Organización de la producción II

Dirección de operaciones 4

AULA ETSEIB

Escola Tècnica Superior d'Enginyers Industrials de Barcelona

Organización de la producción II

Dirección de operaciones 4

Ramón Companys Pascual
Albert Corominas Subias

La presente obra fue galardonada por la UPC en 1991

EDICIONS UPC
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Primera edición: mayo de 1996

La presente obra fue galardonada en el primer concurso «Ajut a l'elaboració de material docent» convocado por la UPC en 1991.

Diseño de la cubierta: Edicions UPC

Edicions de la Universitat Politècnica de Catalunya, SL
C. Jordi Girona Salgado 31, 08034 Barcelona
Tel. 401 68 83 Fax. 401 58 85
ETSEIB (Escola Tècnica Superior d'Enginyers Industrials de Barcelona). Av. Diagonal 647, 08028 Barcelona

Producción: Servei de Publicacions de la UPC
y CPDA (Centre de Publicacions d'Abast)
Av. Diagonal 647, ETSEIB, 08028 Barcelona

Depósito legal: B-2.688-96
ISBN 84-8301-126-3

Capítulo 6 Programación de actividades

6.1 Conceptos

> "... Ja sabeu el camí d'allà on vaig."
> "Senyor", li diu Tomàs, "no sabem tan sols a on vas, com podríem saber-ne el camí?"
>
> Evangeli segons S. Joan, 14, 4-5
> (en *Evangelis i Fets dels Apòstols*, 3ª ed., Ed. Claret, 1978, p. 264).
>
> "... Would you tell me, please, which way I ought to go from here?"
> "That depends a good deal on where you want to get to", said the Cat.
> "I don't much care where ...", said Alice.
> "Then it doesn't matter which way you go", said the Cat.
> "... so long as I get *somewhere*", Alicé added as an explanation.
> "Oh, you're sure to do that", said the Cat, "if you only walk long enough".
>
> Lewis Carroll, Alice's Adventures in Wonderland
> (en *The Complete Illustrated Works of Lewis Carroll*, Chancellor Press, 1982, pp. 63-64).

Una vez realizados la planificación y el cálculo de necesidades, y prosiguiendo en la adopción de decisiones cada vez a mayor nivel de detalle, debe procederse a la programación de operaciones: cada una de las tareas elementales que conducen a la realización del plan maestro de producción debe asignarse a un centro de trabajo, máquina o instalación y debe concretarse el intervalo temporal de realización.

Programas muy detallados son comunes en diversos sectores industriales, de los cuales los horarios de ferrocarriles son uno de los ejemplos más conocidos y vituperados. Cabe señalar que, si bien en algunos países el cumplimiento de los horarios es sólo aproximado, en otros la existencia de conexiones habituales entre diferentes líneas con márgenes bastante reducidos implica la confianza en que dicho cumplimiento alcanza un elevado grado de exactitud, lo que no supone que excepcionalmente no se produzcan incidencias y por tanto retrasos e incumplimientos. La regla es que el programa se cumpla; la excepción, que con frecuencia reducida se produzcan desviaciones. El programa marca la

pauta a seguir y garantiza la sincronización de las actividades en la medida en que las desviaciones de la realidad respecto al programa sean pequeñas.

La programación, dada su proximidad a la realización y el nivel de concreción perseguido, no puede utilizar valores medios, tan usuales en la planificación. Ello representa una modalidad intrínseca que la diferencia cualitativamente de la planificación, al pasar del universo de los números racionales al de los números enteros. Una de las dificultades en los problemas de programación viene marcada por la combinatoria de las soluciones posibles, imposible de explorar, en la mayoría de los casos, en forma exhaustiva.

Un ejemplo muy simple permitirá aclarar este extremo. Supongamos la programación de 10 piezas en 4 máquinas, y que debe realizarse en cada pieza una operación en cada máquina. En principio el número de programas posibles es:

$$(10!)^4 \approx 1{,}7\text{x}10^{26}$$

aun disponiendo de un potente ordenador capaz de generar y evaluar una solución en 10^{-9} segundos, la consideración de todas las soluciones representaría:

$$5{,}4\text{x}10^9 \text{ años}$$

lo que hace dicha consideración inviable. No es preciso indicar que los problemas industriales son de mucha mayor dimensión que el ejemplo citado.

Otra dificultad añadida es la gran variedad de tipos de problema existentes, tanto por las restricciones a que están sometidos, como por los criterios de evaluación utilizados, lo que impide una presentación completa del tema.

El capítulo se estructura de la siguiente manera: en el apartado 6.1.1 se realizan consideraciones generales, se establece la conexión con el plan de necesidades de capacidad y se estudia la carga de máquinas; en el 6.1.2 se describe la situación antes de la secuenciación.

En el apartado 6.1.3 se presentan los problemas de secuenciación y se define la nomenclatura a utilizar, así como las hipótesis de Conway, Maxwell y Miller, bajo las que van a desarrollarse los apartados siguientes. En el apartado 6.1.4 se estudian las secuencias finitas para una sola máquina, que si bien representan un caso muy particular, sirven para evaluar las dificultades que presentarán otros más complejos.

En el apartado 6.1.5 se analizan los problemas con flujo regular (*flow-shop*) estáticos, y se presentan diversas heurísticas así como un procedimiento exacto para el caso 3 máquinas, el método de Lomnicki, cuyo interés fundamental creemos que estriba en servir

de introducción natural a los procedimientos de exploración y evaluación.

En el apartado 6.1.6 se estudian los problemas con flujo regular dinámicos y en especial el algoritmo de Carlier y la heurística de los trapecios dinámicos.

En el apartado 6.1.7 se consideran los problemas de flujo general estáticos, en los que empezamos a relajar las hipótesis de Conway, Maxwell y Miller. Presentamos unos procedimientos heurísticos que pueden servir de núcleo para procedimientos más complejos de utilización industrial.

En el apartado 6.1.8 estudiamos los problemas de flujo general dinámicos, como prolongación del tema anterior dado que en el caso del flujo general la distinción entre estático y dinámico es poco acusada. Incluimos dos ejemplos concretos relativos a trabajos desarrollados en el Laboratori d'Organització de la Producció (sistema Berenice y sistema Palamedes).

En el apartado 6.1.9 incluimos algunas ideas basadas en los trabajos de E. Goldrath, en particular las informaciones disponibles sobre el programa informático conocido como OPT.

6.1.1 Problemática de la programación

La programación de la producción consiste en asignar las órdenes de producción y/o las operaciones en que se descomponen a centros de trabajo específicos dentro de intervalos temporales concretos; en otras palabras, un programa es una asignación más un calendario. El proceso de programación puede verse como una fase más de preparación de las actividades productivas, después de la planificación y del cálculo de necesidades.

En los sistemas productivos continuos (puros) la problemática se circunscribe casi únicamente a la asignación de operaciones a centros, al establecimiento de la ruta y al consiguiente equilibrado de líneas, puesto que a partir de este punto sólo restará establecer la secuencia de los productos a lanzar a las líneas, si es que por las mismas transitan productos con características muy diferentes.

En los sistemas intermitentes la programación adquiere una carta de naturaleza distinta, y posiblemente más compleja, por cuanto la frecuencia del ciclo de programación será más alta y las decisiones deberán tomarse, en gran proporción, en función de las circunstancias reales de cada momento. Las subfunciones que podemos distinguir dentro de la función programación son las siguientes:

- sub-función *carga (loading)*: asignación de las operaciones a centros de trabajo, decisión que se adoptará por comparación entre la capacidad disponible del centro y la carga

requerida por las operaciones ya asignadas al mismo,

- sub-función *secuenciación (sequencing)*: secuenciación de las operaciones asignadas a un centro de trabajo para establecer su orden de ejecución,

- sub-función *temporización (scheduling)*: determinación de los instantes de inicio y fin (programados) de cada operación.

Los objetivos perseguidos (dentro de las consideraciones generales establecidas por la planificación) suelen ser:

- terminar en plazo un alto porcentaje de órdenes,
- obtener una alta utilización del equipo y/o del personal,
- reducir al mínimo las horas extra,
- reducir al mínimo la obra en curso,
- etc.

claramente antagónicos entre sí dada una situación concreta del sistema productivo, por lo que deberá establecerse una jerarquización entre ellos.

La marcha a seguir suele ser la siguiente:

a) partimos de órdenes de trabajo, cada una de ellas con su fecha planificada de terminación o vencimiento,

b) las órdenes de trabajo se transforman en operaciones específicas, para cada una de las cuales se determinan las necesidades de mano de obra, maquinaria, etc. en las diversas alternativas,

c) las órdenes se cargan a centros de trabajo concretos, dentro de intervalos específicos, en función de la carga,

d) se determinan las necesidades agregadas de mano de obra, tiempo máquina, etc. a nivel de centro de trabajo y se compara con las capacidades existentes,

e) en función del resultado de la comparación se toman decisiones con relación a movimientos de plantilla, tasas de producción, horas extra, subcontratación, etc. así como respecto a cambios de asignación de órdenes, modificando el centro y/o el intervalo,

f) se procede a la secuenciación en cada centro de trabajo,

g) se desarrolla el programa detallado.

6.1.1.1 Planificación de las órdenes de trabajo: plan de necesidades de capacidad (CRP)

Las órdenes planificadas se generan automáticamente durante el proceso de cálculo de necesidades (utilizando el método MRP u otro similar), en el que se ha tenido en cuenta el stock y la obra en curso, los plazos y las reglas de lotificación. El plan de necesidades de capacidad (*Capacity Requirements Planning* = CRP) nos dará para cada intervalo en que se divide el horizonte de planificación (por ejemplo, semanas) la carga de cada una de las instalaciones (por ejemplo, en horas estándar). Para la distribución en el tiempo de la carga existen diversas posibilidades; una de las más utilizadas es una técnica de programación "hacia atrás" (*backward scheduling*) que toma la fecha de vencimiento en la que la orden debe estar cumplida (*due date*) como un punto fijo y entonces determina la fecha en que cada operación (en que se descompone la orden) debe comenzar, utilizando unos "plazos inter-operaciones" estimación del tiempo que consumen normalmente los trabajos esperando el transporte, pasando de un centro de trabajo al siguiente y esperando en cola a ser procesados. Es conveniente hacer las siguientes consideraciones:

1. Es posible que una operación se planifique de manera que se inicie en un intervalo (semana) y termine en otro. Debe determinarse cómo se asignan en este caso las horas, a la semana inicial, a la final o repartidas.

2. El CRP se basa en el principio de "carga infinita". Esto significa que la carga requerida se asigna al centro de trabajo sin tener en cuenta si éste tiene suficiente capacidad o no. Puede utilizarse una regla alternativa de planificación de forma que, cuando la carga supera cierta magnitud, se asigne cierto porcentaje a la semana correspondiente y el resto a la siguiente.

3. Un riesgo siempre existente en la programación hacia atrás es el de que la fecha programada de inicio de una operación corresponda a semanas ya transcurridas. Existen dos enfoques posibles para tratar este caso. El primero consiste en acumular todas las horas en un intervalo ficticio "Pasado", a fin de que se adopte "a posteriori" algún mecanismo de solución. El segundo es utilizar para estas órdenes una programación "hacia delante", con el peligro de superar la fecha de cumplimiento de la orden emitida por el sistema MRP. Para evitarlo pueden reducirse, en este caso, los plazos inter-operaciones, y utilizar la descomposición y/o el solapamiento. Aun así pueden resultar ordenes retrasadas que deben quedar bien identificadas, para que un estrecho seguimiento pueda garantizar, en la medida de lo posible, su realización en plazo.

Para situaciones industriales normales la cantidad de información y de cálculo implicado por un CRP es grande, por lo que debe tratarse informáticamente, y aun así el trabajo es considerable. Sin embargo, la precisión del CRP, a pesar de su complejidad, es discutible, ya que se basa en muchas estimaciones de valores medios. Por ello no es utilizable, por

sí mismo, para la toma de decisiones a corto plazo, que suelen ligarse con la programación de operaciones (a capacidad finita), y es preciso proceder a los análisis más localizados que tienen lugar en la secuenciación.

Para analizar cómo funciona el procedimiento determinaremos el plazo (LT) del artículo X2341 (para el lote estándar de 100 unidades) descrito en la figura 4.1.4.11 con los datos sobre los centros de trabajo de la figura 4.1.4.12. Se han resumido los cálculos en la figura 6.1.1.1. Para pasar del tiempo de proceso en horas estándar a días se ha tenido en cuenta el CDE por máquina (resultado de dividir el CDE del centro de trabajo por el número de máquinas existentes en el mismo). Supongamos ahora que una orden de dicha pieza (#273) con dicho tamaño de lote debe estar disponible en la fecha de vencimiento 145 (indicada en número correlativo del día laborable del año); en la figura 6.1.1.2 hemos realizado el cálculo hacia atrás para determinar las fechas de cada operación.

X2341						lote : 100 piezas	
N° op.	CT	tiempo pr. + op. /lote hor. est.	tiempo pr. + op. /lote días	tiempo cola días	tiempo tránsito días	tiempo total días	tiempo acumulado días
10	1	13,5	2,2	1,5	0,5	4,2	4,2
20	3	8,7	1,5	2	0,5	4,0	8,2
30	5	25,2	5,2	2,5	0,5	8,2	16,4
40	8	22,5	4,7	1,5	1	7,2	23,6
50	9		4		1	5,0	28,6
60	7	27,8	2,6	1,5	0,5	4,6	33,2
70	6	42,9	3,9	2,5	0,5	6,9	40,1

Fig. 6.1.1.1 Plazo de fabricación de X2341

orden #273		100 unidades de X2341			fecha vencimiento : 145			
N° op.	CT	tiempo tránsito	llegada al CT	tiempo cola	fecha comienzo	tiempo pr+op	fecha fin	semana proceso
	almacén	0,5	145					
70	6	0,5	138,1	2,5	140,6	3,9	144,5	29
60	7	1	133,5	1,5	135,0	2,6	137,6	28
50	9	1	128,5		128,5	4	132,5	26/27
40	8	0,5	121,3	1,5	122,8	4,7	127,5	25/26
30	5	0,5	113,1	2,5	115,6	5,2	120,8	24/25
20	3	0,5	109,1	2	111,1	1,5	112,6	23
10	1		104,9	1,5	106,4	2,2	108,6	22

Fig. 6.1.1.2 Programación hacia atrás

En la figura 6.1.1.3 hemos asignado a cada sección/semana la carga correspondiente a la orden #273. En el caso de que el proceso se extienda sobre dos semanas hemos repartido las horas estándar proporcionalmente al intervalo en días correspondiente a cada semana.

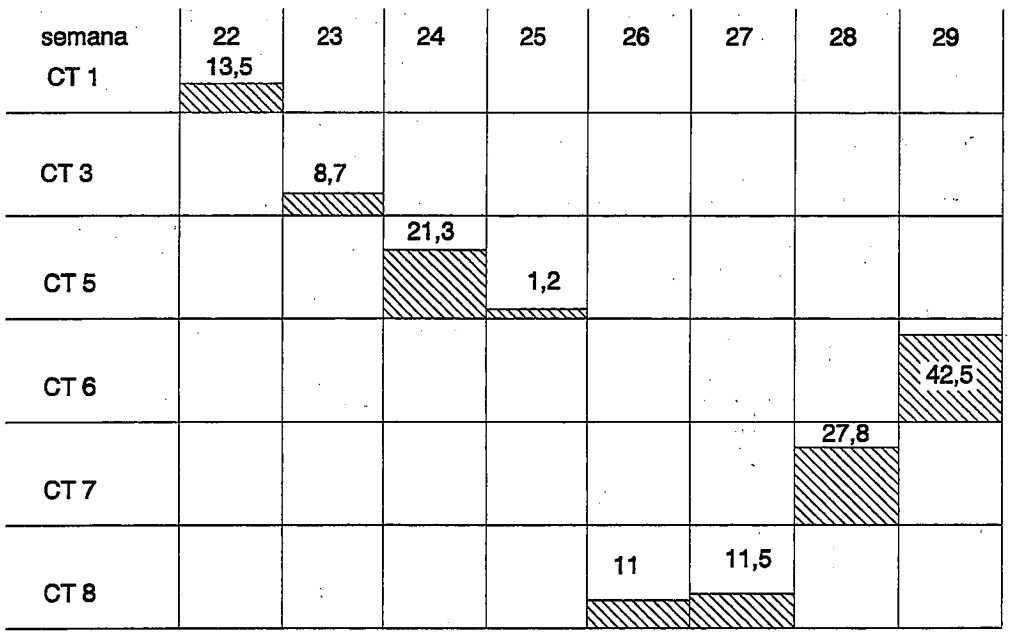


Fig. 6.1.1.3 Carga por centro de trabajo y semana correspondiente a la orden #273

Acumulando la carga de las diferentes órdenes podemos obtener una tabla para cada semana del tipo de la representada en la figura 6.1.1.4. Acumulando la carga podemos compararla con la capacidad (obtenida, por ejemplo, considerando el valor CDE multiplicado por 5, si en la semana existen 5 días laborables). En el ejemplo de la figura detectamos una sobrecarga en el CT número 4, que si consideramos insalvable deberemos corregir desplazando operaciones a otras semanas con carga menor que la capacidad.

Semana 25 Orden de producción	Centro de trabajo							
Número	1	2	3	4	5	6	7	8
#269	14,2			8,5				
#271		18,7						
#272			10,5		20,2			
#273					15,7			12,9
#274							30,6	
#275						25,7		
...	...	...	...	...	...	...	...	...
carga total	108,3	92,0	87,3	66,1	67,4	45,3	38,2	75,1
capacidad (h.e.)	122,5	115,5	115,0	57,5	72,0	54,5	54,5	120,0
carga (%)	88,4	79,7	75,9	115,0	93,6	83,1	70,1	62,6

Fig. 6.1.1.4 Carga de trabajo de las órdenes de trabajo planificadas y en curso (no terminadas) para la semana 25

6.1.1.2 Carga de máquinas por el método de los índices

Cuando existen varios centros de trabajo alternativos que pueden realizar las operaciones, si todos ellos posen la misma eficiencia, la forma de asignar las operaciones, según las circunstancias, podrá oscilar entre los dos extremos siguientes:

- asignar las operaciones procurando mantener una carga homogénea en todas las estaciones de trabajo, es decir, cuando se asigne una nueva operación se hará en el centro de trabajo con menos carga asignada hasta el momento en el intervalo considerado,

- asignar las operaciones de forma que se saturen sucesivamente los centros de trabajo, es decir, se asignarán todas las operaciones al primer centro de trabajo hasta que se llegue a un nivel límite de la carga, pasándose a continuación al segundo, etc.

Si los centros de trabajo tienen eficiencias diferentes, sería deseable asignar cada operación al centro de trabajo que presenta mayor eficiencia para la misma, pero si no existe suficiente capacidad en áquel se deberá recurrir a los demás. El problema puede complicarse si existen varias operaciones en las mismas circunstancias. Un procedimiento muy sencillo que puede utilizarse es el de los índices, que pasamos a describir a continuación. En la tabla de la figura 6.1.1.5 se indican 7 operaciones que pueden desarrollarse en 3 centros de trabajo distintos, midiéndose la eficacia de los mismos a través del tiempo de elaboración correspondiente.

Operación	Centros de trabajo		
	A	B	C
1001	90	(40)	60
1002	50	44	(15)
1003	30	18	(12)
1004	(24)	32	36
1005	14	7	(5)
1006	12	(9)	18
1007	(8)	12	10
Capacidad disponible	50	47	28

Fig. 6.1.1.5 Carga y capacidad (en horas) de centros de trabajo alternativos

Determinamos los índices de la manera siguiente: para cada operación el centro más eficiente recibe el valor 1, mientras que para los demás se calcula el índice efectuando el cociente entre la duración de la operación en dicho centro y la duración en el centro más eficiente.

A continuación asignaremos las operaciones de la siguiente forma: calculamos la diferencia de los dos índices de los dos centros más favorables para la realización de la operación *en los que exista capacidad suficiente para realizarla*. La mayor de las diferencias nos indica la operación a asignar, y lo haremos a aquel centro con capacidad suficiente y menor índice. Disminuiremos la capacidad disponible del centro y proseguiremos aplicando el procedimiento. Cuando una operación sólo pueda realizarse en un centro, la asignaremos directamente a él. En la tabla de la figura 6.1.1.6 hemos determinado los índices para el caso del ejemplo.

Obsérvese que, dada la carga que representan y las disponibilidades de los centros, la operación 1001 sólo puede realizarse en el centro B, y que la operación 1004 no puede realizarse en el C. Iniciaremos el procedimiento asignando 1001 al centro B. La diferencia mayor la tenemos inicialmente para la operación 1002, cuyo mejor índice es 1,00 y el siguiente 2,93. Después de asignar 1001 a B y 1002 a C quedan en B 7 horas disponibles y en C 13, por lo que el número de trabajos realizables en ellos se reduce todavía más. Las operaciones 1004 y 1006 sólo pueden asignarse a A.

	Centros de trabajo					
	A		B		C	
Operación	horas	índice	horas	índice	horas	índice
1001	90	2,25	40	1,00	60	1,50
1002	50	3,33	44	2,93	15	1,00
1003	30	2,50	18	1,5	12	1,00
1004	24	1,00	32	1,33	36	1,50
1005	14	2,80	7	1,40	5	1,00
1006	12	1,33	9	1,00	18	2,00
1007	8	1,00	12	1,50	10	1,25
Capacidad disponible	50		47		28	

Fig. 6.1.1.6 Capacidad e índices

	Centros de trabajo					
	A		B		C	
Operación	horas	índice	horas	índice	horas	índice
1001	90	2,25	40*	1,00	60	1,50
1002	50	3,33	44	2,93	15*	1,00
1003	30	2,50	18	1,5	12*	1,00
1004	24*	1,00	32	1,33	36	1,50
1005	14	2,80	7*	1,40	5	1,00
1006	12*	1,33	9	1,00	18	2,00
1007	8*	1,00	12	1,50	10	1,25
Capacidad disponible	50		47		28	
asignada	44		47		27	
restante	6		0		1	

Fig. 6.1.1.7 Carga después de asignar todas las operaciones

Como resultado, 1003 sólo se puede asignar a C. Quedan pendientes 1005 y 1007, para las que todavía queda elección; aplicando la regla asignamos 1005 a B y 1007 a A, aunque en este caso la dificultad no era muy grande ya que ambas operaciones no competían por el mismo centro de trabajo. La situación final es la de la tabla de la figura 6.1.1.7.

