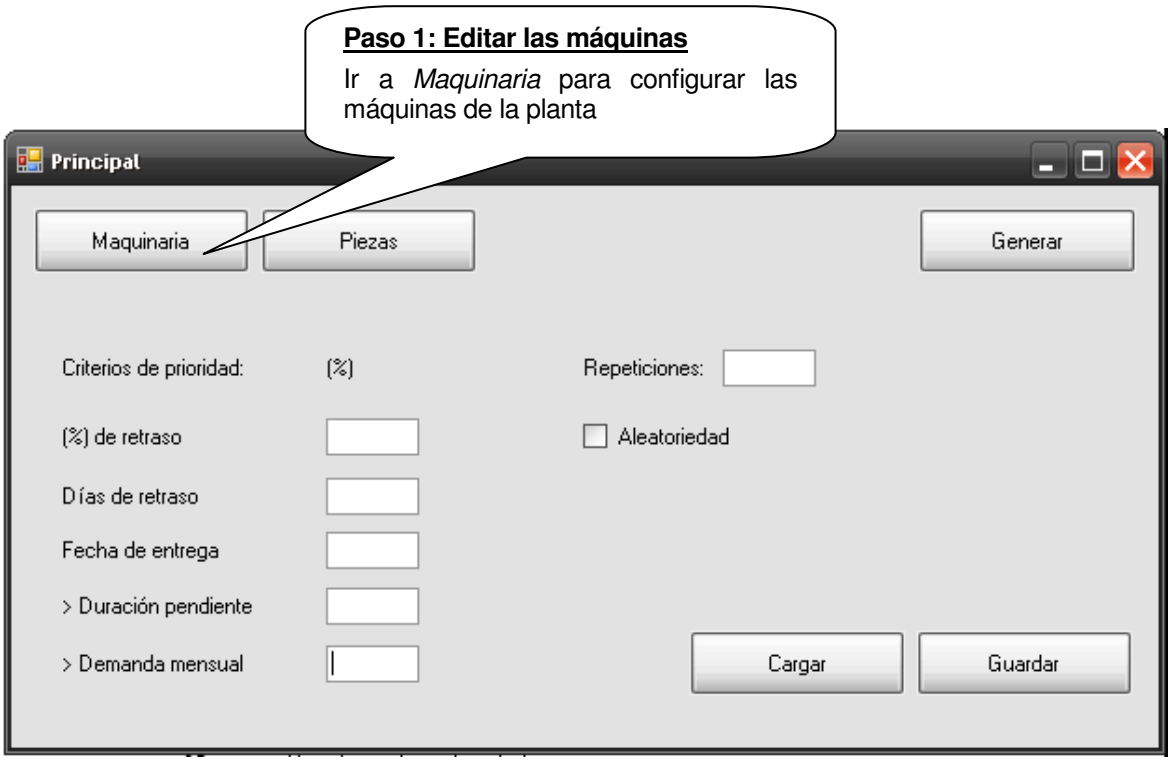
Tabla 5.14. Valores de retraso obtenidos en la resolución del ejemplo práctico

5.5. Manual de usuario de la aplicación informática

El algoritmo de *dispatching* diseñado se ha implementado en una aplicación informática cuyo funcionamiento se describe en este capítulo. El código fuente ha sido implementado en lenguaje C# y compilado en Microsoft Visual Studio 2005. El texto completo del código fuente desarrollado se adjunta en el **Anexo B**.

5.5.1. Introducción de datos

La aplicación es un fichero ejecutable denominado *Secuenciador* que se puede instalar en cualquier directorio del PC. Antes de empezar a utilizar el programa es necesario crear una carpeta de nombre *tmp* en la unidad C. Al ejecutar el archivo se abre la ventana ***Principal*** del programa:

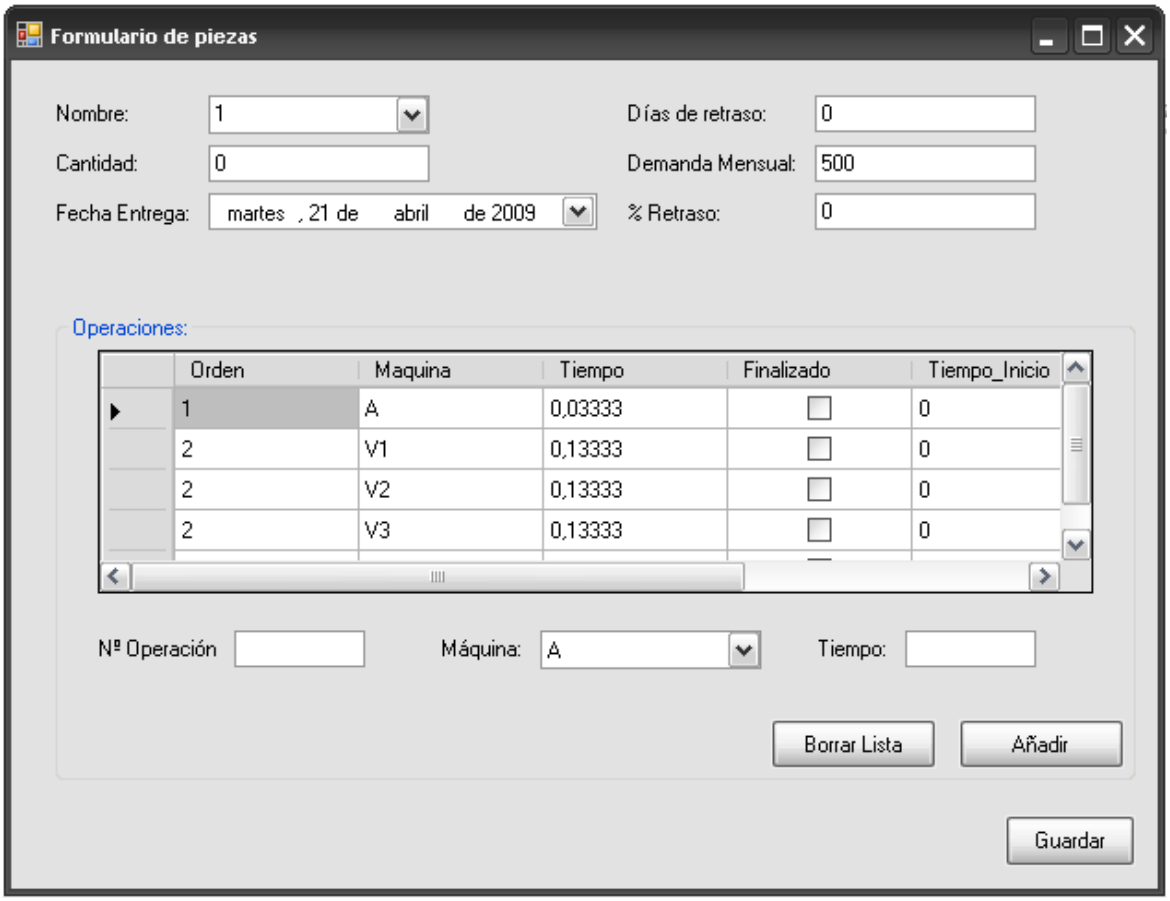


Aparece la ventana ***Maquinaria***, donde se deben rellenar tres campos:

- ***Máquina:*** Se refiere al nombre de la máquina.
- ***Tiempo de inicio:*** Es el instante, en horas, a partir del cual la máquina está disponible. Por defecto se asigna el valor cero.

- ***Tiempo de preparación:*** Tiempo de preparación, en minutos, necesario para realizar una nueva operación. En nuestro caso este tiempo es de 60 para las prensas y 20 para las máquinas de soldar.

Aparece la ventana ***Formulario de Piezas***:



Se debe rellenar los siguientes campos:

- ***Nombre:*** Se refiere al nombre de la pieza, el cual se ha de indicar en el menú desplegable de este campo.

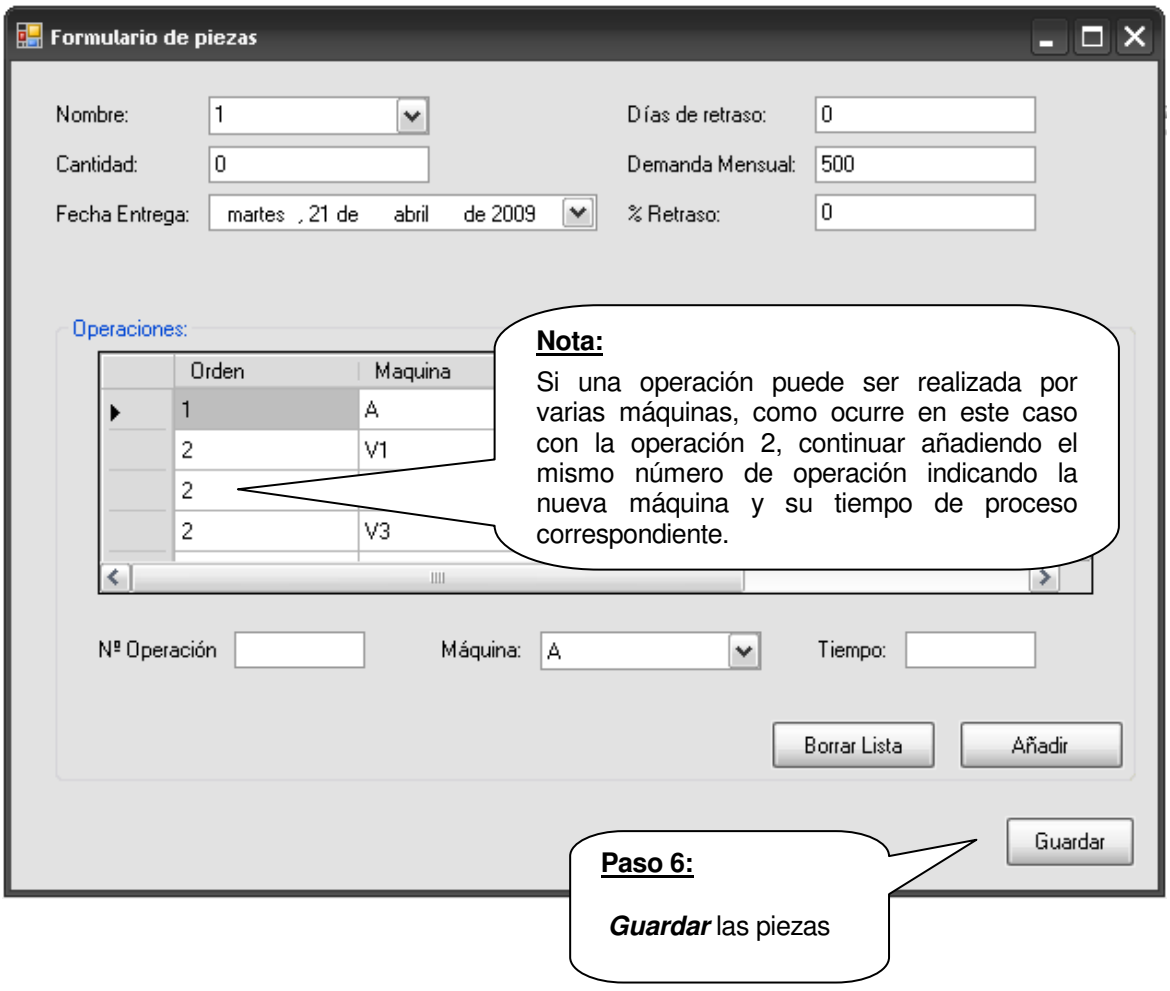
- ***Cantidad:*** El número de unidades a fabricar.

- ***Fecha de entrega:*** Seleccionar en el calendario desplegable la fecha de entrega de la pieza. Si una pieza lleva retraso respecto al pedido anterior, se debe seleccionar un día del calendario que sea domingo. El programa entiende que el tiempo de fabricación que se emplea en dicha pieza debe ser considerado como tiempo de retraso.

- ***Días de retraso:*** En caso de que se trate de una pieza con retraso acumulado en el pedido anterior, indicar el número de días que lleva en retraso.
- ***Demanda mensual:*** Indicar el valor medio de demanda mensual estimada.
- ***% retraso:*** En caso de que sea una pieza con retraso acumulado en el pedido anterior, indicar el porcentaje de unidades de retraso respecto a la cantidad requerida en dicho pedido.