6.1.1.3 Modelos matemáticos

El procedimiento del apartado anterior no posee una gran sofisticación matemática, aunque cualquier intento en tal sentido exige disponer tanto de una formalización no ambigua de la situación de referencia como de unos criterios precisos que permitan evaluar una asignación de tareas a centros de trabajo. A guisa de ejemplo presentamos un modelo lineal debido a S. Eilon.

Supongamos que la producción a realizar de n productos durante un determinado período es $Q_1, Q_2, \ldots, Q_n$ unidades respectivamente; se dispone de m máquinas tales que $r_{j,i}$ es la tasa de producción de la pieza *i* en la máquina *j* medida en unidades por unidad de tiempo, L_j es el tiempo disponible en la máquina *j*.

Llamando $x_{j,i}$ a las unidades de *i* realizadas en la máquina *j*, un posible planteo será:

$$[\text{MIN}]\, z = \sum_{j=1}^{m} \sum_{i=1}^{n} c_{j,i} \cdot x_{j,i}$$

s.a.

$$\sum_{j=1}^{m} x_{j,i} = Q_i \qquad i = 1,2,3,\ldots,n$$

$$\sum_{i=1}^{n} \frac{x_{j,i}}{r_{j,i}} \leq L_j \qquad j = 1,2,\ldots,m$$

$$x_{j,i} \geq 0 \quad (\ y \text{ entero, eventualmente})$$

donde $c_{j,i}$ es el coste unitario de realización de la pieza i en la máquina j. Dado que en esta situación, aunque $x_{j,i}$ deba ser entero, habitualmente no se producirá inconveniente mayor redondeando resultados fraccionarios, podemos utilizar el algoritmo símplex.

La no disponibilidad de costes puede obligar a utilizar otra función económica, tal como el tiempo libre que quede disponible en las máquinas, lo que llevaría al siguiente planteo:

$$[MAX]z = \sum_{j=1}^{m} w_j \cdot y_j$$

s.a.

$$\sum_{j=1}^{m} x_{j,i} = Q_i \qquad i = 1,2,3,\ldots,n$$

$$\sum_{i=1}^{n} \frac{x_{j,i}}{r_{j,i}} + y_j = L_j \qquad j = 1,2,\ldots,m$$

$$x_{j,i} \geq 0 \quad (\ y \text{ entero, eventualmente})$$
$$y_j \geq 0$$

donde w_j son unos pesos.

En el caso en que la misma máquina deba realizar toda la producción de una pieza el modelo se complica. Sea $u_{j,i}$ una variable binaria que vale 1 si la pieza i se realiza en la máquina j y 0 en caso contrario. El modelo quedaría:

$$[MAX]\, z = \sum_{j=1}^{m} w_j \cdot y_j$$

s.a.

$$\sum_{j=1}^{m} u_{j,i} = 1 \qquad i = 1,2,3,\ldots,n$$

$$\sum_{i=1}^{n} u_{j,i} \cdot \frac{Q_i}{r_{j,i}} + y_j = L_j \qquad j = 1,2,\ldots,m$$

$$u_{j,i} = \{0,1\}$$
$$y_j \geq 0$$

por lo que se podría introducir fácilmente la consideración del tiempo de preparación de máquinas (independiente de la secuencia).

6.1.1.4 Análisis entrada/salida

El análisis entrada/salida (IAO = *Input/Output Analysis*) se desarrolló hacia 1970 con la intención de construir una herramienta para controlar las colas de trabajos en espera delante de un centro de trabajo (CT). Puesto que el tamaño de cola afecta el plazo total de realización de las órdenes de fabricación, esta técnica sirve para controlar los plazos o LT. El stock al final de un período obedece a la ley de conservación de flujos o de Kirchoff, y la cola delante de un CT constituye la obra en curso. Midiendo todas las variables en horas estándar:

$$(\text{cola inicial})_t + (\text{trabajos llegados})_t -$$
$$- (\text{trabajos realizados})_t = (\text{cola final})_t =$$
$$= (\text{cola inicial})_{t+1}$$

La tasa de llegadas de nuevos trabajos al CT puede regularse, especialmente para el CT

inicial (puerta), gestionando el número de trabajos emitidos (o liberados), retrasando o adelantando, en su caso, las órdenes sugeridas por el sistema MRP. También es posible regular, en el corto plazo, la tasa de salida del CT programando, por ejemplo, horas extra. IOA es, por tanto, un procedimiento mediante el cual pueden estimarse, para un futuro relativamente próximo, las necesidades de capacidad y la capacidad disponible, y por tanto puede utilizarse como soporte para las decisiones de gestión de la capacidad.

La base de IOA es el informe IOA, del que en la bibliografía se han sugerido diferentes tipos; el reproducido en la figura 6.1.1.8 es uno de los más completos.

CT : A122 Semana	8	9	10	Actual	12	13
Entrada planific. Entrada real Desv. Acum. Media	65 53 -12,0	76 79 -4,5	79 75 -4,3	85	85	85
Salida planific. Salida real Desv. Acum. Media	72 73 +1,0	72 65 -3,0	72 77 -0,3	80	80	80
Cola planific. Cola real 90	83 70	77 84	91 82	87	92	97
Entrada no emitida : 8 h Cola objetivo : 89 h Capacidad demostrada (3 semanas últimas) : 71,7 h/sem.						

Fig. 6.1.1.8 Informe IOA

El informe se extiende sobre seis semanas, tres en el pasado y tres en el futuro, aunque dicho número puede adaptarse a las necesidades; se inscriben datos planificados y reales.

- Entrada planificada: muestra la cantidad prevista de horas estándar que se espera que lleguen a un CT cada semana. Esta información puede provenir de diversas fuentes, por ejemplo del plan de cargas del CRP. No obstante, dichos planes se generan mediante programación hacia atrás utilizando tiempos de cola medios, por lo que las llegadas efectivas de órdenes al CT puede diferir considerablemente de las planificadas. Para la planificación a corto plazo es mejor alisar la entrada considerando la media sobre cierto número de períodos. Para el CT puerta en el que la entrada se determina mediante la emisión de órdenes, esto puede servir para equilibrar la utilización de capacidad; en los otros CT la entrada la define la salida del CT anterior, por lo que es más difícil de predecir y controlar.

- Entrada real: esta información proviene de control de ejecución y deben tomarse precauciones para que la información sea fiable.

- Desviación acumulada media: las desviaciones se acumulan a partir de la primera semana mostrada en el informe y se divide por el número de semanas. Indican si hay tendencia a sobreplanificar o subplanificar, aunque el horizonte de tres semanas adoptado es demasiado corto para obtener conclusiones definitivas; podría utilizarse un horizonte mayor y se mostraría en el informe sólo una parte de los valores.

- Salida planificada: coincide con la capacidad planificada para el CT, cifrada en horas estándar; debe considerarse como una variable de decisión cuyo valor se determinará considerando la cola planificada resultante (en la figura 6.1.1.8 se ha planificado un aumento de la salida a partir de la semana 11 que se corresponderá a una decisión sobre realización de horas extra, por ejemplo).

- Salida real: esta información proviene también de control de ejecución.

- Desviación acumulada media: tiene la misma significación que la indicada en la entrada.

- Cola planificada: representa la cola esperada al final de la semana frente al CT, de acuerdo con la entrada y la salida planificadas. Los valores se recalculan cada semana partiendo de la última cola real.

- Cola real: la cola real se deriva semanalmente de la cola previa y de las entradas y salidas reales.

- Entrada no emitida: un sistema tal como MRP sugiere fechas de emisión de órdenes generalmente sin tener en cuenta el estado de carga de los CT; pueden resultar unos perfiles de necesidades de capacidad muy variables de una semana a otra. El planificador puede influir en la utilización de capacidad adelantando o retrasando la emisión de órdenes. Es inútil emitir una orden si se sabe que el CT ya está sobrecargado la próxima semana (y así se evita la obra en curso física) o si se sabe que el material requerido no llegará a tiempo. Las órdenes no emitidas, cuando lo sugiere MRP, se consideran entradas no emitidas y obviamente para el CT inicial son una cola implícita (posiblemente están incluidas en la entrada planificada). Es prudente, por tanto, considerar en la toma de decisiones la cola como la suma de la cola real y la entrada no emitida.

- Cola objetivo: las colas reales pueden fluctuar y normalmente lo harán de una semana a otra en función de las entradas y salidas reales. Es corriente determinar un tamaño medio de cola deseado que se convierte en la cola objetivo para utilizarlo en el proceso de planificación; los valores de entrada y salida pueden ajustarse de manera que la cola planificada se acerque al objetivo, aceptando fluctuaciones en torno al mismo.

- Capacidad demostrada: puede utilizarse la media de las salidas reales recientes, en nuestro caso una media móvil sobre 3 semanas.

6.1.2 Situación antes de la secuenciación

En los párrafos siguientes nos interesaremos por los problemas de secuenciación de las operaciones de las órdenes de trabajo, que constituye la programación detallada a capacidad finita. Supondremos en general que las operaciones de las órdenes de trabajo han sido asignadas mediante algún procedimiento a centros de trabajo concretos, y que las órdenes están situadas aproximadamente en el tiempo, es decir, que por lo menos existe una fecha en la que deben estar realizadas.

Más concretamente, cada orden está caracterizada por unas fechas de emisión y de vencimiento, una cantidad a producir (que permitirá estimar el tiempo necesario para cada operación) y una ruta o secuencia de operaciones a realizar.

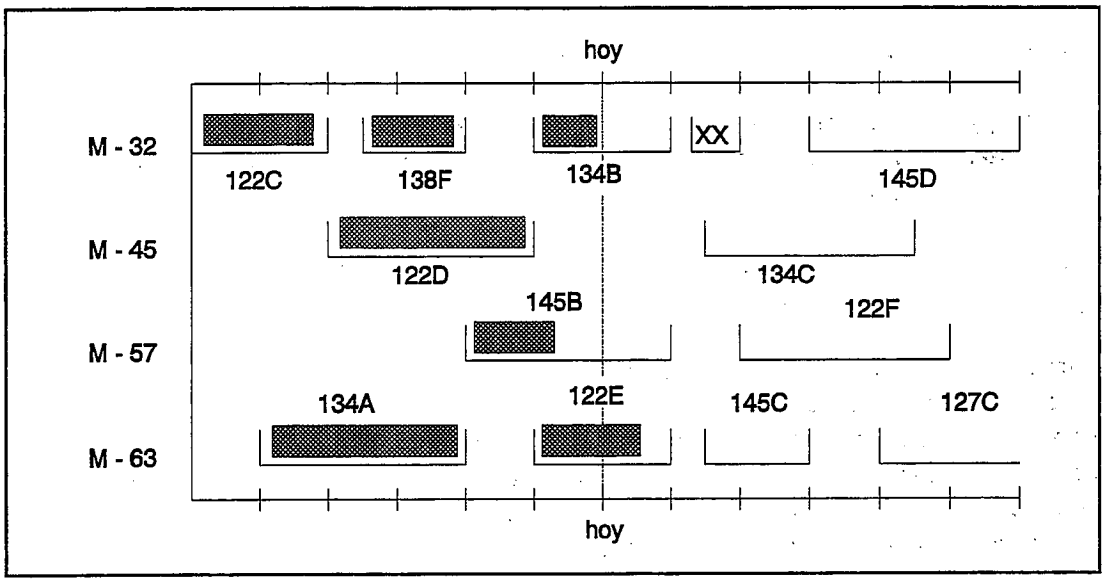


Fig. 6.1.2.1 Diagrama de Gantt: *resulta intuitivo apreciar la riqueza de las convenciones que pueden utilizarse, y añadiendo variaciones de colores y tramas en los rectángulos las posibilidades son más ricas todavía (134B está en programa, 145B retrasada y 122E adelantada: M-32 tiene un período programado de inactividad)*

Uno de los instrumentos más utilizados, tanto para la realización de la programación como para la transmisión de sus resultados, es el diagrama de barras o de Gantt, que puede llegar a alcanzar un gran nivel de sofisticación (véase la figura 6.1.2.1). Existen determinados dispositivos, paneles con ranuras asociados a rectángulos de cartulina de diferentes colores o paneles con orificios destinados a recibir clavijas de diferentes formas y colores, etc. que intentan materializar dicho diagrama. Un gran porcentaje de ellos los hemos visto en los departamentos de las empresas con capas de polvo que mostraban su

reducida utilización, aunque en otros casos (especialmente en el sector textil) su utilización efectiva se remonta a muchos años, habiéndose desarrollado procedimientos propios. El trabajo que exige mantener al día un programa no se elimina con los dispositivos aludidos, y creemos que un papel cuadriculado de amplias dimensiones, un lápiz y una goma siguen siendo las herramientas más utilizadas en la programación por métodos intuitivos de prueba y error. Sólo puede superar su utilidad un sistema informático capaz de presentar en pantalla un diagrama de Gantt, con diversas posibilidades de "zoom", que asuma con rapidez las modificaciones, mostrando sus consecuencias, y capaz de trasladar en papel, para una visión global, el diagrama definitivo.

6.1.3 Secuenciación: introducción

El prototipo de problema de secuenciación se encuentra en el denominado *Problema del Taller Mecánico* (*Job-Shop Problem*) cuyo enunciado básico es de la forma:

n piezas (lotes, trabajos u órdenes) deben realizarse en m máquinas (secciones o puestos de trabajo). La realización de cada pieza consiste en someterla a una serie de operaciones prefijadas; cada operación está asignada a una máquina concreta y tiene una duración determinada conocida. Debe establecerse un programa, es decir, la secuencia de operaciones en cada máquina, que optimice un cierto índice de eficacia (por ejemplo, la ocupación total del taller).

Observemos en primer lugar que existen dos tipos de secuencias:

a) La secuencia en que una pieza "ve" pasar las máquinas, en principio establecida "a priori", constituyendo la "ruta" de las operaciones de la pieza,

b) La secuencia en que una máquina "ve" pasar las piezas, que es precisamente la incógnita del problema.

6.1.3.1 Un ejemplo

Vamos a enunciar un problema de secuenciación introductorio muy simple, en el que no aparecen talleres, máquinas ni piezas, que servirá para apreciar las dificultades que pueden presentarse.

Cuatro amigos, ALEX, BLAS, CARLOS y DAVID comparten un mismo apartamento. Los domingos reciben, a primera hora de la mañana, cuatro periódicos: La Vanguardia (LV), El País (EP), Avui (AV) y El Mundo Deportivo (MD). Todos ellos han adquirido a lo largo de su vida una serie de manías que les llevan a seguir unas rutinas invariables, en particular

están habituados a leer los periódicos en un cierto orden, de forma que están dispuestos a esperar que quede libre el periódico correspondiente, aunque estén libres otros que todavía no han leído. Por otra parte, son capaces de compaginar, si es preciso, la lectura de un periódico con otras actividades: afeitarse, ducharse, vestirse, desayunar, etc.

Alex se levanta a las 8 h y sigue el siguiente orden de lectura: LV durante 70 minutos, AV durante 10 minutos, EP durante 20 minutos y MD durante 5 minutos.

Blas se levanta a las 8 h 30 min., y su orden y tiempos son: EP 60 min., LV 20 min., MD 20 min., AV 5 min.

Carlos se levanta también a las 8 h 30 min., y su orden y tiempos son: AV 45 min., LV 30 min., MD 15 min., EP 5 min.

Finalmente David se levanta a la 9 h, y su orden y tiempos son: MD 30 min., LV 5 min., EP 5 min., AV 5 min.

Un domingo determinado desean salir juntos de excursión lo antes posible, sin abandonar su rutina; ¿cómo deben programar la lectura de los periódicos?, ¿a qué hora podrán salir?

	Alex		Blas		Carlos		David	
hora de inicio	8 h		8 h 30 min.		8 h 30 min.		9 h	
	periód.	tiempo	periód.	tiempo	periód.	tiempo	periód.	tiempo
1ª lect.	LV	70	EP	60	AV	45	MD	30
2ª lect.	AV	10	LV	20	LV	30	LV	5
3ª lect.	EP	20	MD	20	MD	15	EP	5
4ª lect.	MD	5	AV	5	EP	5	AV	5

Fig. 6.1.3.1 Datos del problema Los 4 amigos

Puesto que la ruta de lectura de cada uno de los amigos está prefijada, estos constituyen las "piezas", mientras que los periódicos hacen las veces de las "máquinas". Debemos determinar por tanto una secuencia de lectura para cada periódico. Un conjunto de secuencias posible (solución posible) es, por ejemplo, el de la tabla de la figura 6.1.3.2. Sin embargo, un programa es algo más que un conjunto posible de secuencias, en un programa debemos indicar el instante de comienzo y fin de cada operación, además de *secuenciar* debemos *temporizar*. A cada conjunto de secuencias posible podemos asociar infinitas temporizaciones, calendarios o programas, puesto que antes de cada operación podemos intercalar tanto tiempo muerto como deseemos; sin embargo, a cada conjunto de secuencias posible corresponde un solo programa semiactivo (más adelante concretaremos más el término) sin tiempos muertos innecesarios. En nuestro caso el programa

semiactivo corresponde a una duración total de la lectura (desde que Alex empieza con La Vanguardia hasta que Carlos termina con El País) de 165 minutos, lo que fija el momento de la salida a las 10 h 45 min. Se trata de un programa bastante bueno, pero no el mejor, como veremos más adelante.

La obtención de la duración total se puede realizar mediante el diagrama de Gantt (véase la figura 6.1.3.4) o mediante una estructura tabular (figura 6.1.3.3).

Periódico	LV	EP	AV	MD
1er lector	A	B	C	D
2º lector	B	A	A	B
3er lector	D	D	D	A
4º lector	C	C	B	C

Fig. 6.1.3.2 Una solución posible del problema

Periódico	LV			EP			AV			MD		
	lect.	inic.	fin	lect.	inic.	fin	lect.	inic.	fin	lect.	inic.	fin
1ª lect.	A	0	70	B	30	90	C	30	75	D	60	90
2ª lect.	B	90	110	A	90	110	A	75	85	B	110	130
3ª lect.	D	110	115	D	115	120	D	120	125	A	130	135
4ª lect.	C	115	145	C	160	165	B	130	135	C	145	160

Fig. 6.1.3.3 Programa semiactivo correspondiente a la solución anterior

En el caso considerado, y en sus posibles generalizaciones, cada pieza tiene una operación en cada máquina, por lo que un conjunto de secuencias consta de *m* secuencias, una en cada máquina, constituida por una permutación de las *n* operaciones. Por tanto, existen teóricamente:

$$(n!)^m$$

conjuntos de secuencias diferentes; en nuestro caso:

$$(4!)^4 = 332.676$$

No todos los conjuntos de secuencias son posibles, ya que las rutas de lectura imponen ciertas condiciones. Por ejemplo, no podemos imponer que C lea La Vanguardia antes que A, y A Avui antes que C. En general es más fácil generar conjuntos de secuencias posibles que descartar, de todos los conjuntos imaginables, aquéllos que no son posibles.

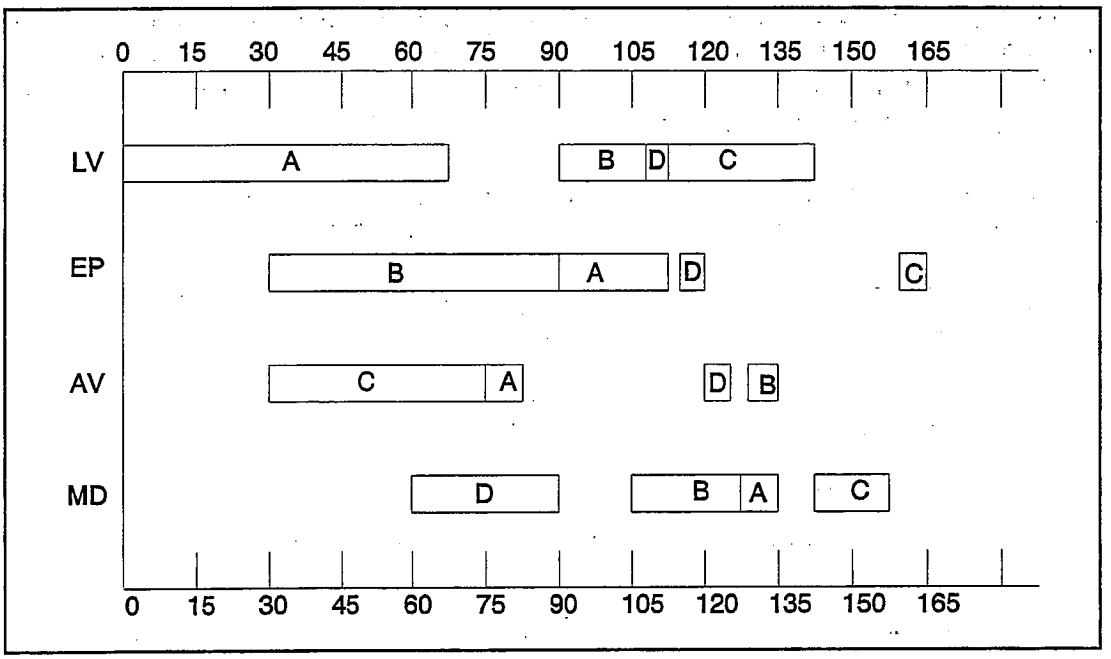


Fig. 6.1.3.4 Diagrama de Gantt correspondiente a la solución del problema Los 4 amigos *indicada en la figura 6.1.3.3*

Periódico	LV	EP	AV	MD
1er lector	A	B	C	D
2° lector	B	C	D	A
3er lector	C	D	A	B
4° lector	D	A	B	C

Fig. 6.1.3.5 Una solución no posible del problema

6.1.3.2 Problemas estáticos y problemas dinámicos

Una clasificación inicial sencilla de los problemas del taller mecánico los divide en estáticos, semidinámicos y dinámicos. Un problema estático es aquél en el que el número de piezas es finito, y están todas disponibles, así como las máquinas, en el instante inicial que habitualmente denominamos instante 0 en tiempo relativo. Habitualmente es este caso se intenta determinar un programa que minimice la ocupación total del taller, marcada por el instante en que termina la última operación, u otro criterio de tipo global análogo. A pesar de su trivialidad aparente el problema puede presentar un grado apreciable de complejidad.

Un problema semidinámico también considera un número finito de piezas, pero los instantes de disponibilidad de las piezas y/o las máquinas no son todos idénticos. El problema "Los 4 amigos" es de este tipo. El nivel de complejidad es superior al anterior, aunque la problemática es del mismo tipo, así como la naturaleza de los criterios para juzgar de la calidad de un programa.

Un problema dinámico considera el horizonte de funcionamiento del taller ilimitado hacia el futuro, aunque el conocimiento concreto en un instante determinado se limite a la situación actual y a una cartera finita de piezas a realizar. Progresivamente algunas piezas terminan su elaboración en el taller y lo abandonan, mientras que otras nuevas llegan a él para ser tratadas. No podemos pretender hallar aquí un programa único, sino un conjunto de programas sucesivos desarrollados a lo largo de ciclos de reprogramación. Por tanto, no podrá juzgarse la calidad de un programa sino, en todo caso, la de un procedimiento de reprogramación, y la naturaleza del criterio que permita juzgar la calidad del procedimiento estará entroncada con el comportamiento medio de los programas obtenidos utilizándolo frente a los *inputs* (características de llegada y estructura de las piezas) a que está sometido.

6.1.3.3 Hipótesis generalmente aceptadas en los problemas estáticos

Siguiendo a Conway, Maxwell y Miller (*Theory of Scheduling*) las hipótesis habitualmente aceptadas en el problema del taller mecánico son las siguientes:

1- Cada máquina está continuamente disponible desde 0 hasta T, con T arbitrariamente grande.

2- No hay montajes (convergencias) ni particiones en lotes (divergencias). Para cada operación i existe una sola precedente inmediata h (se exceptúa la primera operación de cada pieza, que no tiene precedente); y una sola siguiente inmediata j (se exceptúa la última operación de cada pieza que no tiene siguiente).

3- Cada operación puede hacerse en un solo tipo de máquina del taller.

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

5- Cuando una operación ha comenzado debe terminarse antes de empezar 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 tratar una sola operación a la vez.

8- La única restricción activa en el taller es la relativa a las máquinas; no hay problemas de disponibilidad de mano de obra, materiales, etc.

Aunque varias de estas hipótesis pueden parecer muy duras, alejando el problema teórico considerado de los problemas reales, progresivamente relajaremos algunas, adaptando más los conceptos tratados a las situaciones reales.

6.1.3.4 Nomenclatura

La tipología de los problemas de secuenciación suele describirse mediante una notación muy simple formada por cuatro símbolos:

A / B / C / D

A corresponde a la llegada de las piezas. En los problemas dinámicos puede indicar una ley de probabilidad; en los estáticos será el número de piezas (*n* indica un número de piezas arbitrario),

B describe el número de máquinas en el taller (*m* indica un número arbitrario),

C se refiere al tipo de flujo de las piezas por el taller, y generalmente toma uno de los valores F, P, R o G:

> *F* indica flujo regular (*flow-shop*), todas las piezas tienen esencialmente la misma ruta. Las máquinas pueden numerarse 1, 2,....., m de forma que si una pieza tiene una operación en la máquina *r* y la siguiente en la *s*, $r < s$ (habitualmente $s = r + 1$, pero se admite que algunas piezas no tengan ninguna operación en algunas máquinas),
>
> *P* indica un caso particular del anterior en el que, además de los indicado, todas las máquinas tienen la misma secuencia de piezas, es decir la solución buscada es simplemente una permutación de las piezas, que constituye la secuencia en todas las máquinas,
>
> *R* indica rutas aleatorias (*randomly routed job-shop*),
>
> *G* indica flujo general, es decir, la situación en que la ruta de una pieza tiene dos operaciones sucesivas en las máquinas *r* y *s*, y la de otra pieza tiene dos operaciones sucesivas en *s* y *r*,

D se refiere al índice de eficiencia elegido para evaluar los programas, normalmente tomará los valores F_{max}, c_{max}, F_{med}, T_{max}, etc. cuyo significado se concretará en lo que sigue.

Con esta nomenclatura no podemos identificar todos los posibles tipos de problemas del taller mecánico, ni tan siquiera todos los que aparecen en la bibliografía, muy extensa, existente sobre el tema. Algunos autores han propuesto nomenclaturas más voluminosas, generalmente extensión de la indicada, pero creemos que en el presente texto ésta es suficiente.

Para los diferentes conceptos utilizaremos la siguiente notación. Dada una pieza i:

$r_i = r_{1,i}$ es el instante de entrada de la pieza en el taller (*ready time*),

d_i es el instante comprometido de salida de la pieza *i* del taller (*due date*, fecha de vencimiento),

g(i) es el número de operaciones de la pieza *i*,

a_i es el tiempo concedido para la realización de la pieza *i* (*total allowance for time in the shop*),

$$a_i = d_i - r_i$$

Cada operación estará definida por la máquina que debe realizarla y su duración:

Máquina	Duración
$m_{1,i}$	$p_{1,i}$
$m_{2,i}$	$p_{2,i}$
...	...
$m_{g(i),i}$	$p_{g(i),i}$

$$\text{con} \quad 1 \leq m_{k,i} \leq m \qquad p_i = \sum_{k=1}^{g(i)} p_{k,i}$$

La realización de las operaciones puede conllevar tiempos muertos en las máquinas:

$w_{k,i}$ es el tiempo de espera de la operación *k* de la pieza *i* (*waiting time*)

$$\text{con} \qquad w_i = \sum_{k=1}^{g(i)} w_{k,i}$$

Dados todos los valores $w_{k,i}$ quedan fijadas las secuencias y quedan definidos los demás valores:

1) Instantes en que las piezas salen del taller, c_i (*completion time*),

2) Tiempo de permanencia de una pieza en el taller, F_i (*flow time*),

3) Diferencia entre el instante de salida real y previsto, L_i (huelgo o *lateness*).

Cumpliéndose las relaciones:

$$c_i = r_i + w_{1,i} + p_{1,i} + w_{2,i} + p_{2,i} + \ldots\ldots + w_{g(i),i} + p_{g(i),i} = r_i + w_i + p_i$$

$$F_i = w_i + p_i = c_i - r_i$$

$$L_i = c_i - d_i = F_i - a_i$$

Podemos definir también el retraso, T_i (*tardiness*) y el adelanto, E_i (*earliness*), que responden a:

$$T_i = max\ \{ 0, L_i \}$$

$$E_i = max\ \{ 0, -L_i \}$$

Aunque parezca una perogrullada conviene tener presente que sólo podemos tener retraso o adelanto (o huelgo) si existe una fecha de vencimiento comprometida para la finalización.

6.1.3.5 Medidas de eficacia

Una medida de eficacia permite clasificar los programas en clases ordenadas, y por tanto definir, en función de ella, programas mejores y peores. En general un programa será mejor que otro si su medida de eficacia tiene un valor más pequeño (tal vez deberíamos hablar de medida de "ineficacia").

Una medida de eficacia es *regular* si es tal que puede ponerse en función de los instantes de salida de las piezas:

$$M = f(c_1, c_2, \ldots, c_n)$$

y que no disminuye (mejora) cuando algunos o todos los valores de c_i aumentan; por tanto, M aumenta sólo si por lo menos uno de los valores c_i aumenta:

$M' > M$ implica que existe un i tal que $c'_i > c_i$

Nos interesaremos únicamente, de momento, en las medidas de eficacia regulares, aunque no sean las adecuadas para todas las circunstancias. En un contexto JIT, por ejemplo, puede ser tan penalizable un retraso como un adelanto, por lo que una medida de eficacia

podría tomar la forma:

$$\sum_{i=1}^{n} f_1(E_i) + \sum_{i=2}^{n} f_2(T_i)$$

donde f_1 y f_2 fuesen funciones no decrecientes definidas para argumentos positivos, que también podríamos escribir:

$$\sum_{i=1}^{n} f(c_i - d_i) = \sum_{i=1}^{n} f(L_i)$$

donde f fuese una función convexa y no negativa tal que f(0) = 0. Dicha medida de eficacia, que es razonable para este caso, no es regular.

Puesto que dado un programa cualquiera la eliminación de los tiempos muertos a base de adelantar el inicio de las operaciones tanto como sea posible, conservando las secuencias de las mismas en todas las máquinas, sólo puede mejorar una medida de eficacia regular, limitaremos nuestra consideración a dichos programas sin tiempos muertos o bien *semiactivos*. A partir de este momento consideraremos que a toda colección de secuencias (una para cada máquina) posible está asociado el programa semiactivo correspondiente, y que las medidas de eficacia de dicho programa son las medidas de eficacia del conjunto de secuencias. Como el conjunto de programas semiactivos es finito, en los problemas estáticos y semidinámicos (suponiendo finito el número total de operaciones a realizar), y son mejores que los no semiactivos, podrá hablarse de programa óptimo, aunque tal vez exista más de un programa óptimo.

Se comprueba fácilmente que la media o máximo del instante de salida del tiempo de permanencia, del huelgo o del retraso satisfacen las condiciones de medida de eficacia regular.

Algunas medidas de eficacia razonables son:

c_{max}: instante de salida de la *última* pieza del taller,

$$c_{\max} = \max_i \{c_i\}$$

F_{max}: tiempo de permanencia en el taller de la pieza que permanece más tiempo, es equivalente a la anterior si todos los $r_i = 0$,

$$F_{\max} = \max_i \{F_i\}$$

F_{med}: tiempo medio de permanencia en el taller,

$$F_{med} = \frac{1}{n} \cdot \sum_i F_i$$

T_{max}: retraso de la pieza que se retrasa más,

$$T_{max} = \max_i \{T_i\}$$

T_{med}: retraso medio,

$$T_{med} = \frac{1}{n} \cdot \sum_i T_i$$

etc.

Un problema $n/m/F/F_{max}$ (con todos los valores $r_i = 0$) conduce a buscar el programa, en un sistema con flujo regular, que minimiza la ocupación del taller. Si los valores de r_i son distintos entre sí, el problema semidinámico análogo será el $n/m/F/c_{max}$. El problema "Los 4 amigos" es del tipo $4/4/G/c_{max}$.

Algunas de estas medidas están relacionadas entre sí. Partiendo de:

$$L_i = F_i - a_i = c_i - r_i - a_i = c_i - d_i$$

obtenemos, sumando para todas las piezas y dividiendo por el número de las mismas:

$$L_{med} = F_{med} - a_{med} = c_{med} - r_{med} - a_{med} = c_{med} - d_{med}$$

Para un problema concreto a_{med}, r_{med} y d_{med} son constantes (independientes de las secuencias), por tanto, una solución óptima respecto L_{med}, lo es respecto a F_{med} y c_{med}.

Por otra parte dado que:

$$c_i = r_i + w_i + p_i$$

un argumento semejante nos muestra que minimizar c_{med} es equivalente a minimizar la espera media, w_{med}.

Además, puesto que $T_i - E_i = L_i$, tenemos $T_{med} - E_{med} = L_{med}$ pero ello no implica que el mínimo huelgo medio se corresponda con el mínimo retraso medio (salvo si las d_i son tan difíciles de alcanzar que $E_{med} = 0$). Tampoco la relación entre medias implica una semejante entre máximos; una solución óptima respecto a F_{med} lo es respecto a L_{med}, pero una óptima respecto a F_{max} puede no serlo respecto a L_{max}.

Los valores medios pueden estar ponderados con pesos que dependan de la importancia o coste de las piezas. En los problemas estáticos, la utilización media de las máquinas, bajo la hipótesis de las máquinas siempre disponibles y las piezas con instante de llegada 0, es:

$$U_{med} = \frac{\sum p_i}{m \cdot F_{max}}$$

Consideremos ahora el stock o la obra en curso, es decir, el número de piezas existentes en el taller. Sea:

N(t) el número de piezas que hay en el taller en el instante t,

$N_m(t_1,t_2)$ el número medio de piezas en el taller entre los instantes t_1 y t_2

$$N_m(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} N(t) \cdot dt$$

P(t) la suma de las duraciones de todas las operaciones de las piezas que están en el taller en el instante t (contenido de trabajo o trabajo total)

$P_m(t_1,t_2)$ contenido de trabajo medio entre los instantes t_1 y t_2

$$P_m(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} P(t) \cdot dt$$

Podemos establecer subdivisiones de N(t):

$N_p(t)$ número de piezas en proceso en el instante t,
$N_q(t)$ número de piezas en espera (o cola) en el instante t,

$$N(t) = N_p(t) + N_q(t)$$

y definir los valores medios; análogamente con P(t):

$P_o(t)$ suma de duraciones de las operaciones realizadas,
$P_p(t)$ suma de duraciones de operaciones en proceso,
$P_q(t)$ suma de duraciones de operaciones pendientes,

$$P(t) = P_o(t) + P_p(t) + P_q(t)$$

6.1.3.6 Relación entre permanencia y stock: Fórmula de Little

Consideremos un caso estático con todas las piezas disponibles, en el instante 0. En estas condiciones $F_{max} = c_{max}$ indica el instante de finalización de los trabajos en el taller, y el número medio de piezas que hay en éste durante dicho tiempo es fácil de calcular (más en forma gráfica que analíticamente):

$$N_m(0, F_{max}) = \frac{1}{F_{max}} \cdot \int_0^{F_{max}} N(t) \cdot dt$$

si renumeramos las piezas de forma que los valores de F_i estén en orden creciente, tendremos:

$$N_m(0, F_{max}) = \frac{1}{F_{max}} \cdot [n \cdot F_1 + (n-1) \cdot (F_2 - F_1) +$$

$$+ (n-2) \cdot (F_3 - F_2) + \ldots + 1 \cdot (F_n - F_{n-1})] =$$

$$= \frac{1}{F_{max}} \cdot \sum_{i=1}^{n} F_i = \frac{1}{F_{max}} \cdot n \cdot F_{med}$$

Por tanto :

$$\frac{N_m(0, F_{max})}{n} = \frac{F_{med}}{F_{max}}$$

La relación entre los tiempos medio y máximo de permanencia es igual a la relación entre el número medio y máximo de piezas en el taller.

Consideremos el caso semidinámico y numeremos las *n* piezas en el orden de llegada, y supongamos que salen del taller *en el mismo orden*:

$$r_1 \leq r_2 \leq \ldots\ldots\ldots \leq r_n$$
$$c_1 \leq c_2 \leq \ldots\ldots\ldots \leq c_n$$

Por definición tenemos:

$$N_m(0, c_n) = \frac{1}{c_n} \int_0^{c_n} N(t) \cdot dt$$

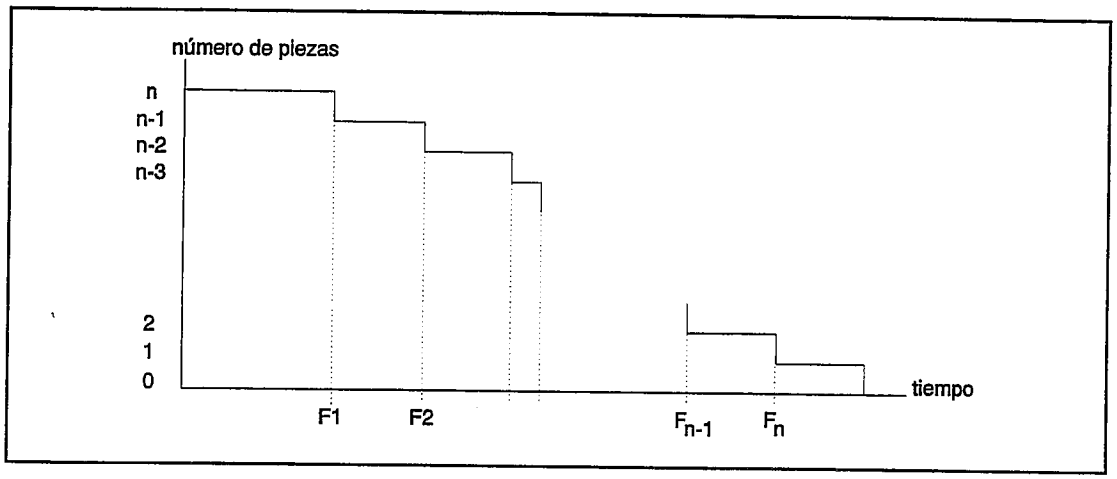


Fig. 6.1.3.6 Número de piezas en el taller en el caso estático. El número de piezas en el instante t, N(t), *toma la forma de una curva escalonada decreciente. El área comprendida entre la curva y los ejes de coordenadas puede considerarse como la yuxtaposición de rectángulos de altura 1 y base* F_1, F_2, .., F_n; *por tanto dicha área es igual a* ΣF_i, n *veces* F_{med}.

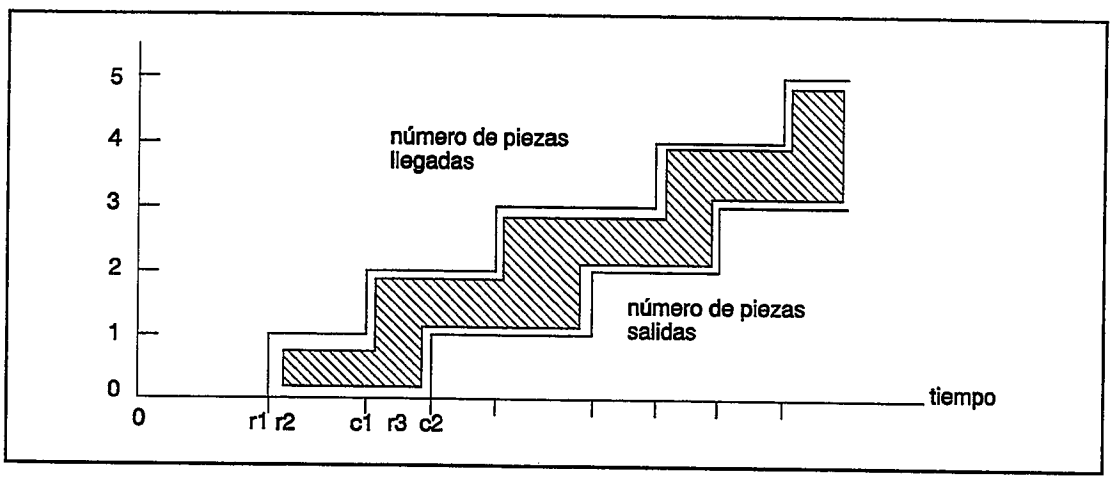


Fig. 6.1.3.7 Número de piezas en el taller en el caso dinámico.

En la figura 6.1.3.7 hemos indicado el número (acumulado) de piezas que llegan al taller en función del tiempo, y el número (acumulado) que salen. La parte sombreada indica las piezas que hay en un instante dado en el taller. Por tanto:

$$\int_0^{c_n} N(t) \cdot dt = \sum_{i=1}^{n} [c_i - r_i] = \sum_{i=1}^{n} F_i$$

de donde:

$$N_m(0, c_n) = \frac{1}{r_n + F_n} \sum_{i=1}^{n} F_i = \frac{n \cdot F_{med}}{r_n + F_n}$$

$$\frac{F_{med}}{N_m(0, c_n)} = \frac{r_n}{n} + \frac{F_n}{n}$$

r_n/n representa el intervalo medio entre llegadas de piezas (cuya inversa, *npu*, es el número medio de piezas que llegan al taller por unidad de tiempo); si n tiende a infinito la situación se transforma en el caso dinámico, si existe régimen permanente, F_n permanece finito y, por tanto, F_n/n tiende a 0

$$N_{med} = F_{med} \cdot npu$$

Si eliminamos la suposición de que todos los trabajos terminan en el mismo orden de llegada al taller, en el instante c_n pueden no haber terminado todos los trabajos. Sea $Y_i(t)$ la fracción (en tanto por uno) de F_i que todavía queda por realizar en el instante t:

$$\int_0^{c_n} N(t) \cdot dt = \sum_{i=1}^{n} [1 - Y_i(c_n)] \cdot F_i = \sum_{i=1}^{n} F_i - \sum_{i=1}^{n} Y_i(c_n) \cdot F_i$$

$$N_m(0, c_n) \cdot \left(\frac{r_n}{n} + \frac{F_n}{n}\right) = F_{med} - \frac{\sum Y_i(c_n) \cdot F_i}{n}$$

Si al crecer n el proceso alcanza el estado permanente F_n/n tenderá a cero y además el sustraendo del segundo miembro también tenderá a cero. Acotando:

$$F_{med} \geq N_m(0, c_n) \cdot \left(\frac{r_n}{n} + \frac{F_n}{n}\right) \geq F_{med} - \frac{N(c_n) \cdot F_{max}}{n}$$

y $N(c_n) \cdot F_{max}$ en estado permanente se conserva finito cuando n tiende a infinito. Por tanto, en cualquier caso se cumple:

$$F_{med} = \frac{N_{med}}{npu}$$

Esta expresión es formalmente idéntica a la utilizada en teoría de colas y conocida como

la fórmula de Little. En estado permanente *npu* también es el numero medio de piezas que salen totalmente elaboradas del taller por unidad de tiempo.

Si a un taller, que está bien equilibrado, llegan como media 5 órdenes nuevas o trabajos por día (y por tanto se terminan también en promedio 5 ordenes por día), y la permanencia media de un trabajo en el taller (elaboración más esperas) es 10 días, el stock medio en el taller es el correspondiente a 50 órdenes.

Esta expresión no depende del tipo de sistema, ni de sus fronteras, ni de la distribución de las llegadas (salvo que no deben congestionar el sistema), ni de cómo se programa, sólo de que se alcance el estado permanente. Se puede aplicar a una fábrica, a una máquina, a la producción global, a una línea de productos, a un producto individual (tomando cada vez los valores correspondientes para N_{med}, F_{med}, npu).

En un taller de m máquinas la utilización media será:

$$U_{med} = \frac{npu \cdot p_{med}}{m} \qquad \text{de donde} \qquad F_{med} = \frac{N_{med} \cdot p_{med}}{m \cdot U_{med}}$$

6.1.4 Secuencias finitas para una sola máquina

La consideracion de sólo 1 máquina puede parecer absurda o irreal, pero interesa desde el punto de vista teórico, como punto de partida y desde el punto de vista práctico pues tiene aplicación a ciertos casos reales en donde existe verdaderamente una sola máquina o instalación compleja funcionando sincronizadamente (industrias químicas y de proceso, detergentes, pinturas, etc.), o bien cuando una sola máquina es el "cuello de botella" del taller. En este caso:

$$m = 1 \; ; \; g(i) = 1 \; ; \; p_i = p_{1,i} \; ; \; w_i = w_{1,i}$$

Si además $r_i = 0$, entonces:

$$a_i = d_i \; ; \; c_i = F_i = w_i + p_i$$

Sea cual sea la medida de eficacia regular buscamos una permutación de las piezas (de las $n!$ existentes). Aquí no son útiles las interrupciones ni los tiempos muertos, como puede suceder en otros casos. Cualquier permutación, sin tiempos muertos, conduce a un programa semiactivo con:

$$F_{max} = p_1 + p_2 + \ldots + p_n = \sum_{i=1}^{n} p_i$$

por lo que F_{max} no es una medida de eficacia interesante. Llamaremos *[k]* a la pieza que

ocupa la *k*-ésima posición en una permutación, por tanto:

$$c_{[1]} < c_{[2]} < \ldots\ldots < c_{[n]}$$

y

$$c_{[k]} = c_{[k-1]} + p_{[k]} \quad para\ k > 1$$
$$c_{[1]} = p_{[1]}$$

En la tabla de la figura 6.1.4.1 hemos considerado un caso con 4 piezas, y hemos determinado algunos resultados correspondientes a 4 de las 24 permutaciones posibles; cada una de ellas es óptima con respecto a alguna medida de eficacia particular:

- la secuencia SPT minimiza F_{med} (y también L_{med}),
- la secuencia EDD minimiza T_{max} (y también L_{max}),
- la secuencia SFT maximiza T_{min} (y también L_{min}),
- la secuencia "X" (no obtenible en forma simple como las demás) minimiza T_{med}.