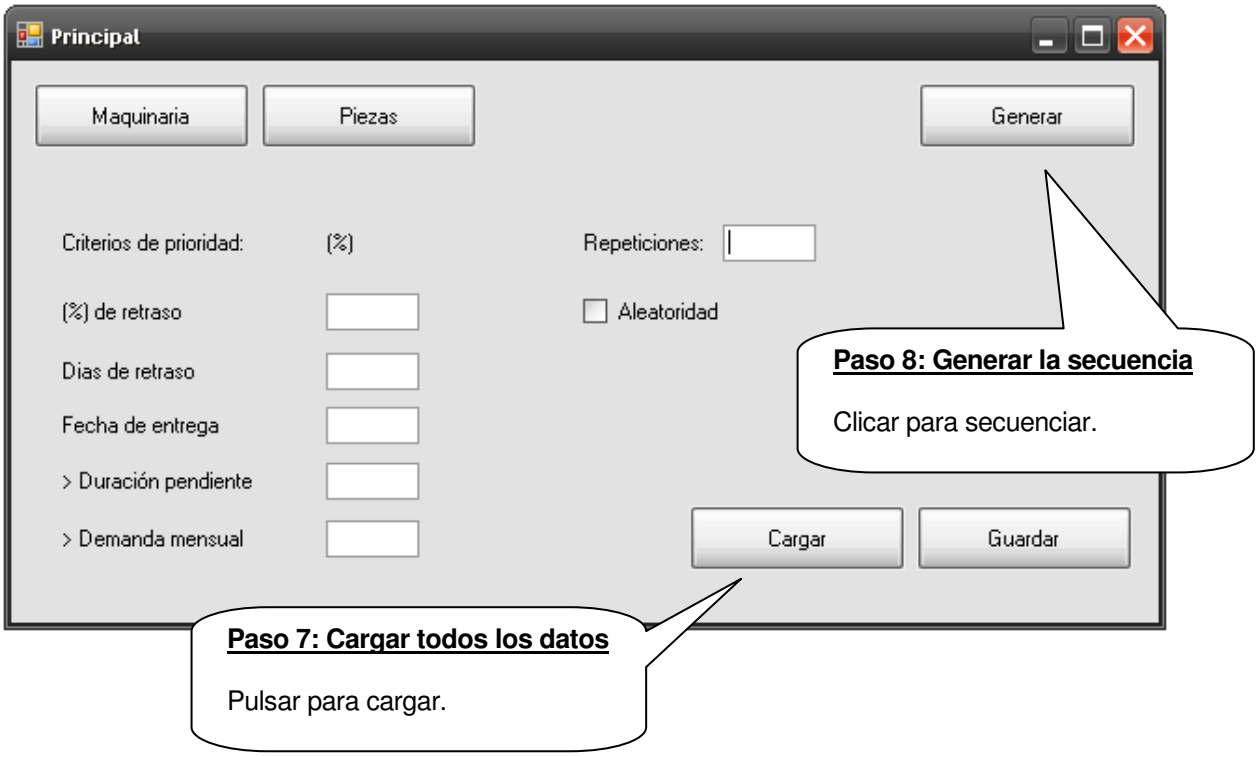
Las operaciones de la pieza se añaden a la lista de la misma manera que se añaden las máquinas. En este caso se deben rellenar los siguientes campos:

- ***Nº Operación:*** Introducir el número de operación de la pieza.
- ***Máquina:*** La máquina en la que puede realizarse la operación.
- ***Tiempo:*** Tiempo de proceso de la operación en la máquina indicada anteriormente.

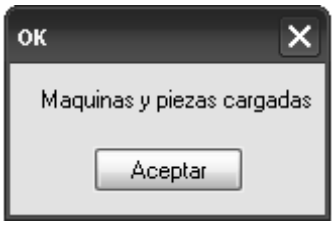


En la lista Operaciones los campos *Finalizado*, *Tiempo_Inicio*, *Tiempo_Fin* y *Pieza* son campos internos de control del algoritmo, por lo que deben ser ignorados.

Una vez guardada la configuración de las piezas, el programa vuelve de nuevo a la ventana ***Principal***, donde se debe ***Cargar***, para que inicialice la configuración de piezas y máquinas y a continuación ***Generar***, para crear la secuencia de operaciones.



Al ***Cargar*** aparecerá el siguiente mensaje de confirmación:

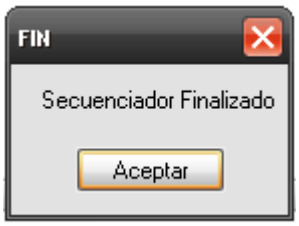


5.5.2. Cálculo de la secuencia: Procedimiento directo

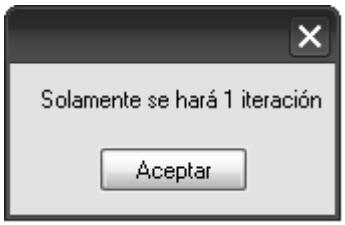
El algoritmo de *dispatching* calcula la elección de la operación siguiendo el procedimiento jerárquico. Es decir, comprueba en primer lugar la primera regla de prioridad. Si dicha regla no sirve para desempatar las operaciones candidatas, procede a verificar la segunda regla,

y así sucesivamente, hasta que llegar a una regla que permite elegir la operación a programar.

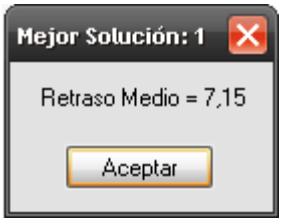
Para generar la secuencia de operaciones según el procedimiento directo (una sola secuencia) se debe ir al botón ***Generar***. Al finalizar el cálculo el programa muestra el mensaje siguiente:



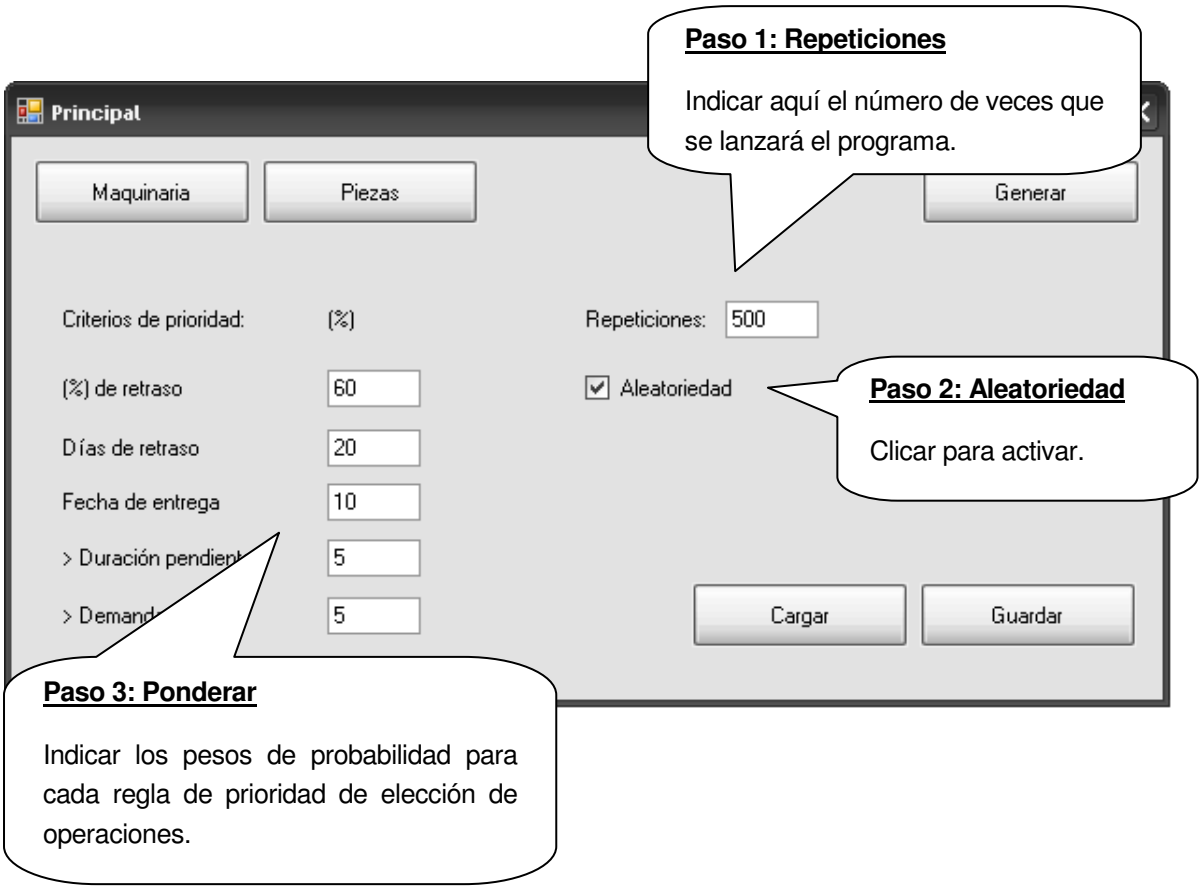
En el caso no aleatorio (la casilla ***Aleatorio*** debe estar desactivada) solamente se calcula una secuencia, por lo que se muestra el recordatorio:



Y a continuación aparece el valor del retraso medio (en horas) producido en la secuencia obtenida y el número de repetición que ha proporcionado dicho valor:



5.5.3. Cálculo de la secuencia: Procedimiento aleatorio



El programa ofrece la posibilidad de ponderar los criterios de prioridad en la elección de las operaciones, introduciendo un valor de probabilidad.

Activando la casilla ***Aleatoriedad***, se asigna una probabilidad a cada una de las reglas de prioridad, por lo que el algoritmo considerará las reglas de forma desordenada, en función del número aleatorio que se obtenga en cada caso.

En el ejemplo, se ha indicado la siguiente ponderación de probabilidades:

Regla	% Probabilidad	Rango aleatorio
1	60	0-59
2	20	60-79
3	10	80-89
4	5	90-94
5	5	95-99

Tabla 5.15. Ponderación de la probabilidad de reglas de prioridad

Supóngase que se debe seleccionar una operación y el programa ha generado el número aleatorio 50. Entonces, se comprobará en primer lugar la primera regla, y si hay un empate entre operaciones, entonces continúa considerando en orden incremental el resto de reglas, hasta que se obtiene la operación a programar. Pero si el número aleatorio es, por ejemplo, 83, entonces es la regla 3 la que se comprobará en primer lugar, y si no determina la operación, entonces se comprueba la regla 4, y después la regla 5, si fuera necesario. Si esta última regla no proporciona una la operación a programar, entonces se selecciona una cualquiera entre las finalistas.

En la casilla ***Repeticiones*** se ha de introducir el número de secuencias que se desean obtener. No existe límite para este campo (y el programa calcularía soluciones hasta que se agotase la memoria). A título orientativo, si se introducen 1000 repeticiones, se genera un archivo de solución en torno a los 10Mb.

5.5.4. Ficheros solución

El programa crea 4 ficheros .txt en el directorio C:\tmp:

- ***maquinas.txt:*** Guarda la configuración de las máquinas introducida por el usuario en la ventana *Maquinaria*.