Pieza		1	2	3	4	F_{med}	T_{med}	T_{max}	T_{min}	L_{med}
Duración	p_i	2	4	3	1					
Fecha de vencimiento	d_i	1	2	4	6					
Secuencia SPT	c_i	3	10	6	1					
4-1-3-2	L_i	2	8	2	-5					
	T_i	2	8	2	0	5	3	8	0	1,75
Secuencia EDD	c_i	2	6	9	10					
1-2-3-4	L_i	1	4	5	4					
	T_i	1	4	5	4	6,75	3,5	5	1	3,5
Secuencia SFT	c_i	6	4	9	10					
2-1-3-4	L_i	5	2	5	4					
	T_i	5	2	5	4	7,25	4	5	2	4
Secuencia "X"	c_i	2	10	5	6					
1-3-4-2	L_i	1	8	1	0					
	T_i	1	8	1	0	5,75	2,5	8	0	2,5

Fig. 6.1.4.1 Secuencias diferentes para una sola máquina

6.1.4.1 Secuencia conforme a la duración (SPT = *shortest processing time*)

Esta secuencia se obtiene situando las piezas en orden creciente de duraciones, es decir:

$$p_{[1]} \leq p_{[2]} \leq \ldots\ldots\ldots \leq p_{[n]}$$

Corresponde a un tipo de regla heurística utilizada en ocasiones en los talleres "lanzar primero la operación de duración más corta". Esta secuencia minimiza F_{med}. Para comprobarlo llamemos:

$$F_{[s]} = \sum_{k=1}^{s} p_{[k]}$$

$F_{[s]}$ es la permanencia en el taller de la pieza que ocupa en la secuencia el lugar s y, por tanto, dado que $r_i = 0$, coincide con el instante de salida del taller de dicha pieza:

$$F_{[s]} = c_{[s]}$$

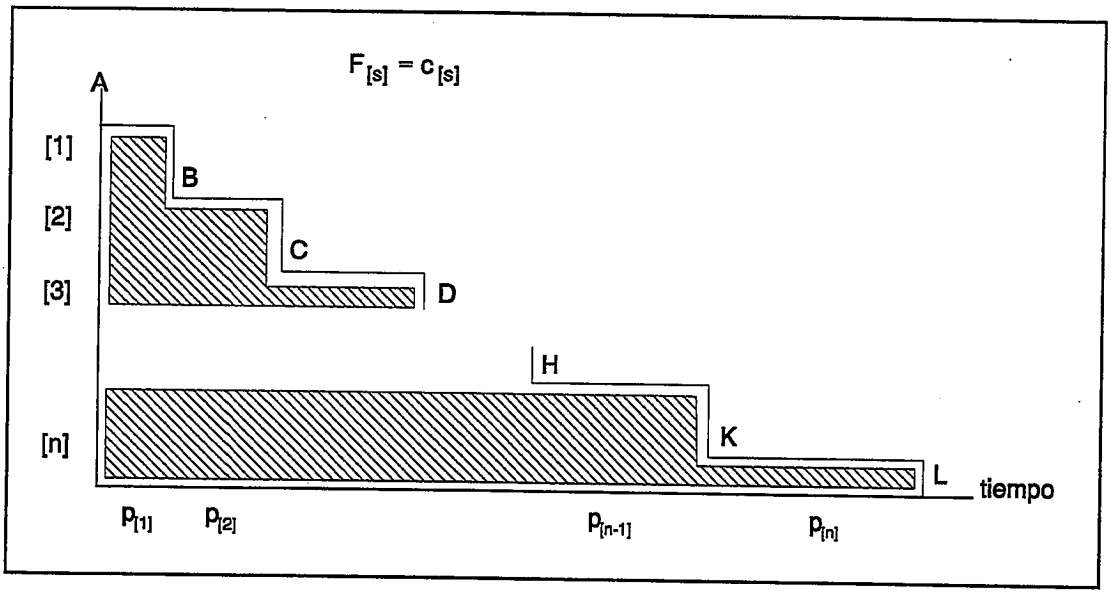


Fig. 6.1.4.2 Permanencia de n piezas en el problema 1-máquina. El área comprendida entre la línea escalonada y los ejes coordenados es proporcional a F_{med} *y por otra parte el punto* L *es fijo pues corresponde a la abscisa* F_{max}. *El área, y por tanto* F_{med}, *serán mínimos cuando la "curva" ABCD..HKL sea convexa.*

En la figura 6.1.4.2 se ha representado la situación. Por tanto:

$$F_{med} = \frac{\sum F_{[r]}}{n} = \frac{1}{n} \cdot \sum_{k=1}^{n} (n-k+1) \cdot p_{[k]}$$

Para minimizar F_{med} es necesario minimizar la suma, que tiene por términos los productos

binarios de los elementos de dos sucesiones:

sucesión 1: 1 2 3 ..... n
sucesión 2: p_1 p_2 p_3 p_n

y es sabido que en este caso (cuando todos los valores son positivos) el máximo se obtiene ordenando, para emparejarlas, ambas sucesiones en el mismo orden (por ejemplo, creciente), y el mínimo ordenándolas en orden contrario. Por tanto, si la primera sucesión se ordena en forma decreciente (que es lo mismo que en orden creciente de k, posición de la pieza en la secuencia) la segunda debe ordenarse en orden creciente, es decir, en el orden SPT.

Ejemplo. Sean cinco piezas, con duraciones:

	1	2	3	4	5
t_i	7	4	6	5	3

$p_1 = 7 ; p_2 = 4 ; p_3 = 6 ; p_4 = 5 ; p_5 = 3$

orden SPT: 5 - 2 - 4 - 3 - 1

5 - 2 - 4 - 3 - 1

en dicho orden: $F_1 = 25 ; F_2 = 7 ; F_3 = 18 ; F_4 = 12 ; F_5 = 3$

$$F_{med} = \frac{25 + 7 + 18 + 12 + 3}{5} = 13$$

en cambio en cualquiera de los otros 119 órdenes posibles F_{med} es mayor; por ejemplo, en el orden 1 - 2 - 3 - 4 - 5 tenemos $F_{med} = 16{,}4$

Intuitivamente, en la figura 6.1.4.2, podemos deducir algo semejante pues el mínimo de F_{med}, que conduce al mínimo del área rayada (proporcional al stock en cola, esperando a ser procesado en la máquina) se obtendrá cuando la curva formada por los vértices A, B, C, ..., K, L sea convexa, es decir, cuando cada operación sea de mayor (o igual) duración que la anterior.

Un método alternativo de demostración, que conduce a un procedimiento constructivo muy general, es el siguiente:

1) Sea una secuencia no SPT, es decir existen dos piezas, i y h que ocupan posiciones contiguas en la secuencia, la k y la $k+1$, que no están en orden creciente de duraciones:

$$p_i > p_h$$

2) Se pueden permutar i y h mejorando el valor de F_{med}. Obsérvese que esta permutación sólo altera los valores de F_i y F_h:

Antes	Después
$F_i = t + p_i$ $F_h = t + p_i + p_h$	$F_i' = t + p_h + p_i$ $F_h' = t + p_h$

donde $t = F_{[k-1]}$.

Obsérvese que $F_h = F_i'$ pero que $F_i > F_h'$, por tanto:

$$\Sigma F_k > \Sigma F_k' \text{ y por tanto } F_{med} > F'_{med}$$

3) Cualquier permutación de las piezas puede transformarse en una secuencia SPT mediante una sucesión de permutaciones de piezas contiguas, *mejorando cada vez* F_{med}.

4) Llegados a la secuencia SPT cualquier permutación de piezas contiguas (salvo empates en los valores p_i) empeorará el valor de F_{med}; es decir, ninguna permutación de piezas contiguas mejorará F_{med}.

Desgraciadamente, en muchos casos no podrá obtenerse el valor óptimo de la medida de eficacia utilizando un procedimiento de permutación de dos piezas (contiguas o no) mejorando en todos los pasos dicha medida de eficacia debido a la existencia de mínimos locales. Por ejemplo, en el problema 3/1/F siguiente:

Pieza *i*	Duración *p(i)*	Fecha comp. *d(i)*
a	1	4
b	2	2
c	3	3
Medida de eficacia $T_{med} = \Sigma\ [max\ \{c_i - d_i, 0\}]/3$		

La secuencia óptima es *bac* pero si estamos en *cab* no se puede alcanzar el óptimo mediante permutación de piezas contiguas; ello se debe a que no existe transitividad del orden bajo esta medida de eficacia y relación de vecindad (en *cab* deducimos que *c* domina *a* y que *a* domina *b*, pero ello no establece que *c* domine a *b*). En estas circunstancias el procedimiento de intercambio constituye sólo una buena heurística para mejorar una primera solución obtenida por otro procedimiento. Recientemente se han desarrollado procedimientos (búsqueda tabú, recocido simulado, etc.) para solventar la dificultad en abandonar un óptimo local de los procedimientos de intercambio.

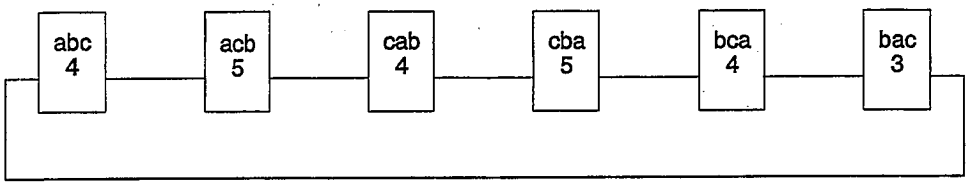


Fig. 6.1.4.3 Encadenamiento de vecindad entre soluciones mediante la operación de permutación de elementos contiguos

La secuencia SPT minimiza también el instante medio de terminación, el tiempo medio de espera y el huelgo medio, así como el número medio de piezas en el taller como se deduce de la fórmula de Little:

$$\frac{N_m(0,F_{\max})}{n} = \frac{F_{med}}{F_{\max}}$$

puesto que n y F_{max} son constantes. Así mismo minimiza $F_{med}(\alpha)$ para $\alpha > 0$

$$F_{med}(\alpha) = \frac{\sum F_i^{\alpha}}{n}$$

6.1.4.2 Secuencia conforme a la fecha de vencimiento (EDD = *earliest due date*)

Consiste en ordenar las piezas de acuerdo con la fecha de vencimiento creciente:

$$d_{[1]} \leq d_{[2]} \leq \ldots\ldots\ldots \leq d_{[n]}$$

que responde a otra regla prudente de taller: "primero lo más urgente".

Puede demostrarse que la secuencia EDD minimiza el huelgo máximo y el retraso máximo.

Teorema de Jackson (1955): En un problema $n/1/L_{max}$ con todas las piezas y máquina disponibles indefinidamente a partir del instante 0 se minimiza el huelgo máximo y el retraso máximo ordenando las piezas según fechas contractuales no decrecientes.

Pueden emplearse como demostración argumentos semejantes a los ya utilizados anteriormente; si las piezas no están en este orden existe un valor de k tal que:

$$d_{[k]} > d_{[k+1]}$$

Sean i y h los nombres de las dos piezas $[k]$ y $[k+1]$. Si permutamos estas piezas los valores correspondientes a las demás permanecen igual, las características que se alteran son las correspondientes a estas piezas. Vamos a demostrar que:

$$\max \{ L , L_i , L_h \}$$

donde L es el máximo del huelgo en las n-2 piezas restantes no aumenta al permutar las piezas i y h. Antes de permutar:

$$L_i = L_{[k]} = t + p_i - d_i$$
$$L_h = L_{[k+1]} = t + p_i + p_h - d_h$$

Después de permutar:

$$L_i' = L'_{[k+1]} = t + p_h + p_i - d_i$$
$$L_h' = L'_{[k]} = t + p_h - d_h$$

Tenemos :

$$L_h > L_i' \text{ ya que } d_i > d_h$$

y

$$L_h > L_h' \text{ ya que } p_i > 0$$

luego:

$$L_h > \max \{ L_i' , L_h' \}$$

y por tanto:

$$\max \{ L , L_i , L_h \} \geq \max \{ L , L_i' , L_h' \}$$

(el signo igual sólo puede darse si el máximo es L). Por otra parte:

$$T_{max} = \max \{ 0 , L_{max} \} \geq \max \{ 0 , L_{max}' \} = T_{max}'$$

6.1.4.3 Secuencia conforme al margen (SFT)

Consiste en ordenar las piezas de acuerdo a valores crecientes del tiempo en exceso disponible sobre la duración de la operación:

$$d_{[1]} - p_{[1]} \leq d_{[2]} - p_{[2]} \leq \ldots\ldots \leq d_{[n]} - p_{[n]}$$

También corresponde a una actitud prudente e intuitiva del taller "primero lo que tiene menos margen", pero el resultado es desconcertante. Puede demostrarse que la secuencia SFT maximiza el mínimo huelgo y el mínimo retraso, por lo que no conduce a nada plenamente deseable (aunque, dadas las circunstancias, maximizar el mínimo huelgo conduce a concentrar todos los huelgos alrededor de un valor medio y por tanto también a dar huelgos máximos y retrasos máximos pequeños).

La demostración es análoga a la anterior, sean i y h dos piezas contiguas tales que:

$$d_i - p_i > d_h - p_h$$

Si las permutamos tendremos:

$$L_i < L_h' \text{ ya que } d_i - p_i > d_h - p_h$$
$$L_i < L_i' \text{ ya que } p_h > 0$$

de donde:

$$L_i < \min \{ L_i', L_h' \}$$

y por tanto:

$$\min \{ I, L_i, L_h \} \leq \min \{ I, L_i', L_h' \}$$

donde I es el mínimo huelgo de las n-2 piezas restantes.

6.1.4.4 Consideración de retrasos

Teorema de Smith (1956): Si en un problema $n/1$ con todas las piezas y máquina disponibles a partir del instante 0 existe una secuencia tal que el máximo retraso es nulo, entonces existe un orden de las piezas con la pieza h en última posición que minimiza el tiempo medio de permanencia (sometido a la condición de que el retraso máximo continúe siendo cero) si y sólo si:

a)
$$d_h \geq \sum_{i=1}^{n} p_i$$

b) $p_h \geq p_i$ para todo i tal que
$$d_i \geq \sum_{k=1}^{n} p_k$$

Este teorema nos permite construir a partir de la secuencia EDD, cuando no se producen retrasos ($T_{max}=0$) otra secuencia, también sin retrasos, con un valor mínimo de F_{med} bajo dicha condición.

6.1.4.5 Medidas ponderadas de eficacia

Si todos los trabajos no son igualmente importantes podremos introducir un coeficiente de ponderación u_i. El tiempo de permanencia medio ponderado será:

$$F_{pon}(u) = \frac{1}{n} \cdot \sum_{i=1}^{n} u_i \cdot F_i$$

aunque es indiferente utilizar

$$F_{med}(u) = \frac{\sum u_i \cdot F_i}{\sum u_i}$$

ya que ambos denominadores son constantes. El orden que intuitivamente conduce al mínimo de $F_{med}(u)$ se obtendría a partir de la substitución en la figura 6.1.4.2 en la escala vertical la anchura de los rectángulos que en todos los rectángulos horizontales es igual (la unidad) por una anchura proporcional a u_i. La línea convexa ABC ... HKL se obtendría ahora ordenando según:

$$\frac{p_{[1]}}{u_{[1]}} \leq \frac{p_{[2]}}{u_{[2]}} \leq \ldots\ldots \leq \frac{p_{[n]}}{u_{[n]}}$$

Puede demostrarse que nuestra intuición es acertada.

6.1.5 Problemas *Flow-Shop* estáticos

En los problemas con rutas regulares o de tipo *flow* es posible numerar las máquinas de forma que las operaciones sucesivas de una pieza se desarrollen en máquinas cuyos números estén en orden creciente. Si una pieza tiene operaciones en todas las máquinas, su operación *k*-ésima tendrá lugar en la máquina número *k*. Por tal motivo podemos modificar ligeramente nuestra notación, de forma que:

$p_{j,i}$ es la duración de la operación de la pieza *i* en la máquina *j*

si $p_{j,i} = 0$ la pieza *no tiene ninguna operación* en la máquina *j*

Por consiguiente en un problema únicamente con rutas *F*, con todas las piezas y máquinas disponibles en el mismo instante, los datos quedarán constituidos por los valores $p_{j,i}$. Obsérvese que entre el caso $p_{j,i} > 0$ y el caso $p_{j,i} = 0$ la diferencia es más cualitativa que cuantitativa.

6.1.5.1 Problemas n/m/F/F_{max}

Johnson enunció en 1954 los dos teoremas o propiedades siguientes (cuya demostración es muy sencilla):

Johnson_1: En un problema *n/m/F*, con todas las piezas y máquinas disponibles

simultáneamente ($r_i = f_j(0) = 0$) y una medida de eficacia regular, basta considerar los programas en que las piezas tienen la misma secuencia en las dos primeras máquinas,

Johnson_2: En un problema n/m/F/F_{max}, con todos las piezas y máquinas disponibles simultáneamente ($r_i = f_j(0) = 0$), basta considerar los programas en que las piezas tienen la misma secuencia en las dos últimas máquinas,

donde $f_j(0)$ es el instante de disponibilidad de la máquina j.

Demostración de Johnson_1

Consideremos la situación de la figura 6.1.5.1, las piezas no están en el mismo orden en las máquinas 1 y 2, por lo que existen por lo menos dos piezas situadas consecutivamente en la máquina 1, la *i* y la *h*, que están dispuestas en orden contrario (primero *h* y luego *i* en la máquina 2. Si permutamos *i* y *h* en la máquina 1, dejando invariadas las restantes piezas en dicha máquina, no perjudicaremos la situación de las piezas en la máquina 2, salvo que *h* tal vez podrá realizarse antes en dicha máquina, y, si reducimos tiempos muertos, también algunas de las piezas situadas tras *h* en la máquina 2, entre ellas *i*. Dichos adelantos pueden repercutir en las máquinas siguientes. Por consiguiente, los tiempos de terminación de las piezas en la última máquina no aumentarán, y en todo caso algunos de ellos, los de *i* y *h* por ejemplo, podrán disminuir, mejorando, en el caso idóneo, el valor de una medida regular de eficacia.

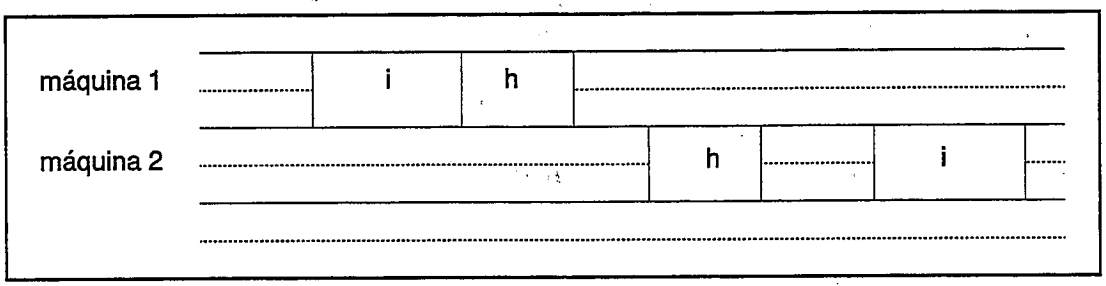


Fig. 6.1.5.1 Justificación del teorema Johnson_1. Si permutamos i *y* h *en la máquina 1 las operaciones en las restantes máquinas no se retrasarán, en todo caso* h *podrá empezar antes en la máquina 2, lo mismo que* i, *y en consecuencia otras operaciones en la máquina 2 y siguientes podrán adelantarse.*

Este argumento puede aplicarse sistemáticamente hasta que todas las piezas en las máquinas 1 y 2 estén en el mismo orden.

Demostración de Johnson_2

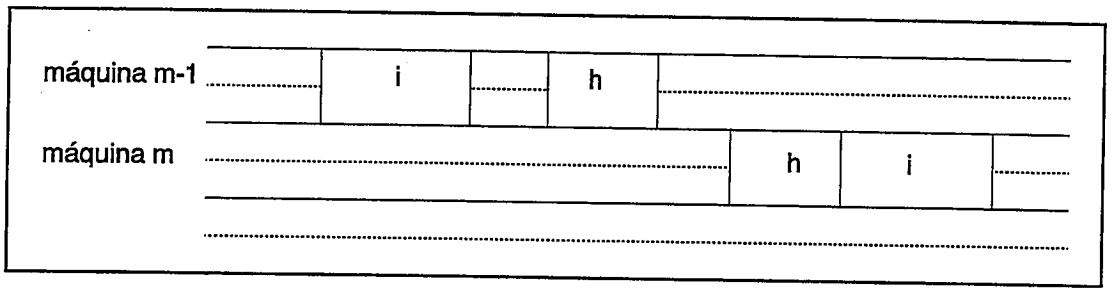


Fig. 6.1.5.2 Justificación del teorema Johnson_2. Si permutamos h *e* i *en la última máquina no se producirá variación. En todo caso* i *podrá comenzar antes de lo que comenzaba* h *en la última máquina.*

Consideremos la situación de la figura 6.1.5.2, las piezas no están en el mismo orden en las máquinas *m-1* y *m*; por tanto, hay dos piezas consecutivas en la máquina *m*, las *h* e *i*, que están en orden contrario en la máquina *m*-1. Si permutamos el orden de las piezas en la máquina *m*, no alteraremos F_{max} salvo si al reducir los tiempos muertos *i* y *h* terminan antes y arrastran al resto de piezas siguientes, en cuyo caso F_{max} disminuirá.

Este argumento puede aplicarse sistemáticamente hasta que todas las piezas en las máquinas *m-1* y *m* estén en el mismo orden.

Lo anterior significa, por ejemplo, que si hay una solución óptima respecto F_{max} que no tiene la misma secuencia en las dos últimas máquinas puede obtenerse a partir de ella una solución con el mismo valor de F_{max}, y por tanto también óptima, que cumpla la condición Johnson_2. Dada la naturaleza de las demostraciones, puede comprenderse que respecto a la medida de eficacia, F_{max}, que es regular, son acumulables.

Johnson_1 es más general, es aplicable a todas las medidas de eficacia regulares; en cambio Johnson_2 se refiere concretamente a F_{max}. Volviendo al esquema de la demostración y llamando c' a los instantes de terminación de las operaciones en la máquina m después de la permutación, y c antes, tendremos:

$$c_i' < c_i \; ; \; c_h' \leq c_i$$

aunque no podemos pronunciarnos sobre la relación entre c_h y c_h', que en muchas circunstancias podrá ser $c_h' > c_h$.

Adecuando, por tanto, la secuencia de la última máquina al orden de la secuencia en la

penúltima no se empeora el valor de c_{max} o F_{max}, pero como un valor c_i puede haber salido perjudicado no podemos generalizar el hecho a todas las medidas de eficacia regulares.

Por consiguiente en los problemas $n/2/F/F_{max}$ y $n/3/F/F_{max}$ existe una solución óptima con la misma secuencia en todas las máquinas (téngase en cuenta que cuando n = 3 la segunda máquina también es la penúltima). Para resolver dichos problemas podemos, por tanto, substituirlos por sus equivalentes $n/2/P/F_{max}$ y $n/3/P/F_{max}$, de acuerdo con nuestra notación.

Unos contraejemplos nos permitirán ver mejor el alcance de los teoremas.

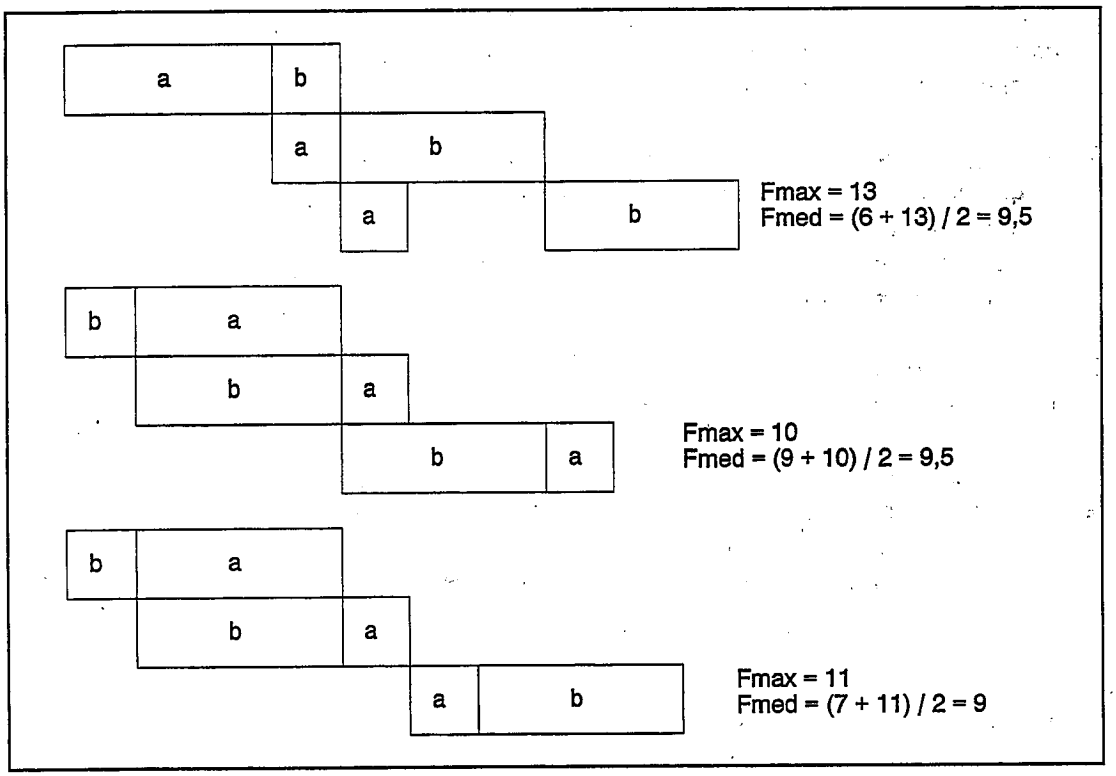


Fig. 6.1.5.3 Problema $411F_{med}$. La solución óptima conserva el orden en las dos primeras máquinas, pues la medida de eficacia es regular, pero lo altera en la tercera. Obsérvese que el diagrama central obtiene el mínimo de F_{max}, conservando el orden en las tres máquinas.

Ejemplo 1. Problema $2/3/F/F_{med}$

Máquina	1	2	3
Pieza *a*	4	1	1
Pieza *b*	1	4	4

Como puede verse en la figura 6.1.5.3 la solución óptima (la tercera) conserva el orden de las piezas en las dos primeras máquinas, ya que F_{med} es una medida regular, pero el orden es diferente en la tercera máquina, es decir, no se conserva el orden en las dos últimas.

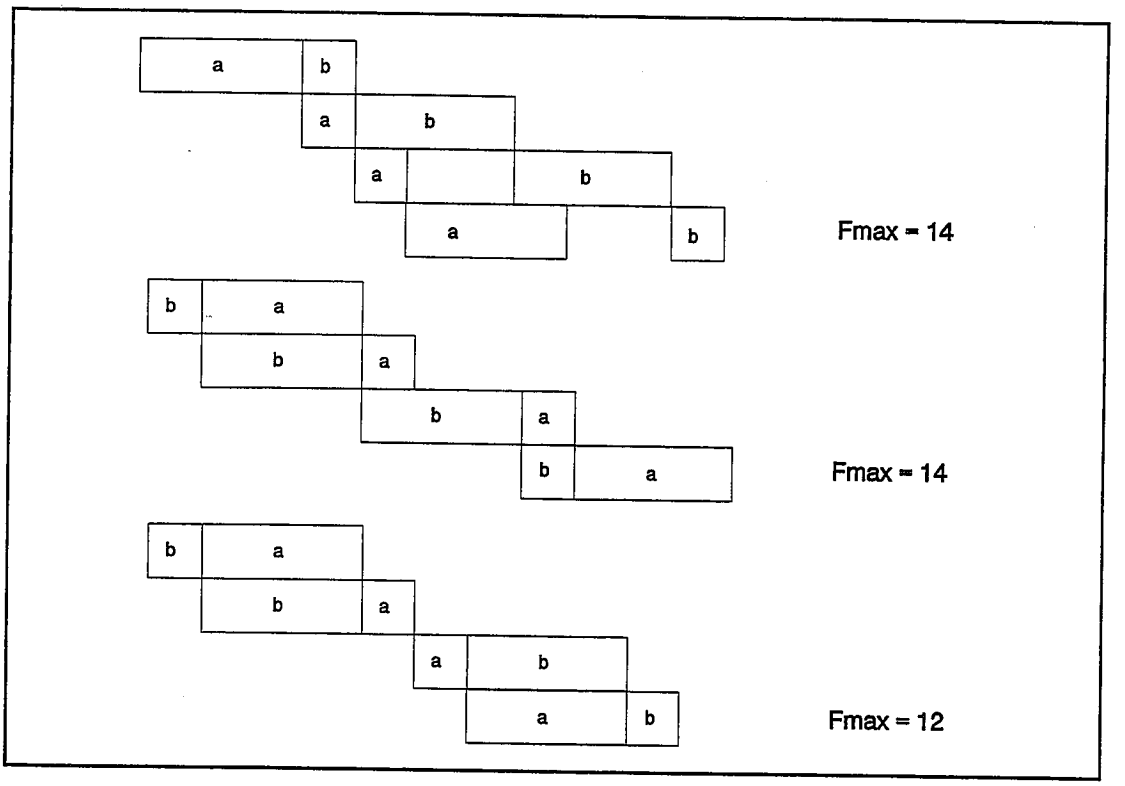


Fig. 6.1.5.4 Problema 4114. La solución óptima utiliza la misma secuencia (2-1) en las dos primeras máquinas, y la misma (1-2) en las dos últimas, pero permuta el orden entre la segunda y tercera máquinas.

Ejemplo 2. Problema 2/4/F/F_{max}

Máquina	1	2	3	4
Pieza *a*	4	1	1	4
Pieza *b*	1	4	4	1

La solución óptima (la tercera de la figura 6.1.5.4) conserva el orden de las piezas entre la primera y segunda máquinas, también entre la tercera y cuarta (penúltima y última respectivamente), pero no entre la segunda y tercera.

6.1.5.2 Problema *n/2/F/F*$_{max}$: Algoritmo de Johnson

De los teoremas de Johnson (realmente solo precisamos uno de ellos) y de las consideraciones anteriores sobre los tiempos muertos, se deducen tres consecuencias inmediatas para los problemas *n/2/F/F*$_{max}$:

1) sólo es necesario considerar permutaciones (las secuencias en la primera y la segunda máquinas pueden ser iguales), en los problemas *n/2/F* con cualquier medida regular de la eficacia,

2) partiendo de la figura 6.1.5.5, llegamos a la conclusión (llamando $p_{1,[s]}$, $p_{2,[s]}$ las duraciones en la primera y segunda máquinas de la pieza que ocupa la posición *s* en la secuencia) de que:

$$F_{max} \geq p_{1,1} + p_{1,2} + p_{1,3} + p_{2,3}$$

n = 3 piezas
m = 2 máquinas

$$F_{max} \geq \sum_{s=1}^{n} p_{1,[s]} + p_{2,[n]}$$

$$F_{max} \geq p_{1,[1]} + \sum_{s=1}^{n} p_{2,[s]}$$

$$F_{max} \geq p_{1,[1]} + p_{2,1} + p_{2,2} + p_{2,3}$$

(en el fondo equivale a decir que el camino crítico es mayor o igual que cualquier camino entre el inicio y el fin de un proyecto, en este caso la realización de las dos piezas). Dado que no conocemos el orden definitivo de las piezas, si adoptamos una actitud prudente podremos establecer las siguientes cotas inferiores de F_{max}:

$$k_1 = \sum_{i=1}^{n} p_{1,i} + \min_i \{p_{2,i}\}$$

$$k_2 = \min_i \{p_{1,i}\} + \sum_{i=1}^{n} p_{2,i}$$

y entonces $F_{max} \geq \max \{ k_1, k_2 \}$

(Estas cotas sugieren el algoritmo de Johnson, que veremos más adelante).

3) los programas semiactivos no tendrán tiempos muertos entre operaciones en la primera máquina; por tanto, podemos limitar nuestra consideración a los programas con tiempos muertos en la segunda máquina solamente.

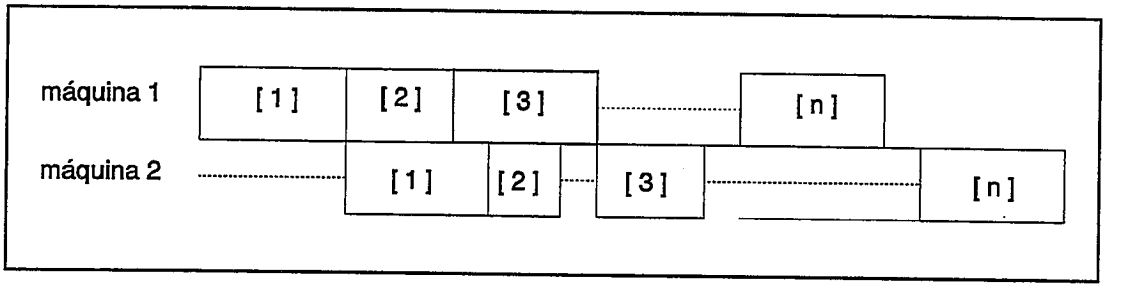


Fig. 6.1.5.5 Esquema para la deducción de las cotas en los problemas n/2/F/F_{max}

Johnson demostró en 1954 que en un problema $n/2/F/F_{max}$ con todas las piezas disponibles simultáneamente en el instante 0, es óptima una secuencia (la misma permutación para ambas máquinas) tal que para todas las parejas de piezas h e i en las que:

$$\min \{ p_{1,h}, p_{2,i} \} < \min \{ p_{1,i}, p_{2,h} \}$$

es decir, el tiempo en la primera máquina de h o el de i en la segunda sea el menor de los cuatro, la pieza h precede a la i en la secuencia. No queda descartado que existan soluciones óptimas que no satisfagan la condición anterior. Es sencillo diseñar un algoritmo que permita ordenar las piezas de forma que cumplan esta relación.

Demostración de la condición de Johnson

Dado que sólo hay dos máquinas utilizaremos la notación:

$$A_i = p_{1,i} ; B_i = p_{2,i}$$

que nos simplificará las deducciones y además coincide con la utilizada en el artículo básico de Johnson. Llamaremos $A_{[k]}$, $B_{[k]}$ a los tiempos en la primera y segunda máquinas de la pieza que ocupa la posición k en la secuencia y X_k al tiempo muerto en la segunda máquina inmediatamente antes de $B_{[k]}$ (recuérdese que en la primera máquina no hay

tiempos muertos, y en todo caso estaría situado detrás de $A_{[n]}$). Podemos escribir:

$$X_1 = A_{[1]}$$

$$X_2 = \max \{ A_{[1]} + A_{[2]} - X_1 - B_{[1]} , 0 \}$$

y en general:

$$X_k = \max \{ \sum_{s=1}^{k} A_{[s]} - \sum_{s=1}^{k-1} X_s - \sum_{s=1}^{k-1} B_{[s]}, 0 \}$$

De otra forma:

$$X_1 = A_{[1]}$$

$$X_1 + X_2 = \max \{ A_{[1]} + A_{[2]} - B_{[1]} , A_{[1]} \}$$

$$X_1 + X_2 + X_3 = \max \{ A_{[1]} + A_{[2]} + A_{[3]} - B_{[1]} - B_{[2]} , X_1 + X_2 \} =$$

$$= \max \{ A_{[1]} + A_{[2]} + A_{[3]} - B_{[1]} - B_{[2]} , A_{[1]} + A_{[2]} - B_{[1]} , A_{[1]} \}$$

Sea en general:

$$Y_k = \sum_{s=1}^{k} A_{[s]} - \sum_{s=1}^{k-1} B_{[s]}$$

y entonces:

$$\sum_{s=1}^{k} X_s = \max \{ Y_1 , Y_2 , \ldots , Y_k \}$$

Si dada una permutación S nos interesamos por $F_{max}(S)$ podremos escribir:

$$F_{\max}(S) = \sum_{k=1}^{n} B_{[k]} + \sum_{k=1}^{n} X_k = \sum_{k=1}^{n} B_{[k]} + \max \{ Y_1 , Y_2 , \ldots , Y_n \}$$

y puesto que el primer término es constante, el mínimo de F_{max} se obtendrá minimizando el máximo de Y_k.

La condición de Johnson se escribirá en esta nueva notación de la siguiente forma, *h* debe preceder a *i* si:.

$$\min \{ A_h , B_i \} < \min \{ A_i , B_h \}$$

Por tanto, una permutación satisfará la condición de Johnson si para todo par de piezas,

h e *i*, tales que *h* precede a *i* (no necesariamente en forma inmediata) se cumple:

$$\min \{ A_h , B_i \} \leq \min \{ A_i , B_h \}$$

En una permutación que no cumpla la condición anterior existirán dos piezas *h*, *i* en la secuencia, la k^a y la $(k+1)^a$, tales que:

$$\min \{ A_h , B_i \} > \min \{ A_i , B_h \}$$

La existencia de dichas piezas contiguas queda garantizada por la transitividad de la relación que demostramos posteriormente (transitividad que exige, para ser total, una condición adicional).

La relación anterior puede escribirse también:

$$\max \{ -A_h , -B_i \} < \max \{ -A_i , -B_h \}$$

Sumando a los cuatro términos la cantidad:

$$\sum_{s=1}^{k+1} A_{[s]} - \sum_{s=1}^{k-1} B_{[s]}$$

que no depende del orden relativo de las piezas *h* e *i* (de cuál ocupa la posición *k* y cuál ocupa la posición $k+1$) tenemos:

$$\max \{ Y'_k , Y'_{k+1} \} < \max \{ Y_k , Y_{k+1} \}$$

en donde las *Y* suponen el orden indicado de las piezas *h* antes de *i*, y las *Y'* el contrario. Puesto que las $n-2$ Y_k restantes no varían tenemos ventaja utilizando el orden que corresponde a *Y'* en lugar del que corresponde a *Y*: la permutación de estas dos piezas no aumenta el valor F_{max} y en algunos casos puede disminuirlo.

Si, adicionalmente, dicha relación es transitiva, además de mostrar la existencia de las dos piezas contiguas que no cumplen la relación si la permutación no la cumple, podremos llegar por permutaciones sucesivas de piezas contiguas a un orden único (salvo los empates cuando la relación se satisfaga con el signo igual), cuyo valor de F_{max} sea mínimo. Nos interesa, por tanto, comprobar si de:

$$\min \{ A_h , B_i \} \leq \min \{ A_i , B_h \} \quad \text{y}$$

$$\min \{ A_i , B_k \} \leq \min \{ A_k , B_i \}$$

se deduce

$$\min \{ A_h , B_k \} \leq \min \{ A_k , B_h \}$$

Se pueden presentar cuatro casos:

Caso 1: $A_h = \min \{ A_h , B_i \} ; A_i = \min \{ A_i , B_k \}$

$A_h \leq B_i, A_i, B_h$ y $A_i \leq B_k, A_k, B_i$

de donde: $A_h \leq B_k, A_k, B_h$

Caso 2: $A_h = \min \{ A_h , B_i \} ; B_k = \min \{ A_i , B_k \}$

$A_h \leq B_i, A_i, B_h$ y $B_k \leq A_h, A_k, B_i$

de donde: $\min \{ A_h , B_k \} \leq \min \{ A_k , B_h \}$

Caso 3: $B_i = \min \{ A_h , B_i \} ; A_i = \min \{ A_i , B_k \}$

$B_i \leq A_h, A_i, B_h$ y $A_i \leq B_k, A_k, B_i$

de donde: $A_i = B_i$

Caso 4: $B_i = \min \{ A_h , B_i \} ; B_k = \min \{ A_i , B_k \}$

$B_i \leq A_h, A_i, B_h$ y $B_k \leq A_h, A_k, B_i$

de donde: $B_k \leq A_h, A_k, B_h$

La relación es transitiva en los casos 1, 2 y 4. El caso 3 es aparentemente contradictorio, pero si $A_i = B_i$ las relaciones de las premisas se cumplen con el signo igual, y por tanto la posición de la pieza *i* es indiferente (puede situarse tanto antes de las *h* y *k* como después) sin alterar F_{max}. Por tanto, no induce ningún orden respecto a *h* y *k*, pero si posiblemente respecto a otras piezas no consideradas. Para solventar esta situación basta impedir que A_i sea igual a B_i introduciendo una perturbación en el problema substituyendo, por ejemplo, B_i por $B_i + \alpha$ donde α es un valor arbitrario muy pequeño. Ello, además de impedir la aparición del caso 3 y garantizar la transitividad, implica situar siempre *i* delante de *h* y *j* cuando se dan las relaciones indicadas (el valor α se utiliza únicamente para la obtención de la permutación, no en el cálculo de F_{max}).

Un procedimiento alternativo consiste en ampliar la condición de Johnson. Dadas dos piezas contiguas *h* e *i* de una permutación invertiremos su orden si

$$\min \{ A_h , B_i \} > \min \{ A_i , B_h \}$$

o bien si

$$\min \{ A_h , B_i \} = \min \{ A_i , B_h \}$$

y

$$A_h - B_h > A_i - B_i$$

llegando después de un número finito de inversiones de orden de piezas contiguas, sin aumentar el valor F_{max}, a una permutación que satisface la condición de Johnson y que es óptima.

6.1.5.2.1 Algoritmo (de ordenación) de Johnson

Regresando a nuestra notación inicial podemos describir el siguiente algoritmo de ordenación:

1) se busca la menor de las duraciones de las piezas no ordenadas todavía, $p_{j,i}$; en caso de empate se toma una cualquiera de ellas,

2) si la duración es una de la primera máquina ($j=1$) la pieza se coloca la primera (de las no ordenadas todavía), si es de la segunda máquinas ($j=2$) la pieza se coloca la última (de las no ordenadas todavía),

3) se considera ordenada ya la pieza i; si quedan piezas por ordenar se vuelve al punto (1).

La permutación obtenida satisface la condición de Johnson (aunque pueden existir permutaciones que satisfagan la condición de Johnson que no puedan obtenerse mediante el algoritmo indicado).

Ejemplo

	Máquinas	
Piezas	M_1	M_2
a	5	3
b	2	7
c	9	1
d	0	6
e	8	4

Calculemos primero las cotas:

$$k_1 = (5 + 2 + 9 + 0 + 8) + \min \{ 3 , 7, 1 , 6 , 4 \} = 24 + 1 = 25$$

$$k_2 = \min \{ 5 , 2 , 9 , 0 , 8 \} + (3 + 7 + 1 + 6 + 4) = 0 + 21 = 21$$

por tanto:

$$F_{max} \geq 25$$

Apliquemos ahora el algoritmo de Johnson. La duración más pequeña es 0, que corresponde a la pieza *d* y a la primera máquina; la pieza *d*, por tanto, se sitúa la primera:

	Máquinas		
Piezas	M_1	M_2	Ordenada?
d	0	6	sí
a	5	3	
b	2	7	
c	9	1	
e	8	4	

A continuación la menor duración es 1, que corresponde a la pieza *c* y a la segunda máquina; por tanto, la pieza *c* pasa a la última posición:

	Máquinas		
Piezas	M_1	M_2	Ordenada?
d	0	6	sí
a	5	3	
b	2	7	
e	8	4	
c	9	1	sí

La nueva menor duración es 2, que corresponde a la pieza *b* y a la primera máquina, por lo que *b* pasa a la cabeza de las piezas no ordenadas (es decir, a la segunda posición):

	Máquinas		
Piezas	M_1	M_2	Ordenada?
d	0	6	sí
b	2	7	sí
a	5	3	
e	8	4	
c	9	1	sí

Ahora el mínimo es 3, que corresponde a la pieza *a* y a la segunda máquina; por tanto, la

pieza *a* pasa al último lugar de las no ordenadas (es decir, al 4° lugar):

	Máquinas		
Piezas	M_1	M_2	Ordenada?
d	0	6	sí
b	2	7	sí
e	8	4	
a	5	3	sí
c	9	1	sí

Y ya no hace falta proseguir más, puesto que la única pieza sin ordenar, la *e*, sólo puede quedar situada en la misma posición en que se halla ya.