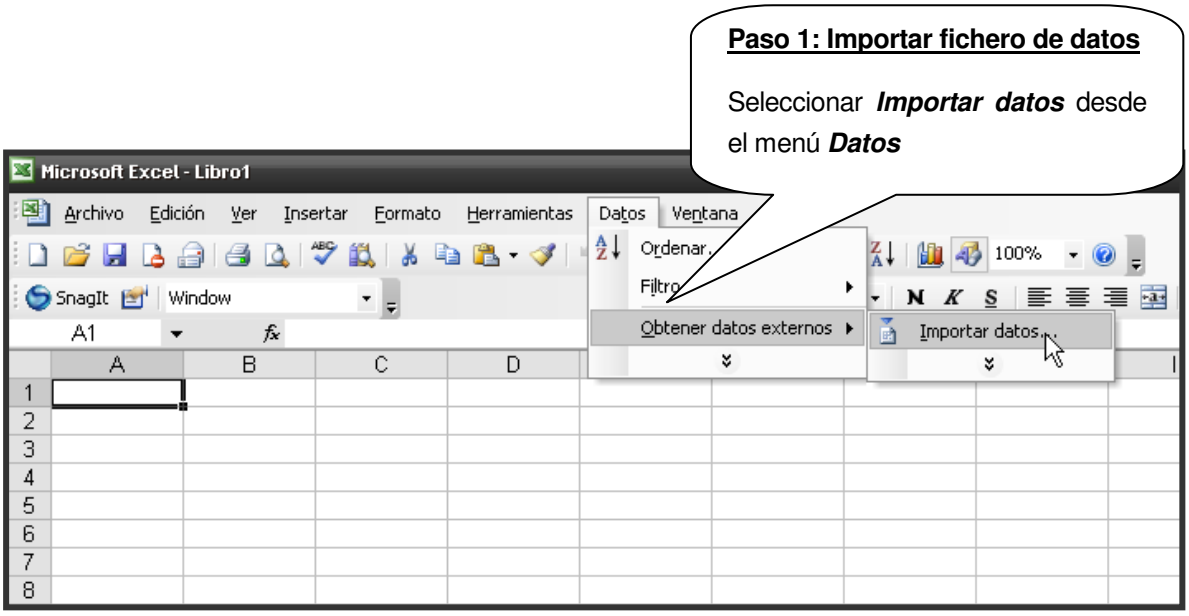
- ***piezas.txt:*** Guarda la lista de piezas configuradas en la ventana *Formulario de Piezas*.

- ***retrasos.txt:*** Muestra la lista de piezas en retraso, el valor del retraso de cada una de ellas y el número de piezas con retraso. Las columnas se refieren, por este orden, al número de repetición, el número de pieza, el número de la última operación y el valor del retraso respecto a la fecha de entrega.

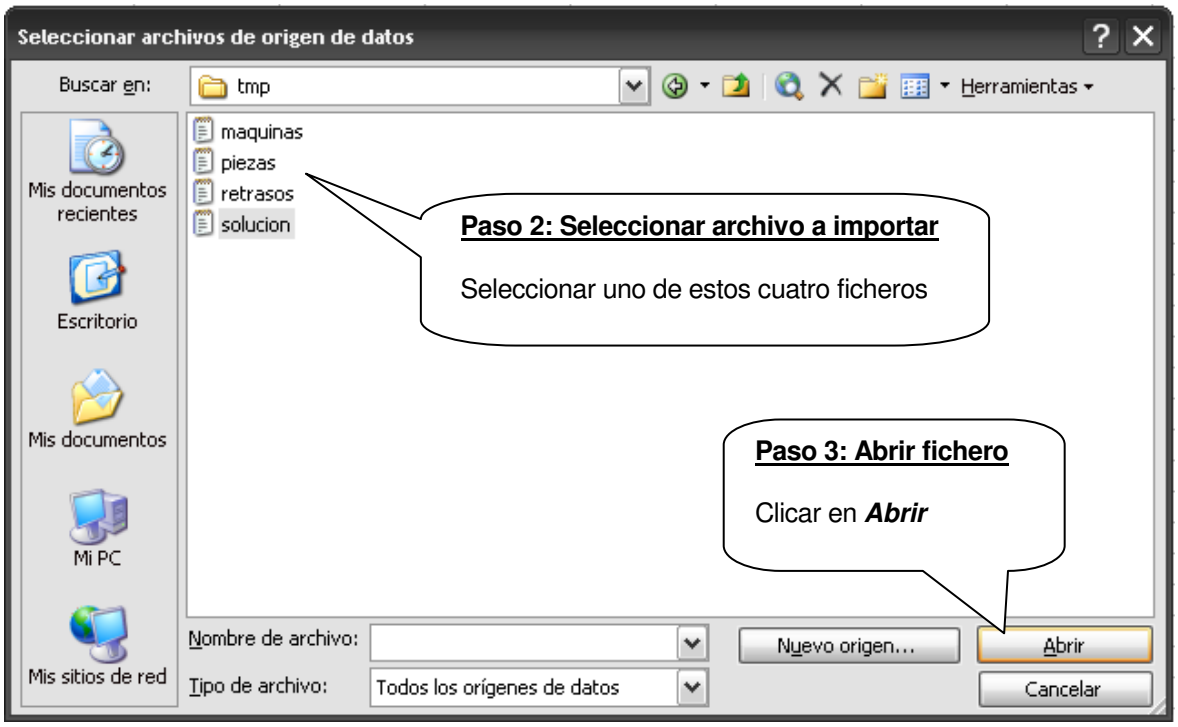
- ***solucion.txt:*** Muestra la secuenciación obtenida, indicando el orden de cada una de las operaciones, la máquina a la que deben asignarse y los instantes de inicio y de fin en los que deben ser programadas. También muestra el valor del retraso medio obtenido para cada secuencia. La **Tabla 5.23** se muestra el resultado obtenido en formato Excel.

Los ficheros obtenidos se abren con la aplicación *Bloc de notas* en formato *txt*, pero para mayor comodidad es recomendable importar los resultados desde una hoja de cálculo de Excel, siguiendo los pasos que se indican:

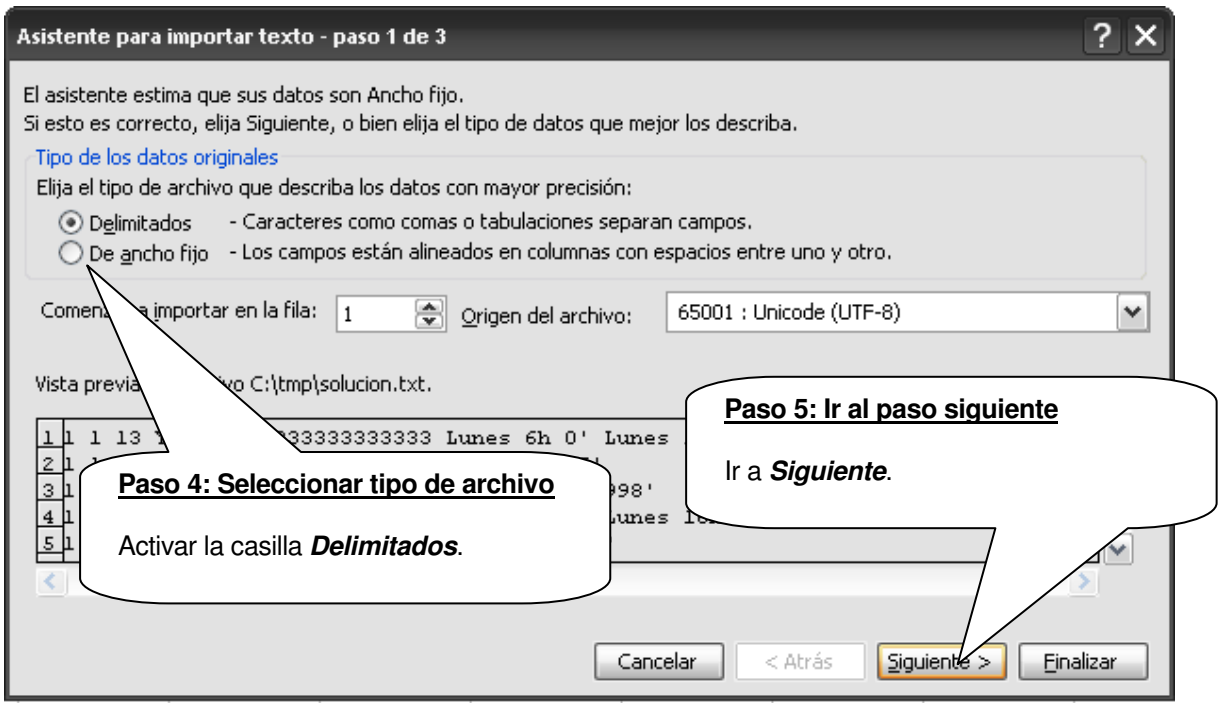
En primer lugar, abrir una hoja de Excel e ir al menú *Datos/ Obtener datos externos/ Importar datos…* tal como muestra la imagen siguiente:



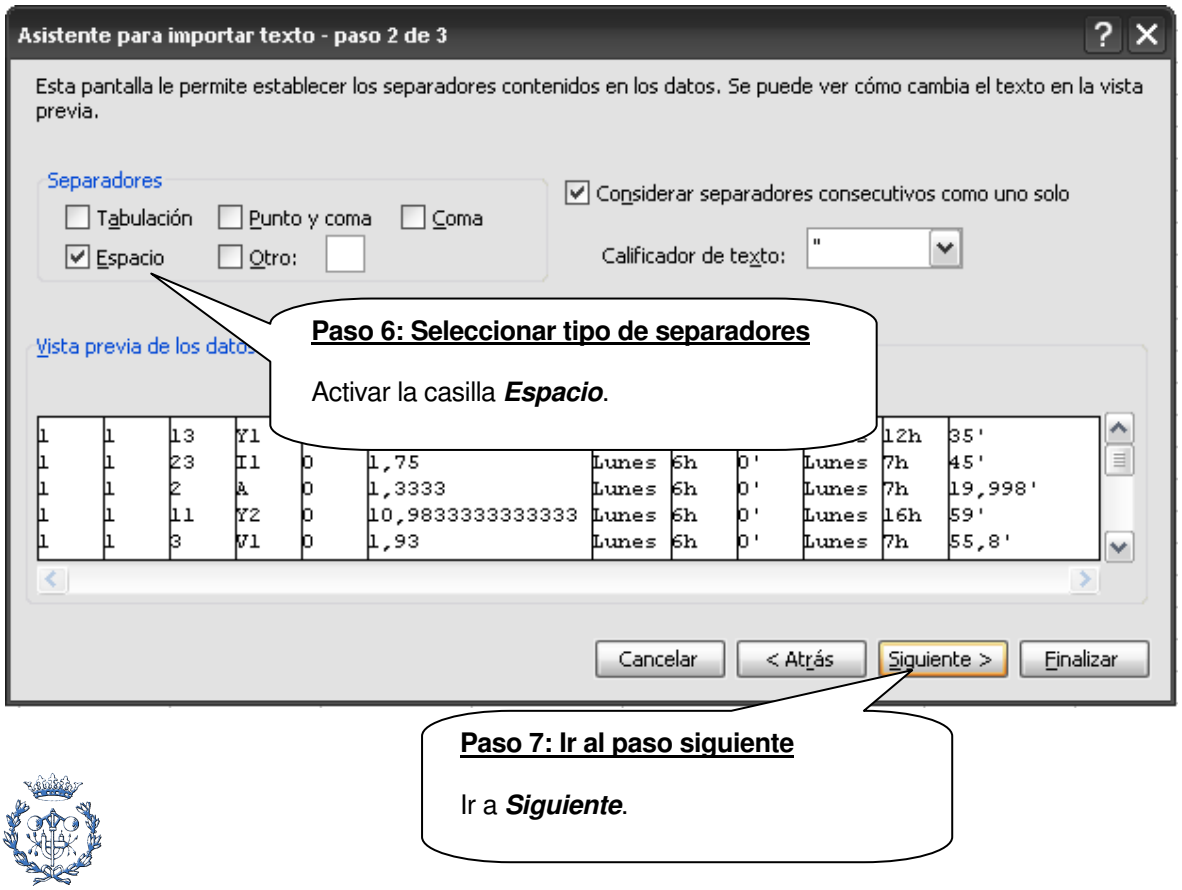
Aparece una ventana en la que se debe seleccionar el directorio ***Mi PC/ C:/ tmp/,*** en el que se encuentran los cuatro ficheros txt que genera el programa. A continuación se selecciona el fichero que interese abrir, por ejemplo, el archivo ***solucion.txt***, que es el que contiene la secuenciación de las operaciones, y clicar en ***Abrir***:



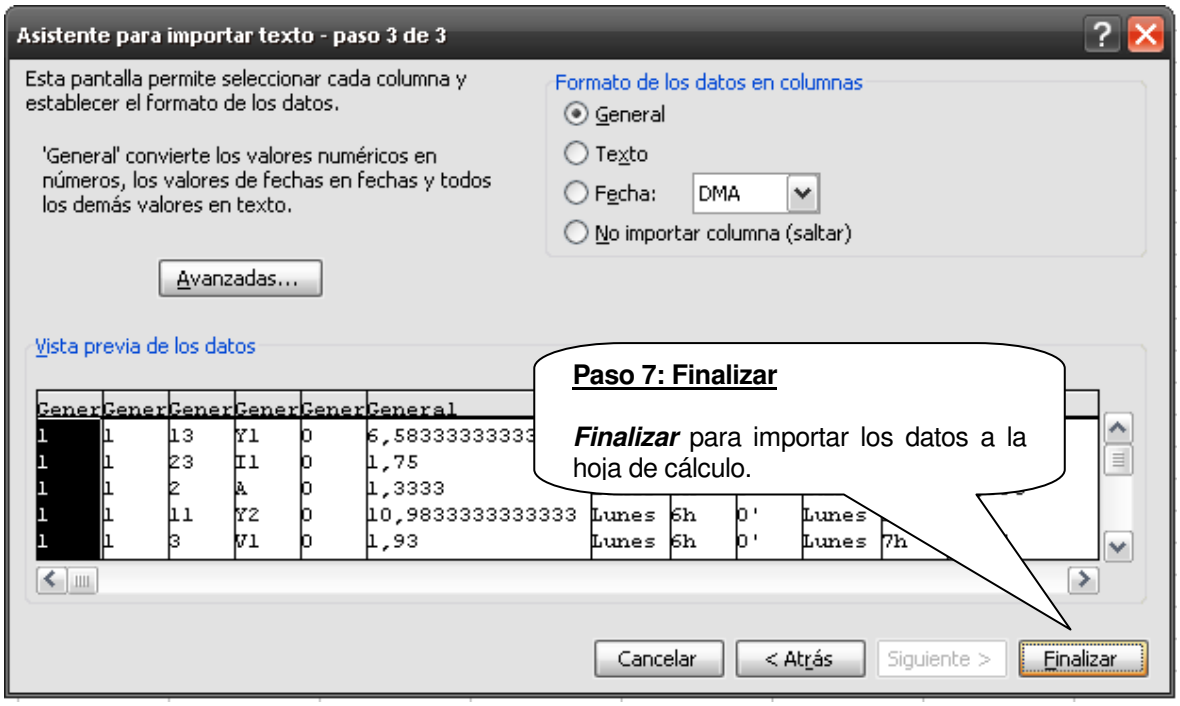
A continuación aparece el asistente para importar los datos desde el fichero seleccionado anteriormente. Se debe activar la casilla ***Delimitados***:



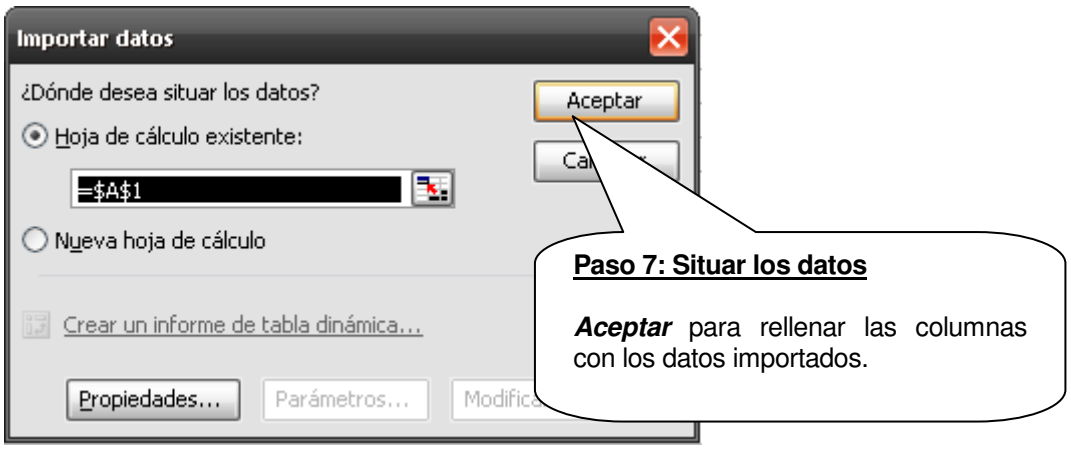
En la pantalla siguiente, de debe seleccionar el tipo de separador contenido en los datos de origen. En nuestro caso, se trata de espacios, por lo que se deberá activar la casilla correspondiente:



Después, aparece una penúltima pantalla, donde se deben dejar tal como están los parámetros por defecto.



Finalmente, en la última pantalla se debe seleccionar dónde se desea importar los datos, en la hoja de cálculo existente o en una nueva hoja. Clicando en Aceptar se finaliza el proceso de importación y aparecen los datos ordenados por columnas dentro de la hoja de cálculo.



Se obtiene la siguiente tabla de resultados, donde se muestra, para cada operación de cada pieza, la máquina a la que se debe asignar, así como los tiempos de inicio y final. Los tiempos se obtienen en tiempo desde el instante cero de programación (horas acumuladas) y en tiempo real (día, hora y minuto en la que se debe empezar o acabar una operación). En la última fila se muestra el valor del retraso medio, en horas:

Op	Pieza	Máq	Inicio (h_{acum})	Final (h_{acum})	Inicio			Final		
					Día	H	Minuto	Día	H	Minuto
1	13	Y1	0	6,58	Lunes	6h	0'	Lunes	12h	35'
1	23	I1	0	1,75	Lunes	6h	0'	Lunes	7h	45'
1	2	A	0	1,33	Lunes	6h	0'	Lunes	7h	20
1	11	Y2	0	10,98	Lunes	6h	0'	Lunes	16h	59'
1	3	V1	0	1,93	Lunes	6h	0'	Lunes	7h	55,8'
1	15	R	0	6,83	Lunes	6h	0'	Lunes	12h	50'
1	19	E	0	5,58	Lunes	6h	0'	Lunes	11h	34,8'
1	17	B	0	5,50	Lunes	6h	0'	Lunes	11h	30'
2	2	V2	0,04	2,37	Lunes	6h	2,5'	Lunes	8h	22,5'
2	23	Z1	0,05	6,38	Lunes	6h	3,12'	Lunes	12h	23,12'
2	3	V3	0,06	1,99	Lunes	6h	3,8'	Lunes	7h	59,6'
1	6	A	1,33	4,77	Lunes	7h	20'	Lunes	10h	46,25'
2	6	V4	1,38	10,74	Lunes	7h	22,75'	Lunes	16h	44,18'
3	6	V1	2,50	10,81	Lunes	8h	30'	Lunes	16h	48,75'
4	6	V2	4,03	10,88	Lunes	10h	2’	Lunes	16h	53’
5	6	V3	4,10	19,72	Lunes	10h	5,8'	Martes	9h	43,3'
1	5	A	4,77	7,44	Lunes	10h	46,25'	Lunes	13h	26,25'
2	19	B	5,50	11,59	Lunes	11h	30'	Lunes	17h	35,32'
2	15	E	5,58	10,95	Lunes	11h	34,8'	Lunes	16h	57,3'
3	23	Y1	6,58	12,92	Lunes	12h	35'	Lunes	18h	55'
3	19	R	6,83	12,92	Lunes	12h	50'	Lunes	18h	55,32'
1	14	A	7,44	11,94	Lunes	13h	26,25'	Lunes	17h	56,25'
6	6	V4	10,74	26,36	Lunes	16h	44,18'	Martes	16h	21,68'
3	15	E	10,95	16,33	Lunes	16h	57,3'	Martes	6h	19,8'
1	21	Y2	10,98	40,57	Lunes	16h	59'	Miércoles	14h	34’
4	19	B	11,59	17,68	Lunes	17h	35,32'	Martes	7h	40,65'
5	19	X1	11,65	44,70	Lunes	17h	39,32'	Miércoles	18h	42,18'
1	8	A	11,94	13,94	Lunes	17h	56,25'	Lunes	19h	56,26'
2	8	I1	11,97	14,61	Lunes	17h	58,50'	Lunes	20h	36,67'
3	8	X2	12,01	30,35	Lunes	18h	0,91'	Martes	20h	20,91'
1	10	Y1	12,92	23,25	Lunes	18h	55'	Martes	13h	15'
2	17	R	12,92	19,92	Lunes	18h	55,3257'	Martes	9h	55,32'
1	7	A	13,94	23,27	Lunes	19h	56,26'	Martes	13h	16,26'
2	7	V1	14,00	27,50	Lunes	19h	59,76'	Martes	17h	29,76'
3	7	V2	14,07	27,57	Lunes	20h	4,01'	Martes	17h	34,01'
4	15	E	16,33	21,70	Martes	6h	19,80'	Martes	11h	42,30'
1	22	B	17,68	19,14	Martes	7h	40,65'	Martes	9h	8,34'
3	17	B	19,14	26,14	Martes	9h	8,34'	Martes	16h	8,34'

4	7	V3	19,72	33,22	Martes	9h	43,30'	Miércoles	7h	13,30'
2	22	R	19,92	21,38	Martes	9h	55,32'	Martes	11h	23,02'
3	22	Z1	19,98	30,31	Martes	9h	58,71'	Martes	20h	18,71'
4	17	R	21,38	26,88	Martes	11h	23,02'	Martes	16h	53,02'
1	26	E	21,70	25,83	Martes	11h	42,30'	Martes	15h	49,80'
1	9	Y1	23,25	33,58	Martes	13h	15'	Miércoles	7h	35,00'
1	12	A	23,27	27,65	Martes	13h	16,26'	Martes	17h	38,78'
1	16	E	25,83	29,83	Martes	15h	49,80'	Martes	19h	49,80'
1	25	B	26,14	27,60	Martes	16h	8,34'	Martes	17h	36,04'
5	7	V4	26,36	39,86	Martes	16h	21,68'	Miércoles	13h	51,68'
2	25	R	26,88	28,34	Martes	16h	53,02'	Martes	18h	20,71'
3	25	B	27,60	29,06	Martes	17h	36,04'	Martes	19h	3,73'
1	18	A	27,65	28,89	Martes	17h	38,78'	Martes	18h	53,06'
4	25	R	28,35	29,81	Martes	18h	20,71'	Martes	19h	48,40'
2	5	B	29,06	40,06	Martes	19h	3,73'	Miércoles	14h	3,73'
3	5	V1	29,17	40,17	Martes	19h	10,23'	Miércoles	14h	10,23'
4	5	V2	29,28	40,28	Martes	19h	16,73'	Miércoles	14h	16,73'
2	14	R	29,81	40,14	Martes	19h	48,40'	Miércoles	14h	8,40'
2	26	E	29,83	33,95	Martes	19h	49,80'	Miércoles	7h	57,30'
5	15	X2	30,35	39,43	Martes	20h	20,91'	Miércoles	13h	25,91'
5	5	V3	33,22	44,22	Miércoles	7h	13,30'	Miércoles	18h	13,30'
1	20	Y1	33,58	41,92	Miércoles	7h	35,00'	Miércoles	15h	55,00'
2	16	E	33,95	38,70	Miércoles	7h	57,30'	Miércoles	12h	42,30'
3	26	E	38,70	42,83	Miércoles	12h	42,30'	Miércoles	16h	49,80'
5	25	X2	39,43	41,76	Miércoles	13h	25,91'	Miércoles	15h	45,91'
1	27	B	40,06	45,23	Miércoles	14h	3,73'	Miércoles	19h	13,73'
3	14	R	40,14	45,81	Miércoles	14h	8,40'	Miércoles	19h	48,40'
4	26	X2	41,76	50,43	Miércoles	15h	45,91'	Jueves	8h	25,91'
2	27	E	42,83	48,83	Miércoles	16h	49,80'	Jueves	6h	49,80'
6	7	X1	44,70	70,04	Miércoles	18h	42,18'	Viernes	12h	2,18'
4	14	B	45,23	52,45	Miércoles	19h	13,73'	Jueves	10h	27,05'
5	14	R	45,81	53,03	Miércoles	19h	48,40'	Jueves	11h	1,73'
3	27	E	48,83	54,83	Jueves	6h	49,80'	Jueves	12h	49,80'
4	27	Z1	48,94	82,60	Jueves	6h	56,30'	Sábado	8h	36,30'
6	14	X2	50,43	78,76	Jueves	8h	25,91'	Viernes	20h	45,91'
3	16	B	52,45	58,45	Jueves	10h	27,05'	Jueves	16h	27,05'
1	28	R	53,03	72,23	Jueves	11h	1,73'	Viernes	14h	13,73'
2	28	B	58,45	77,65	Jueves	16h	27,05'	Viernes	19h	39,05'
4	16	X1	70,04	72,87	Viernes	12h	2,18'	Viernes	14h	52,18'
3	28	R	72,23	91,43	Viernes	14h	13,73'	Sábado	17h	25,73'
4	28	E	72,41	91,61	Viernes	14h	24,73'	Sábado	17h	36,73'
1	24	B	77,65	79,27	Viernes	19h	39,05'	Viernes	21h	15,98'
2	24	B	79,27	80,88	Viernes	21h	15,98'	Sábado	6h	52,90'
3	24	B	80,88	82,50	Sábado	6h	52,90'	Sábado	8h	29,83'
4	24	B	82,50	84,11	Sábado	8h	29,83'	Sábado	10h	6,75'
5	24	X1	82,55	84,22	Sábado	8h	33,21'	Sábado	10h	13,21'
5	28	E	91,61	110,81	Sábado	17h	36,73'	Lunes2	20h	48,73'
7,15										

Tabla 5.16. Importación a Excel de la secuencia obtenida

6. Evaluación de los resultados

6.1. Comparación Algoritmo propuesto vs. Método empresa

En este capítulo se describen las pruebas que se han realizado para comparar la eficiencia del procedimiento de resolución diseñado respecto al método manual de programación de operaciones utilizado en la empresa. Para poder validarlo se han realizado pruebas a partir de 10 ejemplares correspondientes a pedidos proporcionados por la planta. El listado completo de los datos iniciales de los ejemplares de prueba está recogido en el **Anexo C.1**.

Cada ejemplar se ha resuelto empleando primero el algoritmo diseñado (mediante el método directo predeterminado), obteniendo las fechas de finalización de cada pieza, y comparándolas posteriormente con las obtenidas mediante la secuenciación proporcionada por la planta. La comparación se ha realizado determinando el porcentaje de discrepancia entre el valor del retraso medio de ambos métodos. Se calcula como la diferencia entre el valor proporcionado por el algoritmo y el valor obtenido en la empresa, se divide por el valor de la empresa y se multiplica por 100. Si la discrepancia es un valor mayor que cero significa que el valor obtenido mediante el algoritmo es mejor (menor retraso medio) que el proporcionado por la empresa; si es menor que cero indica mejores valores de retraso medio en el método utilizado en la empresa. Finalmente, si se obtiene discrepancia cero significa que se han obtenido idénticos resultados en ambos métodos.

La **Tabla 6.1** representa el resultado de la discrepancia obtenido al realizar la comparación entre los métodos algoritmo y empresa en el primer ejemplar considerado (la tabla de discrepancias completa del resto de ejemplares se muestra en el **Anexo C.1**).

Por columnas, se indican las piezas ordenadas por número de pieza, la cantidad requerida en el pedido, el retraso en % de unidades, los días de retraso y la fecha de entrega (en tiempo absoluto, es decir, en horas acumuladas desde el instante de inicio de programación; si se trata de una pieza en retraso la fecha de entrega es cero). También se muestra la fecha de finalización de la pieza en horas acumuladas desde el instante cero de programación y el retraso medio en horas respecto a la fecha de entrega. Finalmente, se ha calculado el retraso medio y el valor de discrepancia individual entre los valores de retraso medio de ambos métodos. En el caso de este primer ejemplar el algoritmo propuesto mejora en un 11,39% la solución obtenida en la empresa, además de obtenerse en un tiempo de ejecución mucho más rápido (décimas de segundo frente a las cinco horas que se invierten en la empresa para realizar la programación).

En la **Tabla 6.2** se muestra el orden de secuenciación obtenido en este primer ejemplar, según los métodos algoritmo y empresa. Las secuenciaciones del resto de ejemplares se han recogido en el **Anexo C.2**.

Resultados Prueba 1									
Pieza	Cantidad pedida	%Uds retraso	Días retraso	Fecha entrega	Demanda media	Resultado Algoritmo		Resultado Empresa	
						Final (h_{acum})	Retraso	Final (h_{acum})	Retraso
1	0	0	0	0	500	0	0	0	0
2	600	40	1	0	600	2,37	2,38	13,61	13,61
3	465	27,5	1	0	500	1,99	1,99	11,20	11,20
4	0	0	0	0	400	0	0	0	0
5	2.000	0	0	128	1.800	44,22	0	35,97	0
6	2.925	32,5	2	0	2.500	26	26,36	24,97	0
7	5.000	0	0	176	4.500	70,04	0	49,47	0
8	3.600	0	0	160	3.500	30,35	0	72,16	0
9	200	0	0	128	1.000	33,58	0	33,58	0
10	200	0	0	128	200	23,25	0	23,25	0
11	426	35,2	1	0	400	10,98	10,98	10,98	10,98
12	24.400	0	0	176	25.000	27,65	0	28,85	0
13	250	100	3	0	200	6,58	6,58	6,58	6,58
14	2.800	0	0	144	2.500	78,76	0	65,53	0
15	1.750	19,44	1	0	1.500	39,43	39,43	35,15	35,15
16	750	0	0	144	1.000	72,87	0	74,99	0
17	900	0	0	112	800	26,88	0	12,5	0
18	2.000	0	0	176	1.800	28,88	0	24,48	0
19	2.290	8,75	1	0	2.300	44,70	44,70	53,83	53,83
20	640	0	0	128	500	41,92	0	41,91	0
21	23.400	4	2	0	2.000	40,57	40,57	40,57	40,57
22	300	0	0	112	275	30,31	0	17,29	0
23	600	100	2	0	750	12,92	12,92	12,92	12,92
24	400	0	0	160	400	84,22	0	98,41	0
25	300	0	0	112	350	41,77	0	14,83	0
26	1.250	0	0	144	1.000	50,43	0	74,19	0
27	1.000	0	0	144	800	80,00	0	91,21	0
28	1.820	0	0	160	1.500	110,81	0	111,12	0
Retraso medio (h):							**7,15**		**8,07**
Discrepancia (%):									**11,39**

Tabla 6.1. Retrasos medios y % de discrepancia para la Prueba 1

Secuenciación obtenida Prueba 1	
Método Algoritmo	**Método Empresa**
13	13
23	23
2	11
11	6
3	2
15	17
19	3
17	22
6	25
5	19
14	21
21	15
8	5
10	14
7	8
22	10
26	7
9	26
12	9
16	20
25	16
18	18
20	12
27	27
28	28
24	24
1	1
4	4

Tabla 6.2. Secuenciación obtenida para la Prueba 1

Con el propósito de visualizar los resultados de las programaciones resultantes, se han representado los diagramas de Gantt para las soluciones obtenidas mediante los métodos algoritmo y de empresa (**Figuras 6.4 y 6.5**). El eje vertical corresponde a las máquinas, y el horizontal a los tiempos de programación, expresados en hora de cada jornada (de 0 a 16)

y en horas acumuladas desde el instante inicial de programación. Las operaciones se han diferenciado por colores.

Los diagramas de Gantt del resto de pruebas pueden consultarse en el **Anexo C.3**.

A continuación se muestra un resumen con los resultados obtenidos en las diez pruebas realizadas, destacando los valores de retraso medio de cada método y las discrepancias (mejoras) correspondientes a cada caso.

Nº Prueba	Retraso medio		% Discrepancia
	Algoritmo (método directo)	Empresa	
1	7,15	8,07	11,39
2	5,15	5,88	12,31
3	10,98	12,24	10,31
4	7,27	8,37	13,12
5	14,46	16,82	14,02
6	10,62	13,50	21,30
7	16,06	18,50	13,21
8	9,12	12,97	29,68
9	18,63	22,56	17,40
10	10,61	14,63	27,51
Discrepancia media (%):			**17,03**

Tabla 6.3. Resumen de retrasos y discrepancia media obtenida respecto al método directo

Para poder extraer una conclusión sobre la mejora promedio obtenida se ha calculado el porcentaje de discrepancia media, realizando el sumatorio de las discrepancias individuales dividida por el número de ejemplares. Se obtiene un valor de mejora media del 17,03%, entendiendo este porcentaje como el % de mejora en la solución obtenida mediante el algoritmo creado respecto al método utilizado en la planta; de esta forma, y de acuerdo con la empresa, se concluye que la programación que realiza el algoritmo propuesto mejora sensiblemente la obtenida mediante el método intuitivo y manual de la empresa.

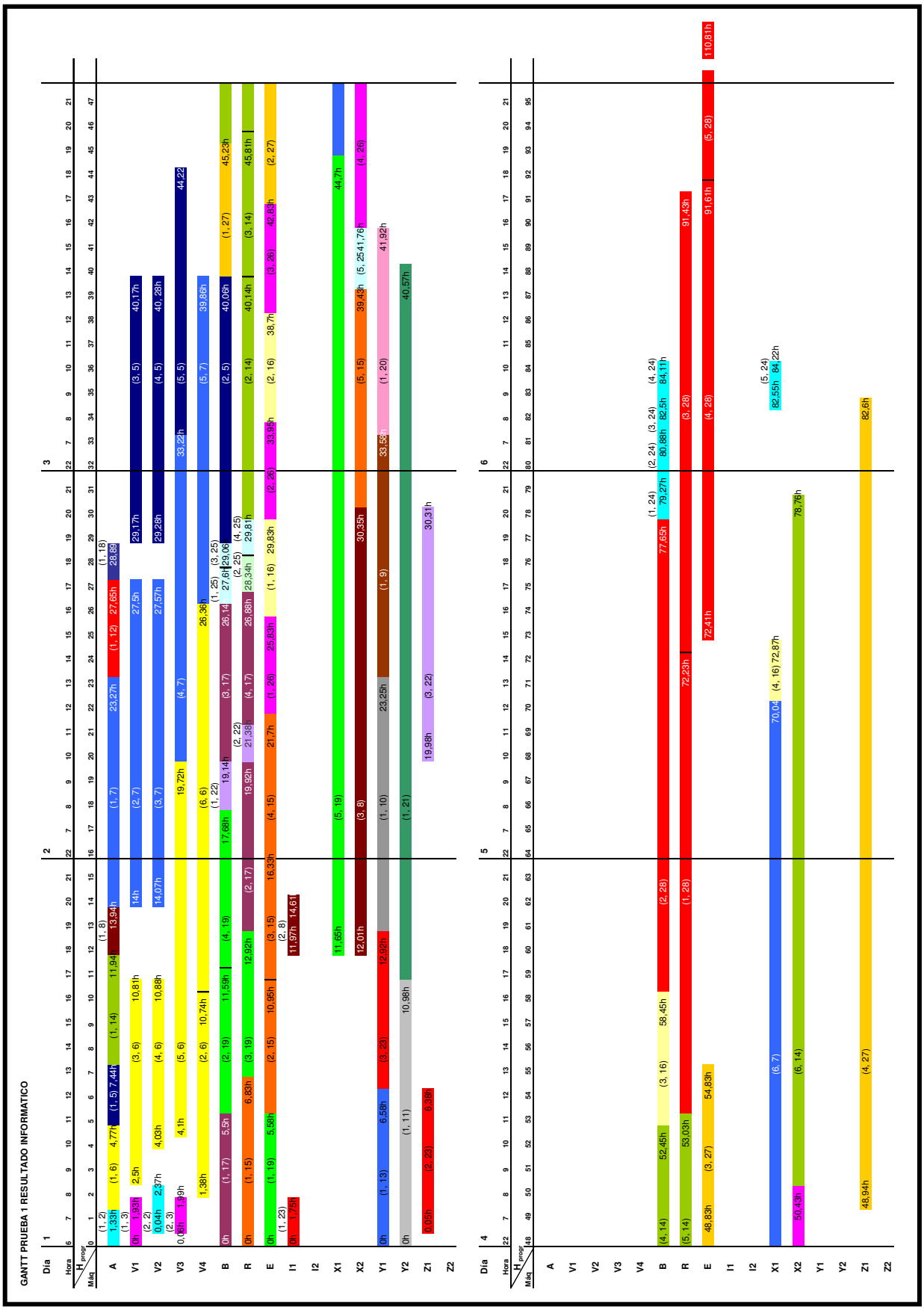


Fig. 6.4. Programación de operaciones para el ejemplar 1, según el algoritmo propuesto