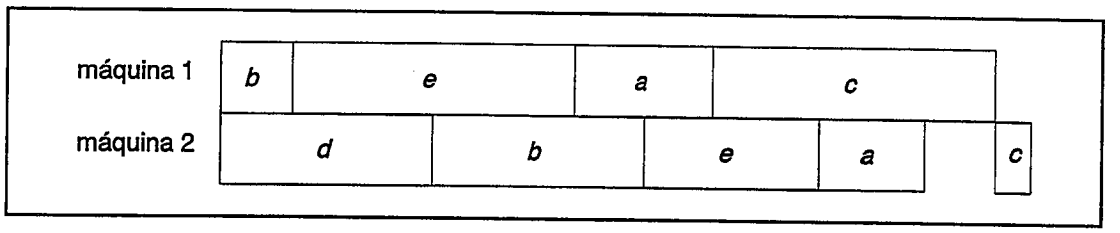


Fig. 6.1.5.6 Solución óptima del problema 5/2/F/F_{max}

El valor de F_{max} correspondiente a este programa (la misma secuencia en las dos máquinas) puede obtenerse mediante el diagrama de Gantt (Fig. 6.1.5.6), o bien en forma tabular. Vamos a describir la forma de obtener esta última. Llamemos $f_j(k)$ el instante en que la máquina *j* queda libre después de haber realizado la *k*-ésima operación, o bien en el caso *F*, para tener en cuenta las duraciones nulas, después de haber "tratado" la pieza que ocupa el lugar *k* en la secuencia. La existencia de duraciones nulas en la máquina 2 implica algunas modificaciones al procedimiento. Inicialmente:

$$f_1(0) = f_2(0) = 0$$

y en general, para la primera máquina, en la que no hay ni tiempos muertos ni esperas (de la máquina) ya que todas las piezas están disponibles en el instante 0, tendremos:

$$f_1(k) = f_1(k-1) + p_{1,[k]}$$

En la segunda máquina para iniciar una operación dependemos de dos condiciones:

- que haya terminado la operación anterior en la máquina,
- que haya terminado la operación anterior de la pieza,

por consiguiente, la operación k podrá iniciarse en el instante:

$$\min \{ f_2(k-1) , f_1(k) \}$$

expresión válida incluso para $k=1$, y en consecuencia:

$$f_2(k) = \min \{ f_2(k-1) , f_1(k) \} + p_{2,[k]} \qquad (\text{si } p_{2,[k]} > 0)$$

Aplicándolo a nuestro caso:

k	1	2	3	4	5
Pieza	*d*	*b*	*e*	*a*	*c*
M_1	0	2	10	15	24
M_2	6	13	17	20	25

Por consiguiente $F_{max}=25$, valor idéntico al de la cota obtenida. Obsérvese que, en otro orden, como el alfabético *a - b - c - d - e*, la ocupación del taller sería mayor, $F_{max}=28$.

k	1	2	3	4	5
Pieza	*a*	*b*	*c*	*d*	*e*
M_1	5	7	16	15	24
M_2	8	15	17	23	28

El cálculo tabular indicado es formalmente equivalente a la determinación del camino crítico en el grafo asociado al problema, una vez se han resuelto las ligaduras disyuntivas estableciendo una permutación concreta de las piezas (figuras 6.1.5.7 (a) y (b)).

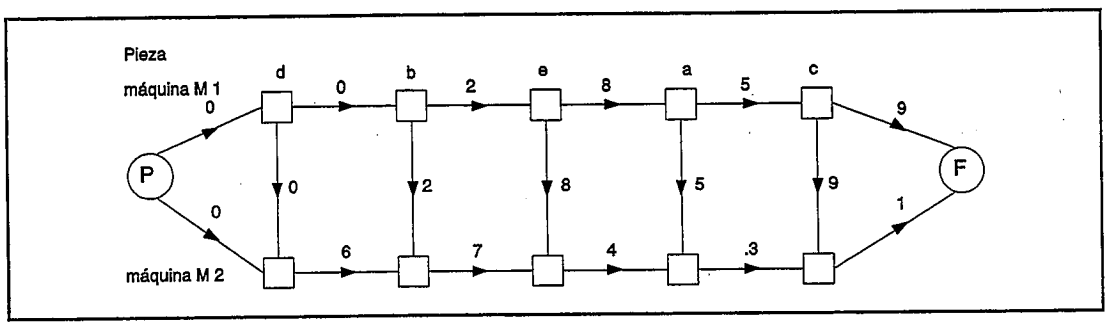


Fig. 6.1.5.7 (a) Grafo asociado al cálculo de F_{max} en el problema 5/2/F/F_{max}, orden óptimo

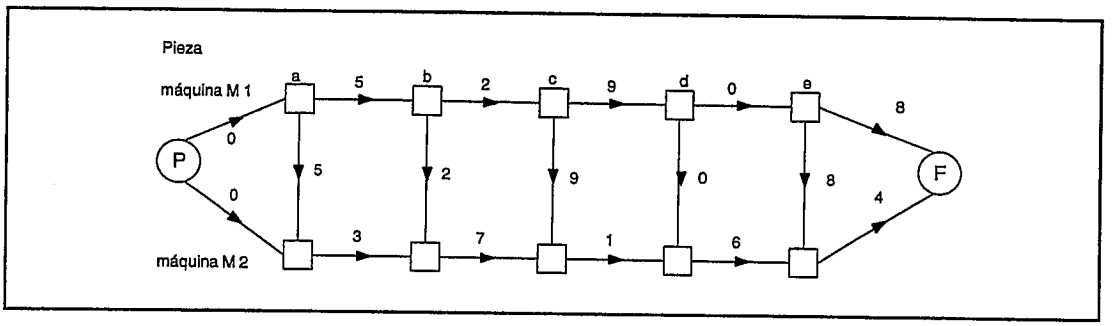


Fig. 6.1.5.7 (b) Grafo asociado al cálculo de F_{max} en el problema 5/2/F/F_{max}, orden a-b-c-d-e

6.1.5.3 Problema *n/2/F/F_{med}*

El algoritmo de Johnson no minimiza F_{med}, al contrario, tiende a dar valores elevados. Estudiemos inicialmente el caso $n=2$ para ver las dificultades, utilizando la notación de Johnson.

En el orden 1-2:

$$c_1 = X_1 + B_1 = A_1 + B_1$$
$$c_2 = X_1 + B_1 + X_2 + B_2 = B_1 + B_2 + \max\{A_1 + A_2 - B_1, A_1\} =$$
$$= A_1 + B_2 + \max\{A_2, B_1\}$$

$$2 \cdot F_{med} = 2.A_1 + B_1 + B_2 + \max\{A_2, B_1\}$$

en el orden 2-1:

$$2 \cdot F_{med}' = 2.A_2 + B_1 + B_2 + \max\{A_1, B_2\}$$

por tanto $F_{med} < F_{med}'$ si:

$$2.A_2 - 2.A_1 + \max\{A_1, B_2\} - \max\{A_2, B_1\} > 0$$

Para dos piezas no habrá dificultad en calcular la expresión, pero sería mejor disponer de un método sencillo.

No basta $A_2 > A_1$, ni $B_2 > B_1$, ni $(A_2+B_2) > (A_1+B_1)$, salvo si sistemáticamente $A_2 \geq A_1$ y $B_2 \geq B_1$ (condición suficiente pero no necesaria).

Para calcular el mínimo de F_{med} basta:

1) considerar permutaciones (medida regular),

2) considerar programas compactos en la primera máquina (sin tiempos muertos),
3) adoptar las definiciones anteriores de X_k e Y_i,

$$F_{[k]} = \sum_{s=1}^{k} B_{[s]} + \sum_{s=1}^{k} X_s = \sum_{s=1}^{k} B_{[s]} + \max \{ Y_1, \ldots, Y_k \}$$

$$F_{med} = \frac{1}{n} \cdot [\sum_{k=1}^{n} \sum_{s=1}^{k} B_{[s]} + \sum_{k=1}^{n} \max \{ Y_1, \ldots, Y_k \}]$$

utilizando esta expresión puede demostrarse que si:

$$A_h \leq A_i \text{ y } B_h \leq B_i$$

existe una permutación óptima en la que *h* precede a *i* pero esto no conduce, en general, a una solución del problema puesto que esta relación no genera un orden total en el conjunto de las piezas.

Ignall & Schrage han aplicado a este problema el método de exploración dirigida (*branch-and-bound*). Han observado que el problema admite un procedimiento de separación con relación arborescente, cada vértice queda definido por la secuencia inicial de *s* de las *n* piezas, faltando por secuenciar las *n-s* restantes. La cota inferior de un vértice viene dada por:

$$SF(s) + \max \{ CFA(n\text{-}s) , CFB(n\text{-}s) \}$$

donde *SF(s)* es la suma de los tiempos de permanencia de las *s* piezas ya secuenciadas, *CFA(n-s)* es la suma de tiempos de permanencia de las *n-s* restantes en el supuesto de que *A* domina a *B* ($A_i > B_i$ para todas las *n-s* piezas), y las piezas se colocan ordenadas en *A* creciente, *CFB(n-s)* tiene el mismo significado pero suponiendo que dominen las *B* (y por tanto se coloquen las piezas restantes en orden de *B* creciente).

6.1.5.4 Problema $n/3/F/F_{max}$

A partir de los teoremas de Johnson sabemos que en este caso, *con todas las piezas disponibles en el instante inicial*, $r_i=0$, basta considerar programas en los que las tres máquinas utilicen la misma secuencia; por tanto, la solución buscada será una de las *n!* permutaciones posibles con las *n* piezas. Johnson generalizó su algoritmo al caso $n=3$ en el que se cumple:

$$\min_h \{ p_{1,h} \} \geq \max_i \{ p_{2,i} \}$$

o bien

$$\max_h \{p_{2,h}\} \geq \min_i \{p_{3,i}\}$$

es decir, cuando la segunda máquina tiene, cuantitativamente, poca importancia frente a una de las extremas. En este caso se puede crear un problema $n/2/F/F_{max}$ ficticio, con los valores de las duraciones:

$$p_{1,i}' = p_{1,i} + p_{2,i}$$

$$p_{2,i}' = p_{2,i} + p_{3,i}$$

y la permutación óptima para el problema ficticio también lo es para el original.

Máquina	1	2	3
pieza *a*	5	3	8
pieza *b*	7	4	3
pieza *c*	6	5	4
pieza *d*	8	2	6
pieza *e*	9	1	5

Fig. 6.1.5.8 Valores de $p_{j,i}$ del problema original

En el caso de la tabla de la figura 6.1.5.8 se cumple la condición, pues:

$$\min \{ p_{1,i} \} = 5 \quad \max \{ p_{2,i} \} = 5$$

Aplicando el algoritmo de Johnson al problema ficticio (cuyos datos están en la tabla de la figura 6.1.5.9) obtenemos la siguiente permutación:

a - c - d - b - e

Máquina	1	2
pieza *a*	8	11
pieza *b*	11	7
pieza *c*	11	9
pieza *d*	10	8
pieza *e*	10	6

Fig. 6.1.5.9 Valores $p_{j,i}'$ del problema ficticio

	Piezas				
	a	*c*	*d*	*b*	*e*
Máquina 1	8	19	29	40	50
Máquina 2	19	28	37	47	56

Fig. 6.1.5.10 Cálculo de la ocupación de máquinas en el problema ficticio

Para calcular F_{max}, utilizaremos como en el caso $n=2$ los valores $f_j(k)$, partiendo de:

$$f_1(0) = f_2(0) = f_3(0) = 0$$

y haciendo:

$$f_1(k) = f_1(k\text{-}1) + p_{1,[k]}$$

$$f_2(k) = \min \{ f_2(k\text{-}1) , f_1(k) \} + p_{2,[k]} \qquad (\text{si } p_{2,[k]} > 0)$$

$$f_3(k) = \min \{ f_3(k\text{-}1) , f_2(k) \} + p_{3,[k]} \qquad (\text{si } p_{3,[k]} > 0)$$

con lo que F_{max} resulta ser 41 (figura 6.1.5.11).

	Piezas				
	a	*c*	*d*	*b*	*e*
Máquina 1	5	11	19	26	35
Máquina 2	8	16	21	30	36
Máquina 3	16	20	27	33	41

Fig. 6.1.5.11 Cálculo de la ocupación de máquinas en el problema original

Para el resto de casos no existen algoritmos exactos enteramente satisfactorios, por lo que se debe recurrir en general a los heurísticos. Los algoritmos que se basan en la programación lineal mixta permiten tratar únicamente problemas de reducido tamaño con gran consumo de tiempo, lo mismo que los basados en la exploración implícita de todas las soluciones (métodos de *branch-and-bound*) como el de Lomnicki que describiremos más adelante. Incidentalmente, la determinación de los valores óptimos de F_{max} en los problemas que siguen han sido realizada mediante el método de Lomnicki.

6.1.5.4.1 Cotas de F_{max} en un problema $n/3/F/F_{max}$

Para juzgar de la bondad de un algoritmo heurístico conviene disponer de una buena cota del valor buscado. La generalización de las cotas halladas anteriormente para $n=2$ nos conduce a los tres valores siguientes:

$$k_1 = \sum_{i=1}^{n} p_{1,i} + \min_{i} \{p_{2,i} + p_{3,i}\}$$

$$k_2 = \sum_{i=1}^{n} p_{2,i} + \min_{h,i} \{p_{1,i} + p_{3,i}\}$$

$$k_3 = \sum_{i=1}^{n} p_{3,i} + \min_{i} \{p_{1,i} + p_{2,i}\}$$

en la que la expresión 2, a diferencia de la 1 o la 3, la suma a minimizar exige la condición de que las piezas h e i puedan ser distintas, ya que la h equivale a la primera de la secuencia y la i a la última. Este razonamiento indica que podemos obligar a que h e i sean forzosamente distintas. Consecuentemente:

$$F_{max} \geq \max \{ k_1 , k_2 , k_3 \}$$

Una forma alternativa de interpretar las cotas consiste en la utilización del grafo asociado al problema (figura 6.1.5.12). Una vez establecida la secuencia o permutación de las n piezas, F_{max} será el valor del camino crítico de dicho grafo, y por tanto igual o superior a la longitud de cualquier camino entre P y F. Consideremos los tres caminos siguientes:

P	*a*	*g*	*b*	*d*	*f*	*F*
P	*a*	*c*	*h*	*d*	*f*	*F*
P	*a*	*c*	*e*	*i*	*f*	*F*

La longitud del primero es:

$$\sum_{k=1}^{n} p_{1,[k]} + p_{2,[n]} + p_{3,[n]}$$

la del segundo:

$$p_{1,[1]} + \sum_{k=1}^{n} p_{2,[k]} + p_{3,[n]}$$

y la del tercero:

$$p_{1,[1]} + p_{2,[1]} + \sum_{k=1}^{n} p_{3,[k]}$$

Puede comprobarse que las expresiones indicadas para k_1, k_2 y k_3 son el resultado de substituir en las anteriores los valores $p_{1,[1]}$, $p_{2,[1]}$, $p_{2[n]}$ y $p_{3,[n]}$ adecuados para garantizar que cualquiera que sea la permutación utilizada los valores de k_1, k_2 y k_3 constituyan una cota.

Al aplicar estas expresiones al problema anterior obtenemos:

$$k_1 = 35 + (1 + 5) = 41$$
$$k_2 = 15 + (5 + 3) = 23$$
$$k_3 = 26 + (5 + 3) = 34$$

por tanto $F_{max} \geq 41$, lo que nos confirma en que la solución hallada es óptima. Lamentablemente en algunos casos la cota puede estar alejada del valor mínimo buscado. En el problema 1987, cuyos datos se hallan en la tabla de la figura 6.1.5.13, la cota obtenida, 39, es inferior al valor óptimo 40 (existen varias soluciones óptimas).

$$k_1 = 30 + (4 + 4) = 38$$
$$k_2 = 30 + (1 + 3) = 34$$
$$k_3 = 30 + (7 + 2) = 39$$
$$F_{max} \geq 39$$

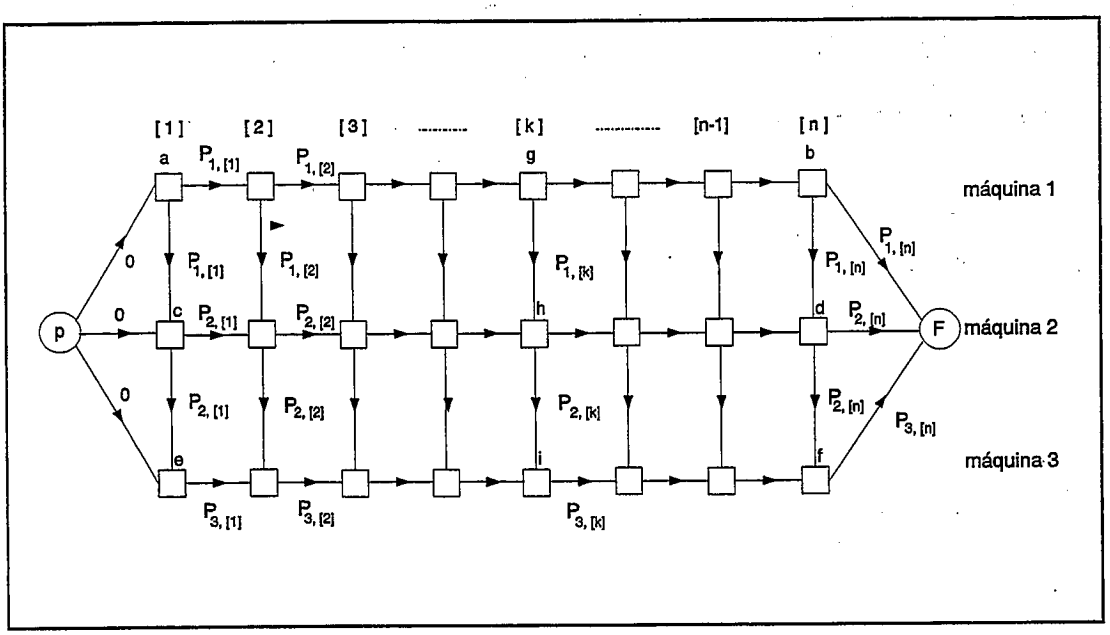


Fig. 6.1.5.12 Grafo asociado a un problema n/3/F/F_{max} para la determinación de cotas

	Máquinas		
	1	2	3
Pieza *a*	1	9	8
Pieza *b*	7	2	9
Pieza *c*	7	11	3
Pieza *d*	10	4	4
Pieza *e*	5	4	6

Fig. 6.1.5.13 Datos del problema 1987

	Piezas				
	a	*b*	*e*	*c*	*d*
Máquina 1	1	8	13	20	30
Máquina 2	10	12	17	31	35
Máquina 3	18	27	33	36	40

Fig. 6.1.5.14 Solución óptima del problema 1987

En el problema de Ignall-Schrage (particularmente difícil de resolver) cuyos datos aparecen en la tabla de la figura 6.1.5.15, la cota obtenida es 59; sin embargo, la solución óptima tiene $F_{max} = 66$.

$$k_1 = 57 + (1 + 1) = 59$$
$$k_2 = 56 + (1 + 1) = 58$$
$$k_3 = 55 + (1 + 2) = 58$$
$$F_{max} \geq 59$$

	Máquinas		
	1	2	3
Pieza *a*	1	2	9
Pieza *b*	5	9	7
Pieza *c*	7	6	8
Pieza *d*	8	9	9
Pieza *e*	3	2	3
Pieza *f*	7	10	4
Pieza *g*	9	7	7
Pieza *h*	8	9	4
Pieza *i*	6	1	3
Pieza *j*	3	1	1

Fig. 6.1.5.15 Datos del problema Ignall-Schrage

Puede obtenerse una cota mejor (aunque alejada del valor 66) a partir de la pieza *d* que tiene la mayor suma de valores $p_{j,i}$, 26. En efecto, suponiendo la ausencia de tiempos muertos, a este valor deberíamos añadir las duraciones $p_{1,i}$ de las piezas que vayan antes de la *d* en la permutación, y las duraciones $p_{3,i}$ de las que vayan después, para obtener F_{max}. Poniendo antes de *d* aquellas piezas en las que $p_{1,i} < p_{3,i}$, y después aquéllas en que $p_{1,i} > p_{3,i}$ (cuando $p_{1,i} = p_{3,i}$, como en *e*, es indiferente) tendremos:

$$h_d = 26 + 1 + 5 + 7 + 3 + 4 + 7 + 4 + 3 + 1 = 61$$

por tanto $F_{max} \geq 61$. Podemos calcular estas cotas *transversales* para todas las piezas, pero ninguna supera el valor 61. La expresión general de dichas cotas será

$$h_i = \sum_{j=1}^{3} p_{j,i} + \sum_{s \neq i}^{n} \min \{p_{1,s}, p_{3,s}\}$$

que se basan en el camino *P - a - g - h - i - f - F* del grafo de la figura 6.1.5.12.

	Pieza									
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
Máquina 1	1	6	13	21	24	31	40	48	54	57
Máquina 2	3	15	21	30	32	42	49	58	59	60
Máquina 3	12	22	30	39	42	46	56	62	65	66

Fig. 6.1.5.16 Solución óptima del problema de Ignall-Schrage (F_{max} = 66)

6.1.5.4.2 Heurísticas para los problemas *n/3/F/F*$_{max}$

Se han propuesto muchísimas heurísticas para la resolución aproximada de estos problemas, de las que vamos a describir algunas, las más sencillas. Wagner y Giglio aplicaron la técnica de calcular

$$p_{1,i}' = p_{1,i} + p_{2,i}$$
$$p_{2,i}' = p_{2,i} + p_{3,i}$$

y aplicar al problema dos máquinas ficticio, con duraciones *p'*, el algoritmo de Johnson, aunque *no se cumpliese la condición* que hace que este procedimiento conduzca al óptimo. Generaron 20 problemas con $n=6$, con duraciones aleatorias, pero sin cumplir la condición, y en 9 casos su método obtuvo el óptimo, y en 8 de los restantes bastaba intercambiar dos piezas adyacentes para hallarlo. El valor medio de F_{max} obtenido era 131,7 mientras que en el óptimo habría sido 127,9 (diferencia del orden del 3 %). En la misma

línea Campbell, Dudeck y Smith proponen calcular para $K = 1,2$ los valores:

$$T_{1,i}(K) = \sum_{j=1}^{K} p_{j,i}$$

$$T_{2,i}(K) = \sum_{j=m+1-K}^{m} p_{j,i}$$

aplicar a las dos T obtenidas para cada K el algoritmo de ordenación de Johnson, determinar F_{max} y tomar de ambas permutaciones la que conduzca al mejor valor.