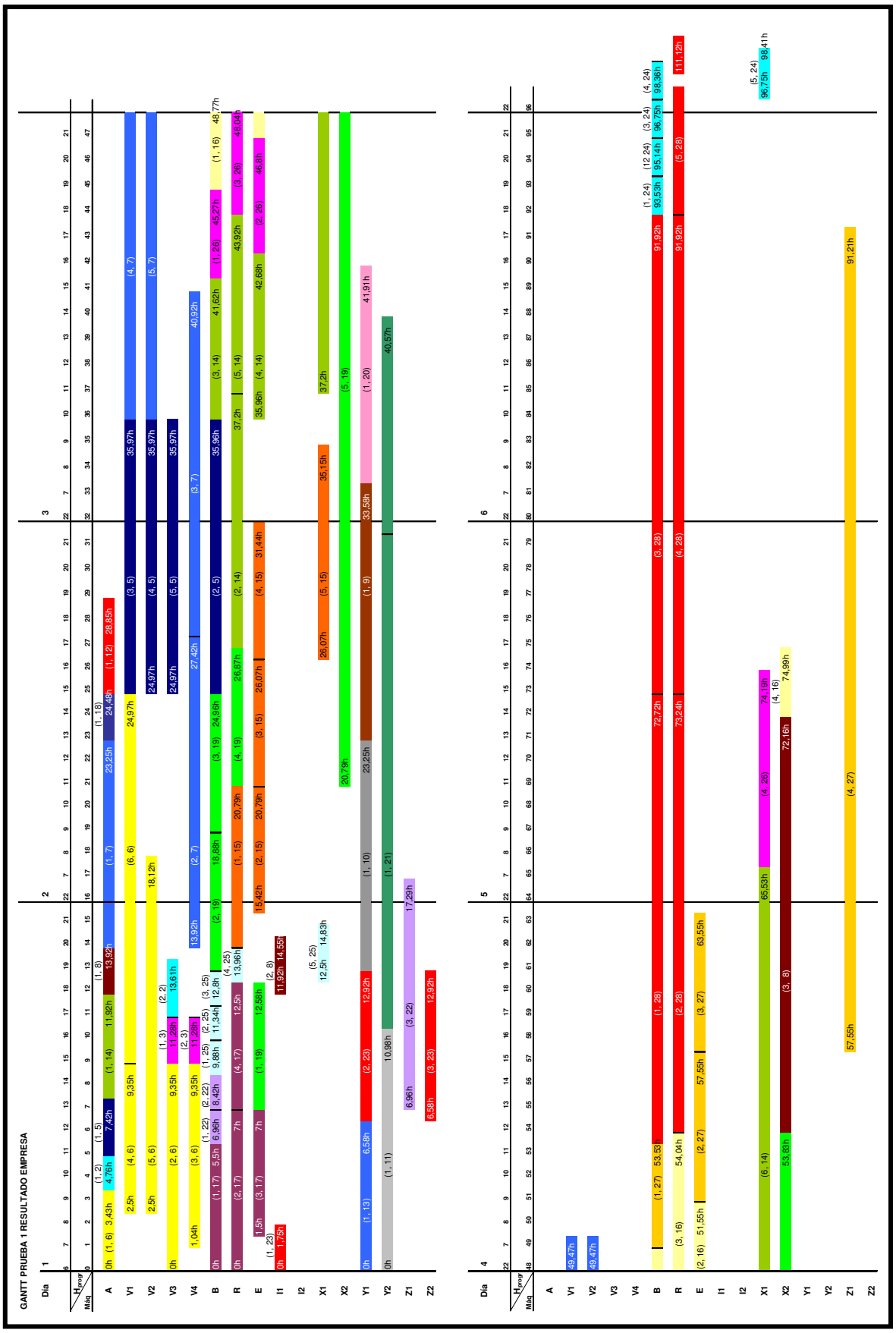


Fig. 6.5. Programación de operaciones para el ejemplar 1, según el método de la empresa

Comparando los resultados obtenidos en las 10 pruebas realizadas, se observan diferencias entre ambos métodos de secuenciación. Desde el punto de vista de la minimización del retraso medio, destacan varias mejoras en el método propuesto respecto al de la empresa:

- La ventaja más destacable procede de la utilización del algoritmo de secuenciación diseñado especialmente para las necesidades de la planta. En la planta no se disponía de ningún procedimiento formalizado; la programación de operaciones se realizaba de forma intuitiva e improvisada, siguiendo ciertas reglas de prioridad de sentido común pero sin un criterio bien definido. Al formalizar el proceso de toma de decisiones en la secuenciación, se dispone de una mayor capacidad de previsión, al obtener un *schedule* (programa) completo con el orden de operaciones previsto, la asignación a las máquinas y sus instantes de inicio y final de fabricación teóricos. Esto permite, a su vez, una mayor capacidad de reacción ante la aparición de sucesos inesperados (averías, pedidos de última hora), puesto que es posible reprogramar nuevamente de forma fácil y rápida desde el punto en el que se ha producido la perturbación.

- Con el objetivo de minimizar el retraso medio, en el diseño del algoritmo se ha priorizado en primer lugar las piezas con mayores unidades en retraso respecto el pedido anterior (en la empresa se programaban primero las piezas con más días de retraso, sin tener en cuenta la cantidad en retraso). La aplicación de esta regla adelanta la programación de las piezas con mayores cantidades pendientes a producir, ya que previsiblemente son más propensas a incurrir en mayores retrasos que las piezas de las que queda menor cantidad por entregar, aunque lleven mayor número de días en retraso. En el ejemplar de la Prueba 1 se observa un ejemplo: en el procedimiento diseñado la pieza 2 se programa antes que la 6, puesto que tiene más unidades en retraso, a pesar de que la pieza 6 lleva más días en retraso. (Ver **Tablas 6.1, 6.2 y 6.3**).

- Otra mejora para la obtención de un menor retraso medio consiste en programar primero las piezas con días de retraso respecto al pedido anterior. Obsérvese, por ejemplo, la pieza 15 del ejemplar de la Prueba 1. Es una pieza que no se entregó a tiempo en el pedido anterior, en concreto lleva un día de retraso en la entrega y falta el 19,44% del pedido por entregar (ver **Tabla 6.1**). En la realidad de la planta las piezas en retraso se acumulan al pedido actual y se entregan en la fecha de entrega del pedido actual. Este hecho no tiene mayor importancia si no se trata de una pieza crítica para el cliente. Sin embargo, si dicha pieza se convierte en crítica de forma inesperada y todavía no se ha fabricado, las penalizaciones son considerables. Por este motivo, con tal de priorizar la fabricación de dichas piezas,

el algoritmo propuesto asigna cero (máxima prioridad) como fecha de entrega, de forma que se programan lo antes posible y disminuye el retraso medio. En efecto, la pieza 15 de la Prueba 1 se fabrica antes en el método propuesto que en el de la empresa (ver secuenciación en la **Tabla 6.2** y diagramas de Gantt en las **Figuras 6.4 y 6.5**).

- La programación, en el procedimiento propuesto, se realiza eligiendo la operación más prioritaria entre todas las operaciones candidatas de entre todas las piezas restantes. En la empresa, sin embargo, se programa de otro modo: se elige la pieza más prioritaria y se programan todas sus operaciones. De este modo, en el método de la empresa las operaciones correspondientes a una misma pieza suelen fabricarse en paralelo, pero esta agrupación de operaciones puede deteriorar el objetivo buscado, puesto que en un instante determinado de programación no se evalúan todas las operaciones posibles. Un ejemplo lo constituye la pieza 17 del ejemplar de la Prueba 1. En el método de la empresa, se han agrupado en lo posible todas las operaciones de la pieza 17, con el propósito de trabajar en paralelo (ver **Figura 6.5**). Este criterio produce un *schedule* más intuitivo, pero a riesgo de obtener un peor resultado en el retraso medio. El método propuesto evita que esto ocurra programando las operaciones una a una, intercalando si es necesario operaciones de piezas diferentes. Así, la primera operación de la pieza 17 se programa en la máquina B. Pero la segunda deberá esperar a que se procesen antes las operaciones de las piezas 15 y 19, que gozan de mayor prioridad (ver **Figura 6.4**). De este modo se mejora el valor del retraso medio.

- En el método de la empresa, el efecto de programar todas las operaciones seguidas de una misma pieza provoca tiempos muertos en las máquinas que podrían estar fabricando alguna operación. Esto no ocurre en el método propuesto, puesto que se prioriza en cada instante la operación más prioritaria en la primera máquina disponible (los huecos ociosos se producen simplemente porque no hay operaciones disponibles). Un ejemplo lo muestra claramente el diagrama de Gantt obtenido en el ejemplar de la Prueba 4. En el método manual se observan huecos ociosos en la máquina E, entre las operaciones (4, 15) y (4, 14); o entre la (4, 17) y la (3,14) (ver diagramas de Gantt en **Anexo C.3**). En el diagrama de Gantt del método propuesto se aprecia un mejor aprovechamiento de las máquinas, al no existir estos tiempos de inactividad provocados por la programación agrupada de operaciones de una misma pieza, lo cual constituye una mejora significativa respecto al método utilizado por la empresa.

Además de las ventajas destacadas en este apartado, se obtienen otros múltiples beneficios adicionales, los cuales se muestran en el **Capítulo 7**.

6.2. Análisis de los tiempos de ejecución

Los tiempos computacionales resultantes de la ejecución del método propuesto resultan ser mínimos en el entorno empresarial donde se aplica: para el procedimiento directo (una sola repetición) se emplean 0,20 segundos. Para el caso aleatorio, se emplea 34,98 segundos para un valor de 1000 repeticiones. En cualquier caso, el cálculo informatizado supone un ahorro enorme de tiempo respecto al método empleado en la empresa, en el que se invierten 5 horas de trabajo para la programación.

La **Tabla 6.6.** resume los tiempos medios de ejecución registrados para cada caso.

Método	Tiempo de ejecución (s)	Nº Repeticiones
Algoritmo, método directo	0,20	1
	3,55	1000
Algoritmo, método aleatorio	0,23	1
	3,63	100
	7,12	200
	10,68	300
	14,28	400
	17,59	500
	20,57	600
	24,52	700
	27,98	800
	31,66	900
	34,98	1.000
Método de la empresa	18.000	1

Tabla 6.6. Tiempo de ejecución medio consumido en cada método

6.3. Comparación de los métodos directo y aleatorio

Se ha realizado una batería de tests de prueba, partiendo de las 10 configuraciones reales presentadas en este capítulo, para determinar con qué frecuencia el método aleatorio

mejora el valor del retraso medio obtenido mediante el método directo. Para ello se han empleado distintas ponderaciones de probabilidad de aplicación de los criterios de prioridad, y salvo alguna excepción, se ha obtenido que el método aleatorio mejora la solución inicial proporcionada por el método directo.

Lo que tiene sentido es calcular la secuencia según el procedimiento directo y a continuación compararla con el resultado del procedimiento aleatorio, generando, por ejemplo, unas 1000 repeticiones, para ver si se obtiene alguna mejora. Si es así, se puede observar si hay una única secuencia o varias que corresponden al menor valor de retraso medio; si sólo hay una, se toma dicha secuencia como solución; si hay varias, entonces se puede elegir aquella que tenga un menor tiempo total de realización (*makespan*).

6.3.1. Ponderación de probabilidades del 60, 20, 10, 5 y 5%

En los 10 conjuntos de datos iniciales diferentes, se generaron las secuencias según el método directo y el aleatorio, para un valor de 1000 repeticiones y una ponderación de probabilidades de las reglas de prioridad del 60, 20, 10, 5, y 5%. Y se obtuvieron los resultados que muestra la **Tabla 6.7**:

Prueba	Método Directo		Método Aleatorio 60, 20, 10, 5, 5%		% Mejora
	Tiempo computacional (s)	Retraso medio (h)	Tiempo computacional (s)	Retraso medio (h)	
1	3,56	7,15	32,73	6,95	2,79
2	3,64	5,15	31,02	4,91	4,66
3	3,55	10,98	31,81	10,69	2,64
4	3,49	7,27	29,15	6,62	8,94
5	3,53	14,46	31,10	11,41	21,09
6	3,55	10,62	31,36	10,63	-0,09
7	3,52	16,06	31,52	14,91	7,16
8	3,58	9,12	29,90	7,98	12,50
9	3,58	18,63	30,34	16,55	11,16
10	3,53	10,61	32,27	10,61	0

Tabla 6.7. Retraso medio, en horas, y % mejora del método aleatorio respecto al directo, pesos de probabilidad del 60, 20, 10, 5, 5% y 1000 repeticiones.

De las 10 pruebas realizadas, en 8 ocasiones se mejora el resultado directo, en 1 se empeora ligeramente y en 1 se mantiene igual, por lo que en el 80% de los casos se obtiene una mejora. Por otro lado, se ha calculado el porcentaje de mejora del resultado

aleatorio respecto al directo, dividiendo la diferencia de resultados respecto al resultado inicial del método directo. El valor del porcentaje medio de mejora en la solución obtenida mediante el método aleatorio respecto el método directo es del 7,08%, por lo que se destaca la conveniencia de también ejecutar el programa en modo aleatorio para tratar de mejorar la solución inicial proporcionada por el método directo.

6.3.2. Ponderación de probabilidades del 5, 5, 10, 20 y 60%

Por otro lado, se puede pensar que una mayor relajación en la jerarquía de los órdenes de prioridad puede conducir a peores resultados en el caso aleatorio respecto al método directo, puesto que las reglas que priorizan la urgencia en las fechas de entrega y los retrasos acumulados pierden importancia al disminuir la ponderación de probabilidad. Sin embargo los resultados obtenidos no avalan esta teoría.

Para establecer una comparación se realizó de nuevo otra batería de tests de prueba con las configuraciones de datos iniciales, pero esta vez se invirtieron los pesos de probabilidad, asignando el 5, 5, 10, 20 y 60% a las reglas de prioridad. El número de repeticiones se fijó en 1000. Se realizaron 10 pruebas y se observó que en el 70% de los casos mejoró la función objetivo, por lo que no resulta determinante asignar los mayores pesos de probabilidad a las primeras reglas de prioridad a efectos de mejorar la función objetivo.

Los resultados se muestran en la **Tabla 6.9**.

Prueba	Método Directo		Método Aleatorio 5, 5, 10, 20, 60%		% Mejora
	Tiempo computacional (s)	Retraso medio (h)	Tiempo computacional (s)	Retraso medio (h)	
1	3,56	7,15	29,94	7,36	-2,93
2	3,64	5,15	30,32	4,95	3,88
3	3,55	10,98	30,22	10,84	1,27
4	3,49	7,27	30,00	6,62	8,94
5	3,53	14,46	30,05	11,49	20,54
6	3,55	10,62	30,18	10,63	-0,09
7	3,52	16,06	30,23	15,83	1,43
8	3,58	9,12	29,99	8,16	10,53
9	3,58	18,63	30,06	16,49	11,49
10	3,53	10,61	29,94	10,95	-3,20

Tabla 6.8. Retraso medio, en horas, y % mejora del método aleatorio respecto al directo, pesos de probabilidad del 5, 5, 10, 20, 60% y 1000 repeticiones.

De la tabla anterior se extrae que en 7 ocasiones se mejora el resultado inicial directo y en 3 se empeora. El porcentaje de mejora media es del 5,18%.

6.3.3. Ponderación de probabilidades del 20, 20, 20, 20 y 20%

Finalmente, se ha realizado la comparación entre ambos métodos para una distribución de probabilidades del 20, 20, 20, 20 y 20%. Los resultados obtenidos son los siguientes:

Prueba	Método Directo		Método Aleatorio 20, 20, 20, 20, 20%		% Mejora
	Tiempo computacional (s)	Retraso medio (h)	Tiempo computacional (s)	Retraso medio (h)	
1	3,56	7,15	31,72	6,95	2,79
2	3,64	5,15	31,33	4,91	4,66
3	3,55	10,98	31,70	10,69	2,64
4	3,49	7,27	30,34	6,63	8,80
5	3,53	14,46	30,92	11,41	21,09
6	3,55	10,62	31,56	10,63	-0,09
7	3,52	16,06	31,51	14,91	7,16
8	3,58	9,12	30,14	8,43	7,56
9	3,58	18,63	30,16	16,49	11,48
10	3,53	10,61	30,20	10,61	0

Tabla 6.9 Retraso medio, en horas, y % mejora del método aleatorio respecto al directo, pesos de probabilidad del 20, 20, 20, 20, 20% y 1000 repeticiones.

De nuevo se concluye que el método aleatorio mejora en la mayoría de los casos la solución obtenida por el método inicial directo, con un promedio de mejora del 6,61%.

6.4. Consideraciones finales

Tal como se ha observado en el apartado anterior, se obtiene una mejora en el resultado del retraso medio al utilizar el método aleatorio. En la tabla **6.12** se muestra los resultados obtenidos al aplicar el procedimiento directo y las 3 configuraciones aleatorias propuestas.

	Directo		Aleatorio 60, 20, 10, 5, 5%			Aleatorio 5, 5, 10, 20, 60%			Aleatorio 20, 20, 20, 20, 20%		
Prueba	T (s)	Retraso medio (h)	T (s)	Retraso medio (h)	% Mejora	T (s)	Retraso medio (h)	% Mejora	T (s)	Retraso medio (h)	% Mejora
1	3,56	7,15	32,73	6,95	2,79	29,94	7,36	-2,93	31,72	6,95	2,79
2	3,64	5,15	31,02	4,91	4,66	30,32	4,95	3,88	31,33	4,91	4,66
3	3,55	10,98	31,81	10,69	2,64	30,22	10,84	1,27	31,7	10,69	2,64
4	3,49	7,27	29,15	6,62	8,94	30	6,62	8,94	30,34	6,63	8,8
5	3,53	14,46	31,1	11,41	21,09	30,05	11,49	20,54	30,92	11,41	21,09
6	3,55	10,62	31,36	10,63	-0,09	30,18	10,63	-0,09	31,56	10,63	-0,09
7	3,52	16,06	31,52	14,91	7,16	30,23	15,83	1,43	31,51	14,91	7,16
8	3,58	9,12	29,9	7,98	12,5	29,99	8,16	10,53	30,14	8,43	7,56
9	3,58	18,63	30,34	16,55	11,16	30,06	16,49	11,49	30,16	16,49	11,48
10	3,53	10,61	32,27	10,61	0	29,94	10,95	-3,2	30,2	10,61	0
% Mejora del valor de retraso medio respecto al método directo					7,08			5,18			6,61

Tabla 6.10 Resumen del % de mejora obtenido en el valor del retraso medio de las tres configuraciones del método aleatorio respecto al método directo.

En la tabla anterior se observa que la configuración más favorable en cuanto al retraso medio obtenido es el método aleatorio con ponderación de probabilidades del 60, 20, 10, 5 y 5%.

Esta mejora observada al emplear el método aleatorio aumenta en consecuencia el % de mejora final (de 17,03 a 22,93%) respecto al método manual utilizado en la empresa:

Nº Prueba	Retraso medio		% Discrepancia
	Algoritmo (método aleatorio)	Empresa	
1	6,95	8,07	13,88
2	4,91	5,88	16,50
3	10,69	12,24	12,66
4	6,62	8,37	20,91
5	11,41	16,82	32,16
6	10,63	13,50	21,26
7	14,91	18,50	19,41
8	7,98	12,97	38,47
9	16,55	22,56	26,64
10	10,61	14,63	27,48
Discrepancia media (%):			**22,93**

Tabla 6.11 Resumen de retrasos y discrepancia media obtenido respecto al método aleatorio

A modo de conclusión, dado que los tiempos de ejecución de la aplicación informática son reducidos, se recomienda realizar la programación utilizando el procedimiento propuesto del siguiente modo:

- Realizar la programación según el método directo.
- Ejecutar el método aleatorio con pesos de probabilidad del 60, 20, 10, 5 y 5%.
- Ejecutar el método aleatorio con pesos de probabilidad del 5, 5, 10, 20 y 60%.
- Ejecutar el método aleatorio con pesos de probabilidad del 20, 20, 20, 20 y 20%.
- Comparar los resultados obtenidos.
- Seleccionar la solución más conveniente.

7. Beneficios obtenidos

Del análisis del capítulo anterior y de acuerdo con la empresa se extraen los siguientes beneficios proporcionados por el procedimiento propuesto respecto al método utilizado en la empresa:

- ***Mejora en la competitividad de los plazos de entrega.*** El método propuesto ofrece un mejor valor de la función objetivo. Tal como se observó en el capítulo anterior, el algoritmo propuesto mejora en un 23% los valores de retraso medio respecto a los obtenidos en la empresa.

- ***Ahorro importante en el tiempo de ejecución.*** En la empresa se invierten 5 horas para elaborar un programa de operaciones. Con la aplicación informática, por el contrario, se obtiene un programa en el orden de 90 segundos, también en caso de reprogramación.

- ***Reducción de paradas improductivas.*** El algoritmo creado asigna una operación en cuanto hay una máquina disponible, lo que reduce los tiempos ociosos que se producen en la planta al programar en bloque las operaciones de una misma pieza.

- ***Optimización de recursos productivos.*** Las programaciones obtenidas pueden consultarse para extraer observaciones sobre futuras necesidades de recursos productivos y de mantenimiento preventivo, a partir de la frecuencia de repetición de un determinado pedido, las cargas de las máquinas, etc.

- ***Sencillez de uso.*** La aplicación informática creada es sencilla de utilizar; cualquier persona del departamento de producción puede hacerse cargo de ella en caso necesario, por lo que ya se no depende de la presencia permanente de un planificador experto.

- ***Minimización de la confusión.*** El programa puede reprogramarse desde cualquier punto de su desarrollo en el caso de que se produzca algún imprevisto, obteniéndose un nuevo programa adaptado a la nueva situación. Esto hace que se mantenga el control sobre la planificación y evita la confusión entre los usuarios.

- ***Mejora de la robustez frente a imprevistos.*** En el diseño del procedimiento propuesto se han incluido reglas de robustez para que el programa sea más resistente frente a los sucesos inesperados (averías, pedidos urgentes de última hora), de modo que las modificaciones que se deban hacer tras la reprogramación sean las mínimas posibles.

- ***Mejora de la información para la toma de decisiones del planificador.*** La aplicación permite una gestión anticipada, al dotar el proceso de decisiones de una mayor predictibilidad. Al disponer de un programa bien definido, se obtiene de la información necesaria para poder organizar los recursos productivos de forma más eficiente, anticipar posibles imprevistos y prever las alternativas necesarias.

- ***Mejora de la información al cliente.*** La programación informática permite ofrecer una respuesta clara e inmediata a la pregunta "¿Estará en la fecha pactada?", detectar anticipadamente los retrasos y actuar en consecuencia, incrementando así la satisfacción del cliente, ya que se minimizan los incumplimientos en los compromisos de entrega.

8. Evaluación económica

A continuación se lleva a cabo un análisis económico del coste de ejecución del proyecto, analizando cada una de las fases de las que consta:

- ***Estudio de la situación inicial:*** análisis de las particularidades del caso real, indicando las características que debe contemplar el programa.
- ***Definición del problema:*** búsqueda y estudio de la literatura referente a la secuenciación de máquinas en ambiente *Job Shop* flexible con minimización del retraso medio, especificando el caso que mejor se ajusta a las necesidades.
- ***Diseño:*** desarrollo de la estrategia a seguir para obtener la solución a partir de los datos iniciales, implementación en lenguaje de programación c# y realización de pruebas para comprobar la fiabilidad del programa y corregir errores.
- ***Licencia de software:*** derechos de uso del software desarrollado.