En las tablas adjuntas (figuras 6.1.5.17 a 23) se recogen los resultados obtenidos al aplicar el procedimiento a los problemas 1987 e Ignall-Schrage. En el primero no aparece ninguna dificultad especial, aunque las soluciones obtenidas no son demasiado buenas. En el segundo se produce una gran ambigüedad en la colocación de la pieza *e*, para la que se produce igualdad de ambos valores de T, por lo que dicha pieza podría situarse en cualquier posición entre la 2ª y la 9ª (para $K=1$) y entre la 2ª y la 8ª ($K=2$). En ambos casos la hemos situado la segunda. La solución para $K=1$ es bastante buena. (Como puede advertirse en la figura 6.1.5.22, la diferencia respecto al óptimo en la figura 6.1.5.16 no estriba en la colocación de *e* sino en la de la pareja *f* y *g*).

	$K = 1$		$K = 2$	
	$T_{1,i}$	$T_{2,i}$	$T_{1,i}$	$T_{2,i}$
Pieza *a*	1	8	10	17
Pieza *b*	7	9	9	11
Pieza *c*	7	3	18	14
Pieza *d*	10	4	14	8
Pieza *e*	5	6	9	10

Fig. 6.1.5.17 Valores de $T_{j,i}(K)$ para el problema 1987

$K=1$	Piezas				
	a	*e*	*b*	*d*	*c*
Máquina 1	1	6	13	23	30
Máquina 2	10	14	16	27	41
Máquina 3	18	24	33	37	44

Fig. 6.1.5.18 Valores $f_j(k)$ para la permutación con $K=1$ en el problema 1987 ($F_{max}=44$)

$K=2$	Piezas				
	b	*e*	*a*	*c*	*d*
Máquina 1	7	12	13	20	30
Máquina 2	9	16	25	36	40
Máquina 3	18	24	33	39	44

Fig. 6.1.5.19 Valores $f_j(k)$ para la permutación con $K=2$ en el problema 1987 ($F_{max}=44$)

	$K=1$		$K=2$	
	$T_{1,i}$	$T_{2,i}$	$T_{1,i}$	$T_{2,i}$
Pieza *a*	1	9	3	11
Pieza *b*	5	7	14	16
Pieza *c*	7	8	13	14
Pieza *d*	8	9	17	18
Pieza *e*	3	3	5	5
Pieza *f*	7	4	17	14
Pieza *g*	9	7	16	14
Pieza *h*	8	4	17	13
Pieza *i*	6	3	7	4
Pieza *j*	3	1	4	2

Fig. 6.1.5.20 Valores de $T_{j,i}(K)$ para el problema Ignall-Schrage

$K=1$	Pieza									
	a	*e*	*b*	*c*	*d*	*g*	*f*	*h*	*i*	*j*
Máquina 1	1	4	9	16	24	33	40	48	54	57
Máquina 2	3	6	18	24	33	40	50	59	60	61
Máquina 3	12	15	25	33	42	49	54	63	66	67

Fig. 6.1.5.21 Valores $f_j(k)$ para la permutación con $K=1$ en el problema Ignall-Schrage ($F_{max}=67$)

$K=1$	Pieza									
	a	*b*	*c*	*d*	*e*	*g*	*f*	*h*	*i*	*j*
Máquina 1	1	6	13	21	24	33	40	48	54	57
Máquina 2	3	15	21	30	32	40	50	59	60	61
Máquina 3	12	22	30	39	42	49	54	63	66	67

Fig. 6.1.5.22 Valores $f_j(k)$ para la permutación con $K=1$ en el problema Ignall-Schrage situando e en 5º lugar ($F_{max}=67$)

K = 2	Pieza									
	a	*e*	*c*	*b*	*d*	*f*	*g*	*h*	*i*	*j*
Máquina 1	1	4	11	16	24	31	40	48	54	57
Máquina 2	3	5	17	26	35	45	52	61	62	63
Máquina 3	12	15	25	33	44	49	56	65	68	69

Fig. 6.1.5.23 Valores $f_j(k)$ para la permutación con $K=2$ en el problema Ignall-Schrage ($F_{max}=69$)

Heurísticas de Palmer y de los trapecios

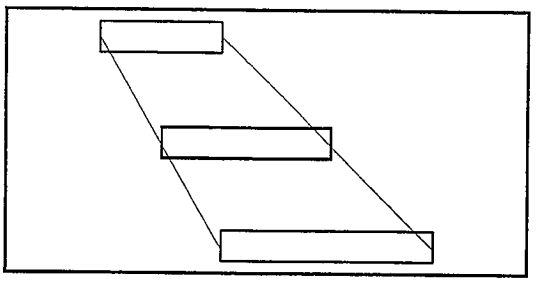


Fig. 6.1.5.24 "Trapecio" ficticio que substituye a las tres barras del diagrama de Gantt según la concepción de Palmer

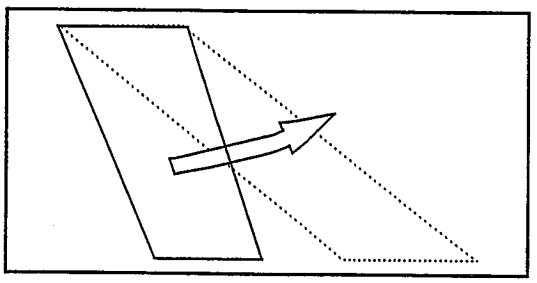


Fig. 6.1.5.25 Sentido de la deformación del trapecio a partir de la posición bloqueada

Palmer considera (*Op. Res. Quart.*, vol. 16, nº 1, March, 1965) la colocación de las piezas en el diagrama de Gantt como la construcción de un rompecabezas en donde las piezas a colocar son unos trapecios deformables (figura 6.1.5.24). La deformación es posible a partir de una posición ideal, en sentido contrario a las agujas del reloj (figura 6.1.5.25); en el sentido de las agujas del reloj, y a partir de la posición ideal, el trapecio está rígido o bloqueado. Para definir la pendiente de los lados oblicuos del trapecio en la posición ideal utiliza los valores $S_{1,i}$ y $S_{2,i}$ (que de hecho definen en forma relativa la posición del centro

de gravedad de los extremos izquierdos y derechos respectivamente de los rectángulos que en el diagrama de Gantt representan las operaciones):

$$S_{1,i} = 2.p_{1,i} + p_{2,i}$$

$$S_{2,i} = p_{2,i} + 2.p_{3,i}$$

La posición representada en (a), con los trapecios bloqueados, deja espacio muerto entre los trapecios. Palmer prefiere colocarlos en orden contrario, tal como indica (b).

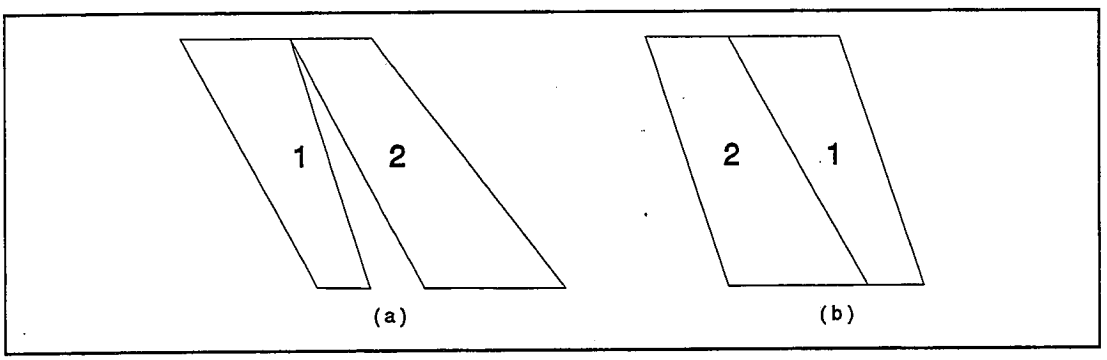


Fig. 6.1.5.26 Colocación idónea de los trapecios según Palmer

La regla de Palmer ordena las piezas en orden creciente de:

$$S_{3,i} = S_{1,i} - S_{2,i}$$

atendiendo a las razones aducidas en la figura 6.1.5.26.

Nosotros (R. Companys, *Métodos heurísticos en la resolución del problema del taller mecánico*, Estudios Empresariales, vol. 5, nº 66/2, junio 1966; *Introducción a los métodos de exploración dirigida*, Cuadernos de estadística Aplicada e Investigación operativa, vol. V, fasc. 3, 1968; *Problema taller mecánico*, CPDA, 1972) hemos propuesto ordenar las piezas utilizando el algoritmo de Johnson con los valores $S_{1,i}$ y $S_{2,i}$ (método trapecios), que entre otras propiedades conduciría al orden óptimo de colocación de trapecios deformables tal como los postula Palmer (desgraciadamente queremos colocar piezas y no trapecios).

Puesto que los dos procedimientos son muy sencillos, y se basan en los mismos juegos de índices, pueden emplearse ambos y adoptar como solución la permutación que ofrezca mejores resultados. Normalmente al ordenar las piezas, tanto con una regla como con la otra, pueden producirse empates cuya resolución influye en el resultado (cosa que no ocurría en el caso $n/2/F/F_{max}$), por ello adoptamos como reglas suplementarias, en caso de

empate:

- utilizar la regla alternativa, es decir con Palmer de primer criterio, trapecios de segundo, y viceveresa,

- si subsiste el empate, dar prioridad a la pieza con $p_{1,i}$ menor,

La aplicación de estas heurísticas al problema 1987 nos proporciona dos soluciones óptimas (la primera es la que hemos indicado anteriormente). En el problema Ignall-Schrage no se produce ambigüedad en la colocación de la pieza *e* en el método de Palmer, y utilizando dicha colocación para resolver la existente en el trapecios obtenemos una nueva solución con $F_{max}=67$; dicha solución se convertiría en óptima permutando las dos piezas contiguas *g* y *f*.

	Máquinas					
	1	2	3	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
Pieza *a*	1	9	8	11	25	-14
Pieza *b*	7	2	9	16	20	-4
Pieza *c*	7	11	3	25	17	8
Pieza *d*	10	4	4	24	12	12
Pieza *e*	5	4	6	14	16	-2
Orden Palmer	*a - b - e - c - d*			F_{max} = 40		
Orden trapecios	*a - e - b - c - d*			F_{max} = 40		

Fig. 6.1.5.27 Aplicación de Palmer y trapecios al problema 1987

	Piezas				
	a	*e*	*b*	*c*	*d*
Máquina 1	1	6	13	20	30
Máquina 2	10	14	16	31	35
Máquina 3	18	24	33	36	40

Fig. 6.1.5.28 Valores $f_i(k)$ para la permutación trapecios en el problema 1987 ($F_{max}=40$)

	Máquinas					
	1	2	3	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
Pieza *a*	1	2	9	4	20	-18
Pieza *b*	5	9	7	19	23	-4
Pieza *c*	7	6	8	20	22	-2
Pieza *d*	8	9	9	25	27	-2
Pieza *e*	3	2	3	8	8	0
Pieza *f*	7	10	4	24	18	6
Pieza *g*	9	7	7	25	21	4
Pieza *h*	8	8	4	24	16	8
Pieza *i*	6	1	3	13	7	6
Pieza *j*	3	1	1	7	3	4
Orden Palmer	*a-b-c-d-e-g-j-f-i-h*			F_{max} = 70		
Orden Trapecios	*a-b-c-d-e-g-f-h-i-j*			F_{max} = 67		

Fig. 6.1.5.29 Aplicación de Palmer y trapecios al problema Ignall-Schrage

	Pieza									
	a	*b*	*c*	*d*	*e*	*g*	*j*	*f*	*i*	*h*
Máquina 1	1	6	13	21	24	33	36	43	49	57
Máquina 2	3	15	21	30	32	40	41	53	54	66
Máquina 3	12	22	30	39	42	49	50	57	60	70

Fig. 6.1.5.30 Valores $f_j(k)$ para la permutación Palmer en el problema Ignall-Schrage ($F_{max} = 70$)

	Pieza									
	a	*b*	*c*	*d*	*e*	*g*	*f*	*h*	*i*	*j*
Máquina 1	1	6	13	21	24	33	40	48	54	57
Máquina 2	3	15	21	30	32	40	50	59	60	61
Máquina 3	12	22	30	39	42	49	54	63	66	67

Fig. 6.1.5.31 Valores $f_j(k)$ para la permutación trapecios en el problema Ignall-Schrage ($F_{max} = 67$)

Hemos aplicado las heurísticas Palmer y trapecios a colecciones de 1000 problemas cuyas duraciones se generaban aleatoriamente a partir de una distribución uniforme entre 1 y 20. Algunos resultados se recogen en la tabla 6.1.5.32.

	8 piezas	9 piezas	10 piezas
Valor medio de las cotas	105,674	116,014	127,165
Valor medio sol. Palmer	112,435	122,999	134,140
media Palmer/media cota	1,06398	1,06021	1,05485
Valor medio sol. trapecios	110,378	120,992	131,922
media trapecios/media cota	1,04451	1,04291	1,03741
Valor medio mejor solución	109,491	120,054	131,061
media mej. sol/media cota	1,03612	1,03482	1,03064
Nº veces idéntica solución	251	256	229
Nº veces mejor Palmer	220	209	217
Nº veces mejor trapecios	529	535	554

Fig. 6.1.5.32 Comparación resultados de las heurísticas Palmer y trapecios

Otras heurísticas

Asignar la paternidad, tanto en las heurísticas que siguen como en las anteriores, es difícil por cuanto han sido descubiertas y redescubiertas innumerables veces y aparecen en publicaciones con modificaciones no esenciales bajo diferentes firmas. Por otra parte, la casi totalidad de las referencias bibliográficas incluidas en los artículos y libros se centra en textos publicados en inglés omitiendo el resto. Por tanto, salvo las excepciones en las que podemos apoyarnos en argumentos sólidos, nos adaptaremos a las denominaciones habituales en los textos consultados, aun a sabiendas de que podemos estar perpetuando una injusticia. Para evitar incomprensiones hemos tenido interés en citar referencias con la fecha correspondiente. En los casos en que las heurísticas sean generalizables para $m>3$ escribiremos las expresiones en formato literal genérico.

Dudeck & Teuton (*JORSA*, vol. 12, nº 3, May 1964) han propuesto un algoritmo generalizable para $m>3$ si nos limitamos a las permutaciones ($n/m/P/F_{max}$). El algoritmo selecciona la primera pieza, luego entre las $n-1$ restantes la segunda, etc.

Sea $X_{j,i}$ el tiempo muerto en la máquina j que precede directamente a la operación que ocupa el lugar i, y $TM_{j,i}$ a la suma de todos los tiempos muertos en la máquina j hasta la posición i de la secuencia:

$$TM_{j,i} = \sum_{k=1}^{i} X_{j,k}$$

y $TM_{j,i}([1],[2],\ldots,[i])$ la misma suma con explícita identificación de las piezas particulares hasta la posición i de la secuencia.

Seleccionaremos para la primera posición una pieza h tal que:

$$TM_{j,2}(h,i) \leq TM_{j,2}(i,h) \text{ para toda } i \neq h \text{ y } j = 2,3,\ldots,m$$

(si no existe dicha pieza, los autores indicaron un procedimiento para seleccionar un subconjunto de piezas y se procede con cada pieza de este subconjunto como si fuese única).

Habiendo elegido *h*, se coloca en segunda posición una pieza h' tal que:

$$TM_{j,3}(h,h',i) \leq TM_{j,3}(h,i,h') \text{ para toda } i \neq h,h' \text{ y } j = 2,3,\ldots,m$$

y se continúa en dicha forma hasta completar la secuencia. Hay mucho trabajo, pero menos que en una enumeración completa. El algoritmo es muy eficaz, Dudeck & Teuton resolvieron 168 problemas con $m=$ 3 o 5 y $n=$ 3, 4, 5, 6 o 7 y obtuvieron siempre el óptimo. Concluyeron que esto ocurría siempre pero desgraciadamente Karush encontró un contraejemplo:

	Duración		
Pieza	Máq. 1	Máq. 2	Máq. 3
a	3	22	2
b	22	20	20
c	20	14	18

$$TM_{2,2}(a,b) = 3 < TM_{2,2}(b,a) = 22$$
$$TM_{3,2}(a,b) = 43 < TM_{3,2}(b,a) = 44$$
$$TM_{2,2}(a,c) = 3 < TM_{2,2}(c,a) = 20$$
$$TM_{3,2}(a,c) = 37 < TM_{3,2}(c,a) = 38$$

y sin embargo:

$$F_{max}(a,b,c) = 83 \; ; \; F_{max}(a,c,b) = 85 \; ; \; F_{max}(b,c,a) = 82$$
$$F_{max}(b,a,c) = 96 \; ; \; F_{max}(c,a,b) = 96 \; ; \; F_{max}(c,b,a) = 86$$

Teixidó (comunicación oral, noviembre 1967) propone, en la versión más simplificada, analizar las parejas de piezas (h,i) como si fuesen únicas en el taller y determinar su ocupación en las diferentes máquinas. Sea $f_j(h,i)$ el instante en que queda libre la máquina *j* cuando en el taller (con todas las máquinas libres en el instante 0) se ejecutan las operaciones sobre las piezas *h* e *i* en el orden *h-i*. Si las piezas se ejecutan en orden contrario el instante de liberación de la máquina será $f_j(i,h)$.

Consideraremos que *h* domina a *i* si:

$$f_m(h,i) < f_m(i,h)$$

o bien, aunque es menos contundente, si:

$$f_k(h,i) = f_k(i,h) \text{ para } j+1 \leq k \leq m$$

y

$$f_j(h,i) < f_j(i,h)$$

Teniendo en cuenta que siempre $f_1(h,i) = f_1(i,h)$, si a todo j:

$$f_j(h,i) = f_j(i,h)$$

entonces la dominancia entre h e i es recíproca, con lo que se permite así definir una relación de dominancia sobre todas las parejas de piezas. Considerando dicha relación como una relación de orden (aunque la transitividad no está garantizada) podemos obtener una secuencia de las piezas de acuerdo con dicha relación (construyendo el grafo de las dominancias, dicho orden está determinado por un camino hamiltoniano del mismo)

	a	*b*	*c*	*d*	*e*
a	-	(	1	1	1
b	1	-	1	1	
c			-	1	
d				-	
e		1	1	1	-
Σ	1	1	3	4	1

Fig. 6.1.5.33 Matriz de dominancias en el problema 1987

En la matriz de la figura 6.1.5.33 hemos indicado las dominancias entre piezas en el problema 1987 (un 1 significa que la pieza de la fila domina a la de la columna, un cero o blanco lo contrario). Si hallamos la suma de las columnas, en el caso en que la dominancia fuese transitiva hallaríamos todos los valores desde 0 a *n-1* (aunque puede darse dicha propiedad sin que la dominancia fuese transitiva) en cuyo caso sería fácil hallar el camino hamiltoniano ordenando las piezas en orden creciente de dicha suma.

En nuestro caso no es así: hay un triple empate a 1, que señala un circuito entre las piezas *b - a - e* (en este orden), por tanto la regla nos conduce a tres posibles soluciones:

$$b - a - e - c - d \quad F_{max} = 41$$
$$a - e - b - c - d \quad F_{max} = 40$$
$$e - b - a - c - d \quad F_{max} = 42$$

En las aplicaciones prácticas utilizamos la matriz de dominancias (escribiendo 0,5 en ambas casillas cuando la dominancia es recíproca), ordenando en orden creciente de la suma de las filas y deshaciendo los empates mediante trapecios (o en su defecto Palmer). En el caso presente la solución de este procedimiento es la segunda secuencia indicada de valor óptimo, 40.

La aplicación al problema de Ignall-Schrage puede verse en la matriz de la figura 6.1.5.34. En este caso existe un camino hamiltoniano único (la dominancia es transitiva) y el valor obtenido es semejante al de trapecios.

Una variante (Teixidó_2) consiste en comparar para todos los valores de *j* las ocupaciones $f_j(h,i)$ y $f_j(i,h)$, si

$f_j(h,i) < f_j(i,h)$ se añade a la casilla *(h,i)* de la tabla de dominancias generalizadas *(j-1)*
$f_j(h,i) > f_j(i,h)$ se añade a la casilla *(i,h)* de la tabla de dominancias generalizadas *(j-1)*
$f_j(h,i) = f_j(i,h)$ se añade a las dos casillas *(h,i)* y *(i,h)* de la tabla de dominancias generalizadas el mismo valor *(j-1)/2*

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
a	-	1	1	1	1	1	1	1	1	1
b		-	1	1	1	1	1	1	1	1
c			-	1	1	1	1	1	1	1
d				-	1	1	1	1	1	1
e					-				1	1
f					1	-		1	1	1
g					1	1	-	1	1	1
h					1			-	1	1
i									-	1
j										-
Σ	0	1	2	3	7	5	4	6	8	9
Orden Teixidó_1: *a-b-c-d-g-f-h-e-i-j* $F_{max} = 67$										

Fig. 6.1.5.34 Matriz de dominancias en el problema de Ignall-Schrage

se suman los valores de las columnas de la tabla y se ordenan las piezas por dichos valores en sentido creciente deshaciendo los empates mediante trapecios (y en su defecto mediante Palmer). Las matrices de las figuras 6.1.5.35 y 6.1.5.36 presentan las dominancias generalizadas para los problemas 1987 e Ignall-Schrage. Para este último hemos obtenido una solución óptima (véase la posición de *e*, *f* y *g*).

	a	*b*	*c*	*d*	*e*
a	-	1	3	3	3
b	2	-	2	2	1
c		1	-	3	1
d		1		-	0,5
e		2	2	2,5	-
Σ	2	5	7	10,5	5,5
Orden Teixidó_2: *a-b-e-c-d* F_{max} = 40					

Fig. 6.1.5.35 Matriz de dominancias generalizadas en el problema 1987

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
a	-	3	3	3	3	3	3	3	3	3
b		-	2	3	2	3	3	3	3	3
c		1	-	2	2	2	2	2	3	3
d			1	-	2	2	3	2,5	3	3
e		1	1	1	-	1	1	1	3	3
f			1	1	2	-	1	3	3	3
g			1		2	2	-	2	3	3
h			1	0,5	2		1	-	3	3
i									-	2,5
j									0,5	-
Σ	0	5	10	10,5	15	13	14	16,5	24,5	26,5
Orden Teixidó_2: *a-b-c-d-f-g-e-h-i-j* F_{max} = 66										

Fig. 6.1.5.36 Matriz de dominancias generalizadas en el problema Ignall-Schrage

J. N. D. Gupta (*Op. Res. Quart.*, vol. 22, nº 1, march 1971) propone calcular (utilizamos la nomenclatura $S_{4,i}$ dado que Gupta comparó su heurística con la de Palmer):

$$S_{4,i} = \frac{x(i)}{\underset{1 \le j \le m-1}{MIN} \{p_{j,i} + p_{j+1,i}\}}$$

donde $x(i) = 1$ si $p_{1,i} \geq p_{m,i}$

$x(i) = -1$ en los demás casos

y ordenar las piezas en orden creciente de $S_{4,i}$, resolviendo los empates a favor de la pieza cuya suma de tiempos en todas las máquinas sea menor (aunque este criterio de desempate no parece el mejor en la gran mayoría de los casos). Se puede observar que esta regla coincide con el algoritmo de Johnson para $m=2$ y para el caso especial $m=3$. En la tabla de la figura 6.1.5.37 hemos indicado la solución Gupta del problema 1987 y en la 6.1.5.38 la del problema de Ignall-Schrage.