La valoración económica resultante es la siguiente:

Concepto	Horas	Coste €/h	Precio total (€)
1. Estudio de la situación inicial	80	120	960
2. Definición del problema			
2.1. Búsqueda de información	200	60	12.000
2.2. Definición de las especificaciones	50	60	3.000
3. Diseño			
3.1. Diseño del algoritmo	90	120	10.800
3.2. Diseño de la interfaz e implementación	100	60	6.000
3.3. Depuración y Pruebas	150	60	9.000
4. Licencia de software			800
		SUBTOTAL	42.560
		BENEFICIO (20%)	8.512
		IVA (16%)	6.809
		TOTAL	**57.881**

Fig. 8.1. Evaluación económica del proyecto

Para evaluar el valor de retorno de la inversión (ROI) se ha utilizado la fórmula que proporciona este valor calculado en función de la inversión realizada y el beneficio o ahorro que se espera obtener:

$$\textbf{ROI = (beneficios − inversión) / (inversión)} \qquad \text{(Ec. 8.1)}$$

De acuerdo con la empresa, se espera que la implantación del procedimiento creado permita reducir los costes de penalización por impuntualidad en las entregas y conservar en su cartera de pedidos la pieza 19 (cuya permanencia peligra actualmente a causa de los retrasos en su entrega), lo que reportaría un beneficio total previsto de 180.000 euros. Según este dato, se obtiene el siguiente valor del ROI:

$$ROI = (180.000) - (57.881) / 57.881 = 2,11$$

El porcentaje de beneficios obtenidos en el primer año de la inversión es del 211%, es decir, de cada euro invertido se obtiene un beneficio de 2,11 euros, y de acuerdo con la empresa se valida este valor como satisfactorio.

9. Impacto ambiental

La implantación del método propuesto en la planta puede proporcionar beneficios medioambientales en cuanto al ahorro de energía y de costes, entre los que destacan:

Maquinaria:

- Al disponer de un *schedule* (programa) detallado se conocen los tiempos de utilización de las máquinas evitando así las paradas y puestas en marcha innecesarias de las mismas, lo que se traduce en un ahorro de energía eléctrica.

Equipos ofimáticos:

- Configuración de los ordenadores en modo “ahorro de energía”.
- Apagado de los equipos y desconectado del alimentador de corriente una vez finalizada la jornada de trabajo.
- Configuración del protector de pantalla a negro, el único que ahorra energía.
- Utilización de portátiles y monitores de pantalla plana, más eficientes que los ordenadores de mesa y las pantallas convencionales.

Consumo de papel:

- Se ha evitado su uso en lo posible, guardando los documentos en formato digital.
- Se ha utilizado de forma preferente papel reciclado (en su defecto, se ha optado por el papel blanco totalmente libre de cloro), y se ha impreso por las dos caras.

Iluminación:

- Uso de interruptores independientes para iluminar sólo las zonas necesitadas de un mismo área.
- Aprovechamiento de la iluminación natural, organizando los puestos de trabajo de manera que reciban luz natural.
- Apagado de las luces que no son necesarias, incluso para períodos cortos de tiempo.

Transporte:

- Se obtiene un ahorro energético en el transporte, ya que al minimizar los retrasos en las entregas se disminuye también el número de viajes necesarios, reduciendo el consumo de combustible y las emisiones de CO_2.

Residuos:

- Las piezas metálicas defectuosas, así como el resto de residuos generados en el área de producción se gestionan por una entidad autorizada conforme a la normativa vigente.

Conclusiones

Este trabajo tiene su origen en la planta de fabricación de una empresa del sector automoción en la que se requiere encontrar una solución (eliminar, o al menos, minimizar) a los retrasos en las entregas al cliente. Para resolver este caso se ha identificado el ambiente de trabajo como un *Flexible Job-Shop Scheduling Problem (FJSSP)* con el retraso medio como función objetivo. Como método de resolución se ha diseñado una heurística basada en un algoritmo de *dispatching* en el que se han planteado dos procedimientos: por un lado, un método jerárquico directo, fuertemente priorizado; y por otro, un método aleatorio en el que se permite al usuario ponderar probabilísticamente los criterios de prioridad para relajar la jerarquía de los mismos.

En un entorno de programación a corto plazo, como el correspondiente a la planta de estudio, el número de sucesos aleatorios que pueden ocurrir suele ser elevado, por lo que la reprogramación constituye un hecho habitual. Para afrontar el proceso de reprogramación se ha tratado que el programa sea robusto, introduciendo reglas de robustez en el propio procedimiento del algoritmo, de manera que los cambios a introducir después de una perturbación sean los mínimos posibles.

El método de resolución se ha implementado en lenguaje c# para crear una aplicación informática, denominada *Secuenciador*. A partir de los datos de máquinas y piezas iniciales, el programa proporciona un plan de trabajo y el valor del retraso medio obtenido: se obtiene el orden en que se han de realizar las operaciones, a qué máquina se han de asignar y los tiempos de inicio y de final de fabricación.

La evaluación de los resultados obtenidos muestra que el algoritmo mejora sensiblemente el método utilizado en la empresa, aportando muchas otras ventajas, entre las que destacan: gran ahorro en el tiempo de ejecución, reducción de los tiempos muertos de máquina, mejora en la información para la toma de decisiones del planificador y, en definitiva, mejora en la competitividad de los plazos de entrega.

Como recomendación de un futuro desarrollo para este trabajo destaca diseñar una metaheurística (por ejemplo, tipo búsqueda tabú o recocido simulado) que conduzca a mejores resultados a través de la exploración del vecindario de la solución obtenida mediante la heurística inicial.

Bibliografia

Referencias bibliográficas

[1] AIEX, R.M., BINATO S. and RESENDE, M.G.C. (2001). *Parallel GRASP with Path-Relinking for Job Shop Scheduling*. Parallel Computing, 29. pp. 393-430, (2003).

[2] ALLAHVERDI A., CHENG, T.C.E. and KOVALYOV, M.Y. *A Survey of Scheduling Problems with Setup Times or Costs*. European Journal of Operational Research 187, pp. 985-1032, (2008).

[3] AKERS, S.B., FRIEDMAN J., *A Non-Numerical Approach to Production Scheduling Problems*, Operations research, 3 pp. 422-429, (1955).

[4] APPLEGATE, D., and COOK, W., *A computational study of the Job-Shop Scheduling Problem,* ORSA Journal on Computing, Vol.3, Nº2, pp. 149-156. (Spring 1991).

[5] BALAS, E., *Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm,* Operations Research, Vol. 17, No. 6. (Nov. - Dec.), pp. 941-957, (1969).

[6] BELLMAN, R., *An introduction to the theory of dynamic programming*, The RAND Corporation, Report R-245, (1953).

[7] BINATO S., HERY, W.J., LOEWENSTERN, D.M. and RESENDE, M.G.C., *A GRASP for Job Shop Scheduling*. In: Essays and Surveys in Metaheuristics, Ribeiro, Celso C., Hansen, Pierre (Eds.), Kluwer Academic Publishers, (2002).

[8] BRUCKER, P., JURISCH, B. y SIEVERS, B., *A Branch and Bound Algorithm for Job-Shop Scheduling Problem*, Discrete Applied Mathematics, Vol 49, pp. 105-127, (1994).

[9] CARLIER, J., and RINNOOY KAN, A.H.G., *Scheduling subject to nonrenewable-resource constraints*. Operations Research Letters 1, pp. 52–55, (1982).

[10] CERNY, V., *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimization Theory and Applications, Vol. 45, pp. 41-45, (1985).

[11] CHENG, T.C. and SIN, C.C., *A State-of-the-art Review of Parallel-machine Scheduling Research*. European Journal of Operational Research 47, pp.271-292, (1990).

[12] CONWAY, R., MAXWELL, W., MILLER, *Theory Of Scheduling*, Dover Publications, (2003).

[13] COMPANYS, R. *Dirección de Operaciones. Secuenciación: programación de proyectos y de taller. Equilibrado y secuenciación de líneas*. CPDA-ETSEIB, (2003).

[14] DUARTE, A., PANTRIGO, J., GALLEGO, M., Metaheurísticas, URJC – Ciencias Experimentales y tecnología, pp. 35-36, (2007).

[15] FEO, T.A, and RESENDE, M.G.C, *Greedy randomized adaptative search procedures*. Journal of Global Optimization, 6, pp. 109-133, (1995).

[16] GIFFER, B., and THOMPSON, G.L., *Algorithms for Solving Production Scheduling Problems*, Operations Research, Vol.8, pp. 487-503, (1960).

[17] GLOVER, F., *Future Paths for Integer Programming and Links to Artificial Intelligence*, Computers & Operations Research, Vol. 13, Nª. 5, pp. 533-549, (1986).

[18] HOLLAND. H., *Adaptation in Natural and Artificial Systems*. University of Michigan Press Ann Arbor, (1975).

[19] JACKSON, J.R., *An extension of Johnson's Result on Job Lot Scheduling*, Naval Research Logistics Quaterly, 3, pp. 201-203, (1956).

[20] JOHNSON, S.M., *Optimal two and three stage production schedule with setup times included.* Naval Research Logistics Quaterly, 1, pp. 61-67 (1954).

[21] KENNEDY, J., and EBERHART, R.C., *Particle swarm optimization*, IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942-1948. (1995).

[22] KIM, D.-W. KIM, K.-H., JANG, W, and CHEN, F.F., *Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing.* Robotics and Computer Integrated Manufacturing 18, pp. 223-231, (2002).

[23] KIRKPATRICK, S., GELATT, C.D., VECCHI, M.P, *Optimization by Simulated Annealing*, Science, Vol. 220, Nº4598, pp. 671-680, (1983).

[24] LAND A.H., and DOIG, A.G. *An Automatic Method of Solving Discrete Programming Problems.* Econometrica, (1960).

[25] MARTÍ, R., *Procedimientos metaheurísticos en Optimización Combinatoria.* Matemàtiques 1 (1), pp. 3-62 (2003).

[26] MUTH, J.F., and THOMPSON, G.L., *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ., (1963).

[27] OLIVEIRA, J.A.V. *Aplicação de Modelos e Algoritmos de Investigação Operacional ao Planeamento de Operações em Armazéns*, Ph.D. Thesis, Universidade do Minho, Portugal, (2000).

[28] PINEDO, M., *Planning and Scheduling in Manufacturing and Services*. Springer, (2005).

[29] RAVETTI, M.G., MATEUS, G.R., ROCHA P.L. and PARDALOS, P.M. *A Scheduling Problem with Unrelated Parallel Machines and Sequence Dependent Setups*. International Journal of Operational Research 2, pp. 380-399 (2007).

[30] ROY, B, and SUSSMAN, B, *Les problemes d'ordonnancement avec constraintes disjonctives*: SEMA, Note D.S., Nº. 9, Paris, (1964).

[31] WANG, L. and ZHENG, D. *An effective hybrid optimisation strategy for job-shop scheduling problems,* Computers & Operations Research, Vol. 28, pp. 585-596, (2001).

[32] WILLIAMSON, D.P., HALL, L.A., HOOGEVEEN, J.A., HURKENS, C.A.J., LENSTRA, J.K., SEVAST'JANOV, S. and SCHMOYS, D.B., *Short Shop Schedules*, Operations Research, March - April, 45(2), pp. 288-294, (1997).

Bibliografía complementaria

BAUTISTA, J., MATEO, M., RUA, C., IBAÑEZ, J.M., y COMPANYS, R. *Dirección de Operaciones. Transparencias*, CPDA-ETSEIB, (2003).

BRUCKER, P. *Scheduling Algorithms*, Springer, (2007).

BRUCKER, P., and KNUST, S., *Complex Scheduling*, Springer, (2006).

HERRMANN, J.S., *Handbook of Production Scheduling*, Springer, (2006).

JANSEN, K., MASTROLLINI, M., and SOLIS, R. *Aproximation algorithms for flexible job shop problems*. Lecture Notes In Computer Science; Vol. 1776, pp. 68-77, (2000).

MAYO, J. *C# al descubierto*, Prentice Hall, (2002).

KAO, M-Y, *Enciclopaedia of Algoritms*, Springer, (2008).

PINEDO, M., *Scheduling: Theory, Algorithms and Systems*, Springer, (2008).

ZRIBI, N., KACEM, I., ELKAMEL, E., *Hierarchical Optimization for the Flexible Job-Shops Scheduling Problem*. 11th IFAC Symposium on Information Control Problems in Manufacturing (Incom04). Brasil, (2004).