	Máquinas			
	1	2	3	$S_{4,i}$
Pieza *a*	1	9	8	-1/10
Pieza *b*	7	2	9	-1/9
Pieza *c*	7	11	3	1/14
Pieza *d*	10	4	4	1/8
Pieza *e*	5	4	6	-1/9
Orden Gupta: *e - b - a - c - d* $F_{max}=42$				

Fig. 6.1.5.37 Aplicación de Gupta al problema 1987

	Máquinas			
	1	2	3	$S_{4,i}$
Pieza *a*	1	2	9	-1/3
Pieza *b*	5	9	7	-1/14
Pieza *c*	7	6	8	-1/13
Pieza *d*	8	9	9	-1/17
Pieza *e*	3	2	3	1/5
Pieza *f*	7	10	4	1/14
Pieza *g*	9	7	7	1/14
Pieza *h*	8	8	4	1/12
Pieza *i*	6	1	3	1/4
Pieza *j*	3	1	1	1/2
Orden Gupta: *a - c - b - d - f - g - h - e - i - j* $F_{max}=69$				

Fig. 6.1.5.38 Aplicación de Gupta al problema Ignall-Schrage

	Pieza									
	a	*c*	*b*	*d*	*f*	*g*	*h*	*e*	*i*	*j*
Máquina 1	1	8	13	21	28	37	45	48	54	57
Máquina 2	3	14	23	32	42	49	58	60	61	62
Máquina 3	12	22	30	41	46	56	62	65	68	69

Fig. 6.1.5.39 Valores $f_j(k)$ para la permutación Gupta en el problema Ignall-Schrage ($F_{max}=69$)

En las tablas de las figuras 6.1.5.40 y 6.1.5.41 hemos realizado una comparación de los cinco métodos (Palmer, trapecios, Teixidó_1, Teixidó_2 y Gupta) contrastando los resultados obtenidos con 10000 problemas (con $m=3$ y $n=8,9$ o 10) con duraciones generadas aleatoriamente mediante una ley uniforme entre 1 y 25. Aunque las conclusiones son difíciles de establecer, aparece la superioridad de trapecios en cuanto a valor de la solución y de Teixidó_1 en cuanto al número de veces que obtiene la mejor solución (superior al 50 % si se incluyen los empates). Curiosamente, cuando un único método proporciona la mejor solución, éste suele ser trapecios o Teixidó_2.

Dannenbring (1977) ha propuesto una variante de la heurística CDS (Campbell, Dudeck y Smith, apartado 6.1.5.4.2) que llama heurística RA (*rapid access procedure*) seguida de unos procedimientos de mejora RACS (*rapid access with close order search*) o RAES (*rapid access with extensive search*). Para RA calcula para cada pieza:

$$TD_{1,i} = \sum_{j=1}^{m} (m-j+1)\cdot p_{j,i} \; ; \quad TD_{2,i} = \sum_{j=1}^{m} j \cdot p_{j,i}$$

y aplica el algoritmo de Johnson a $TD_{1,i}$, $TD_{2,i}$. Habiendo observado que en muchos casos el óptimo se obtiene mediante permutación de dos piezas en la secuencia, propone estudiar las secuencias obtenidas a partir de la anterior permutando dos piezas contiguas. En RACS genera las *n-1* secuencias y adopta la mejor; en RAES si la mejor no es la original genera los *n-1* vecinos de la nueva secuencia mejor, y prosigue en esta forma mientras obtiene disminución de F_{max}. Claramente, RAES obtiene mejores soluciones que RACS, pero es más costoso en tiempo.

Número de piezas	8	9	10
Valor medio de la cota	130,751	143,994	157,510
Val. medio Palmer	139,133	152,694	166,481
Val. Palmer/Val. cota	1,06411	1,06042	1,05695
Val. medio trapecios	136,657	150,011	163,728
Val. trapecios/val. cota	1,04517	1,04179	1,03948
Val. medio Teixidó_1	136,475	150,065	163,859
Val. Teix_1/Val. cota	1,04378	1,04216	1,04031
Val. medio Teixidó_2	137,222	150,786	164,698
Val. Teix_2/Val. cota	1,04949	1,04717	1,04563
Val. medio Gupta	136,747	150,381	164,168
Val. Gupta/Val. cota	1,04586	1,04436	1,04227
Val. medio mejor solución	134,291	147,536	161,139
Val. mejor/Val. cota	1,02707	1,02460	1,02304

Fig. 6.1.5.39 Comparación de soluciones obtenidas mediante los métodos de Palmer, trapecios, Teixidó_1, Teixidó_2 y Gupta en 10.000 problemas con duraciones uniformemente distribuidas entre 1 y 25

	% de obtención de la mejor solución		
Número de piezas	8	9	10
Todos los métodos	10,96	9,10	7,99
Palmer	18,97 (6,31)	19,54 (7,21)	19,95 (7,64)
Trapecios	44,32 (7,17)	44,74 (9,35)	45,08 (10,23)
Teixidó_1	50,86 (3,81)	49,76 (4,16)	48,86 (4,54)
Teixidó_2	39,73 (7,99)	38,66 (8,61)	37,52 (8,59)
Gupta	48,37 (5,27)	46,50 (5,32)	46,57 (6,40)

Fig. 6.1.5.40 Comparación de soluciones obtenidas en 10.000 problemas con duraciones uniformemente distribuidas entre 1 y 25. En los porcentajes por método se han descontado los casos en que todos ellos dan la misma solución; entre paréntesis se indica la proporción cuando el método es el único en dar la mejor solución.

Puesto que RA es idéntico a la heurística trapecios (aunque diez años más joven), centraremos el interés en las heurísticas de mejora por intercambio, bastante tradicionales también como ya se ha indicado en 6.1.3. Existen algunos aspectos a tener en cuenta para diseñar alguna de estas heurísticas:

A1) Secuencia inicial de partida,

A2) Definición del entorno (vecinos) de una secuencia (permutación de dos piezas

contiguas o de dos cualesquiera),

A3) Estrategia de cambio de solución (generar todos los vecinos y evaluarlos eligiendo la mejor o pasar a una solución cuando es mejor que la en curso),

A4) Tratamiento de los empates.

Tomando las mismas colecciones de 1.000 problemas que hemos tratado con las heurísticas de Palmer y trapecios, y aplicando un procedimiento de mejora, a partir de las secuencias obtenidas, en el que se consideran como vecinas las secuencias obtenidas de una dada mediante la permutación de dos piezas cualesquiera, cambiando de solución incumbente o retenida cada vez que se produce una mejora y sin tratamiento de empates, los resultados son los de la tabla de la figura 6.1.5.42 (los tiempos de cálculo son de 5 a 8 veces superiores a los que nos han permitido construir la 6.1.5.32).

Número de piezas	8	9	10
Valor medio de la cota	105,674	116,014	127,165
Val. med. a partir de Palmer	107,958	118,522	129,535
Val. Palmer/Val. cota	1,02161	1,02162	1,01864
Val. med. a partir trapecios	107,687	118,018	128,954
Val. trapecios/Val. cota	1,01905	1,01727	1,01407
Val. medio mejor solución	107,452	117,793	128,736
Val. mejor/Val. cota	1,01682	1,01533	1,01235
Nº veces igual solución	723	675	653
Nº veces mejor Palmer	91	84	87
Nº veces mejor trapecios	186	241	260

Fig. 6.1.5.42 Comparación de soluciones obtenidas mediante mejora por intercambio a partir de la secuencia Palmer y de la secuencia trapecios

6.1.5.4.3 Procedimientos basados en la PLM

Wagner, Story & Giglio han propuesto un procedimiento basado en la programación lineal mixta (PLM). La notación es la siguiente:

z(k,i) = 1 si la pieza *i* está en la posición *j*
= 0 en los demás casos
Z(k) = [z(k,1) z(k,2) .... z(k,n)]'
x(j,k) = tiempo muerto en la máquina *j* antes de la pieza que ocupa el lugar *k*
y(j,k) = tiempo muerto de la pieza que ocupa la posición *k* entre las operaciones en las máquinas *j* y *j+1*

P1, P2, P3 columnas de los tiempos de proceso en las tres máquinas

Podemos plantear:

$$\sum_i z(k,i) = 1 \qquad \sum_k z(k,i) = 1$$

$$x(2,k+1) + P2' \cdot Z(k+1) + y(2,k+1) - y(2,k) - P3' \cdot Z(k) - x(3,k+1) = 0$$

$$P1' \cdot Z(k+1) + y(1,k+1) - y(1,k) - P1' \cdot Z(k) - x(2,k+1) = 0$$

$$k = 1,2,.....,n-1$$

$$[MIN] \sum_{j=1}^{n} x(3,j)$$

Existen $n^2 + 4 \cdot n$ incógnitas (de ellas n^2 de valor 0 o 1) y $4 \cdot n-2$ ecuaciones (para $n=10$, 140 incógnitas y 38 ecuaciones). Este procedimiento no proporciona resultados eficientes debido a las dificultades de resolución de los PLM.

6.1.5.4.4 Método de Lomnicki

Lomnicki (en *Operational Research Quarterly*, vol. 16, nº 1, marzo 1965) presentó un método de exploración dirigida (*branch and bound*) muy parecido a uno propesto por Ignall & Schrage, publicado casi simultáneamente (en *Operations Research*, vol. 13, nº 3, mayo 1965), aunque Lomnicki asegura que el suyo es más eficiente. Resuelve exactamente el problema propuesto y es aplicable a $m>3$ si limitamos las soluciones a las permutaciones (problemas $n/m/P/F_{max}$). El procedimiento consiste en una exploración inteligente del conjunto de permutaciones de las n piezas; dichas permutaciones se consideran agrupadas en paquetes, en cada uno de los cuales se han concretado las piezas que ocupan determinadas posiciones (las primeras). Para cada paquete puede determinarse una cota inferior del valor de F_{max} de las permutaciones que lo componen, y dado que el número de permutaciones del paquete es inferior al número global de las mismas, es verosímil que la cota será más ajustada que la obtenida en 6.1.5.4.1. En principio la cota será tanto más ajustada cuanto menos permutaciones contenga el paquete. Las cotas de diferentes paquetes serán en general diferentes, lo que hará la exploración de algunos paquetes más interesante que la de otros. Un paquete se explora dividiéndolo en subpaquetes. En el fondo lo más útil será que algunos paquetes tengan una cota tan mala que podamos rechazarlos sin explorar.

Vamos a describirlo sobre un ejemplo (problema 1972) cuyos datos están en la tabla de la figura 6.1.5.43.

Pieza/Máquina	1	2	3
1	1	3	2
2	2	5	3
3	3	1	1
4	2	7	6
5	3	4	1

Fig. 6.1.5.43 Datos del problema 1972

Las cotas longitudinales son:

$k_1 = 11 + (1 + 1) = 13$
$k_2 = 20 + (1 + 1) = 22$
$k_3 = 13 + (1 + 3) = 17$

y, dado que las cotas transversales no superan 17, podemos concluir:

$$F_{max} \geq 22$$

Aplicando las heurísticas tendremos:

Palmer 4 - 1 - 2 - 5 - 3 $F_{max} = 23$
Trapecios 1 - 2 - 4 - 5 - 3 $F_{max} = 24$

No tenemos garantía de haber hallado el valor óptimo, dado que la mejor solución disponible es superior a la cota. Vamos a intentar hallar una solución mejor, para lo que deberíamos explorar el conjunto de las 5! = 120 soluciones posibles. Inicialmente formaremos 5 paquetes o subconjuntos de soluciones, cada uno conteniendo las 24 soluciones que comienzan por una pieza determinada. Para cada paquete será posible obtener una cota asociada al mismo, que en algunos casos podrá ser superior a 22. Sea por ejemplo el subconjunto de las permutaciones que comienzan por la pieza 1; dicha pieza ocupará las máquinas hasta los instantes:

$$f_1(1) = 1 \; ; \; f_2(1) = 4 \; ; \; f_3(1) = 6$$

y calculemos para dicho subconjunto una cota de F_{max} siguiendo las mismas ideas utilizadas anteriormente para las cotas longitudinales con la salvedad de que la parte inicial está ya definida:

$$k_j(1) = f_j(1) + \sum_{i \neq 1} p_{j,i} + \min_{i \neq 1} \{p_{j+1,i} + \ldots + p_{m,i}\}$$

$$j = 1, 2, 3$$

cuya interpretación es sencilla: instante en que queda libre la máquina más resto de los tiempos de proceso en dicha máquina más el tiempo de la pieza más favorable en el resto de máquinas. Al no haber considerado tiempos muertos ni retrasos, la suma constituye una cota. En nuestro caso:

$$k_1(1) = 1 + 2 + 3 + 2 + 3 + (1 + 1) = 13$$
$$k_2(1) = 4 + 5 + 1 + 7 + 4 + (1) = 22$$
$$k_3(1) = 6 + 3 + 1 + 6 + 1 = 17$$

Por tanto la cota de este subconjunto sigue siendo 22. Veamos ahora el subconjunto de las permutaciones que empiezan por la pieza 2:

$$f_1(2) = 2\ ;\ f_2(2) = 7\ ;\ f_3(2) = 10$$
$$k_1(2) = 2 + 1 + 3 + 2 + 3 + (1 + 1) = 13$$
$$k_2(2) = 7 + 3 + 1 + 7 + 4 + (1) = 23$$
$$k_3(2) = 10 + 2 + 1 + 6 + 1 = 20$$

En este caso la cota es mayor, 23, y este subconjunto podría dejar de ser considerado en lo que sigue dado que ya disponemos de una solución de valor 23. Si establecemos una lista de subconjuntos y cotas, ordenados por orden creciente de las cotas, tendremos:

Subconjunto	Cota
1	22
2	23
4	23
3	24
5	24

De la lista deducimos que es más prometedor el subconjunto de las soluciones que comienzan por la pieza 1. Los demás subconjuntos no nos ofrecen soluciones mejores que las ya disponibles, y podríamos suprimirlos; en particular vemos que la solución Palmer tenía el mejor valor posible para las permutaciones que comienzan por la pieza 4. La posible solución mejor que las disponibles es una de las 24 permutaciones que empiezan por la pieza 1; las 96 permutaciones restantes quedan eliminadas de la exploración.

Exploremos el primer subconjunto, descomponiéndolo en 4, colocando a continuación de la pieza 1 cada una de las otras 4. Por ejemplo, en el subconjunto de las soluciones que empiezan por 1-2 tendremos:

$$f_1(1\text{-}2) = 3\ ;\ f_2(1\text{-}2) = 9\ ;\ f_3(1\text{-}2) = 12$$
$$k_1(1\text{-}2) = 3 + (3 + 2 + 3) + (1 + 1) = 13$$
$$k_2(1\text{-}2) = 9 + 1 + 7 + 4 + (1) = 22$$
$$k_3(1\text{-}2) = 12 + 1 + 6 + 1 = 20$$

por lo que la cota sigue siendo 22. En la determinación de k hemos utilizado el siguiente procedimiento. Sea I el conjunto de los índices de las piezas ya consideradas y $P(I)$ la permutación de las mismas, permutación inicial de todas las soluciones del subconjunto cuya cota queremos hallar. Sea N el conjunto de los índices no considerados. Las soluciones del subconjunto se obtienen colocando detrás de $P(I)$ todas las permutaciones de los elementos de N. Sea $f_j(P(I))$ el instante en que la máquina j queda libre cuando se elaboran en el taller las piezas de I en el orden $P(I)$. Entonces:

$$k_j(P(I)) = f_j(P(I)) + \sum_{i \in N} p_{j,i} + \min_{i \in N} \{p_{j+1,i} + \ldots + p_{m,i}\}$$

Determinando las cotas para los 4 nuevos subconjuntos obtenemos la lista

Subconjunto	Cota
1 - 2	22
1 - 3	22
1 - 4	22
1 - 5	22
2	23
4	23
3	24
5	24

Reiterando el procedimiento y descomponiendo el subconjunto 12, obtenemos más cotas de valor 22 todavía (en caso de empate de las cotas damos preferencia de cara a la exploración o descomposición al subconjunto con menos permutaciones, es decir, con más elementos en I):

1 2 3 4 5

1 - 3 2 4 5

Subconjunto	Cota
1 - 2 - 3	22
1 - 2 - 5	22
1 - 3	22
1 - 4	22
1 - 5	22
2	23
4	23
1 - 2 - 4	24
3	24
5	24

La descomposición de 1-2-3 y 1-2-5, sin embargo, conduce a valores superiores:

Subconjunto	Cota
1 - 3	22
1 - 4	22
1 - 5	22
2	23
4	23
1 - 2 - 3 - 4	24*
1 - 2 - 4	24
3	24
5	24
1 - 2 - 3 - 5	27*
1 - 2 - 5 - 3	27*
1 - 2 - 5 - 4	27*

Los subconjuntos señalados con un asterisco contienen una sola solución o permutación, exactamente las 1-2-3-4-5, 1-2-3-5-4, 1-2-5-3-4, 1-2-5-4-3. Las cotas indicadas coinciden con los valores de F_{max}. Por tanto, la mejor solución hallada por este procedimiento hasta el momento tienen $F_{max} = 24$, y si buscásemos una sola solución óptima a partir de este momento podríamos prescindir de los subconjuntos de la lista posteriores a 1-2-3-4 (de hecho, dado que disponemos de soluciones dadas por las heurísticas bastaría mantener los tres primeros, pero vamos a actuar como si aplicaramos Lomnicki autónomamente). Pasamos a descomponer 1-3 que está en cabeza de lista:

Subconjunto	Cota
1 - 4	22
1 - 5	22
1 - 3 - 2	23
1 - 3 - 4	23
2	23
4	23
1 - 2 - 3 -4	24*
1 - 3 - 5	26

Prescindimos de 1-3-5 y pasamos a explorar 1-4

Subconjunto	Cota
1 - 4 - 2	22
1 - 4 - 3	22
1 - 4 - 5	22
1 - 5	22
1 - 3 - 2	23
1 - 3 - 4	23
2	23
4	23
1 - 2 - 3 - 4	24*

Exploremos ahora 1-4-2

Subconjunto	Cota
1 - 4 - 2 - 3	22*
1 - 4 - 2 - 5	22*
1 - 4 - 3	22
1 - 4 - 5	22
1 - 5	22
1 - 3 - 2	23
1 - 3 - 4	23
2	23
4	23
1 - 2 - 3 - 4	24*

Los subconjuntos 1-4-2-3 y 1-4-2-5 constan de un sólo elemento, 1-4-2-3-5 y 1-4-2-5-3 respectivamente cuya F_{max} es de valor 22, coincidente con la cota. Siendo dichas cotas las menores de la lista (los subconjuntos están en cabeza) dichas soluciones son óptimas.

Si deseamos comprobar si existen más soluciones óptimas prescindiremos de los subconjuntos de cota superior a 22, y continuaremos la exploración de 1-4-3, 1-4-5 y 1-5.

Obtendremos:

Subconjunto	Cota
1 - 4 - 2 - 3	22*
1 - 4 - 2 - 5	22*
1 - 4 - 3 - 2	22*
1 - 5 - 2	23
1 - 4 - 3 - 5	24*
1 - 4 - 5 - 2	24*
1 - 4 - 5 - 3	24*
1 - 5 - 3	24
1 - 5 - 4	25

Luego hay tres soluciones óptimas con F_{max} = 22; 1-4-2-3-5, 1-4-2-5-3 y 1-4-3-2-5.

Normalmente los subconjuntos que van apareciendo en el transcurso de la aplicación del algoritmo se representan en forma de árbol (de aquí la expresión inglesa *branch*, ramificar) tal como se indica en la figura 6.1.5.44. Los arcos del árbol indican la relación de inclusión entre los subconjuntos; los vértices pendientes, los que no están ramificados, corresponden a los subconjuntos todavía no explorados, o bien a subconjuntos que contienen una sola solución (y que por tanto no pueden subdividirse).

Pasemos a la descripción formal. En un momento dado de la aplicación del algoritmo disponemos de una partición del conjunto de permutaciones o soluciones en paquetes o subconjuntos (correspondientes a los vértices pendientes del árbol). Cada paquete se distingue por el conjunto de índices *I* de las piezas consideradas y por una permutación de estos índices *P(I)*. Pueden existir dos paquetes con el mismo *I* pero con distintas permutaciones *P(I)* y *P'(I)*. Cada uno de estos paquetes corresponde a las soluciones, permutaciones de las *n* piezas, que empiezan por *P(I)*. Por consiguiente, si $|I|$ es el número de elementos de *I*, el número de soluciones del subconjunto es $(n - |I|)! = |N|!$, donde *N* es el conjunto de índices de piezas que no están en *I*. Existe un procedimiento de partición o exploración que consiste en, una vez elegido un paquete que contenga varias soluciones, descomponerlo en varios paquetes de forma que cada una de las soluciones del paquete padre esté en uno y solo uno de los paquetes hijos. En nuestro caso hemos empleado el procedimiento de crear $|N|$ paquetes hijos concatenando cada una de las piezas de *N* detrás de *P(I)* en la permutación característica del mismo. Este procedimiento debe ser capaz de distinguir, según lo dicho, los paquetes que contienen una sola solución (vértices terminales del árbol).

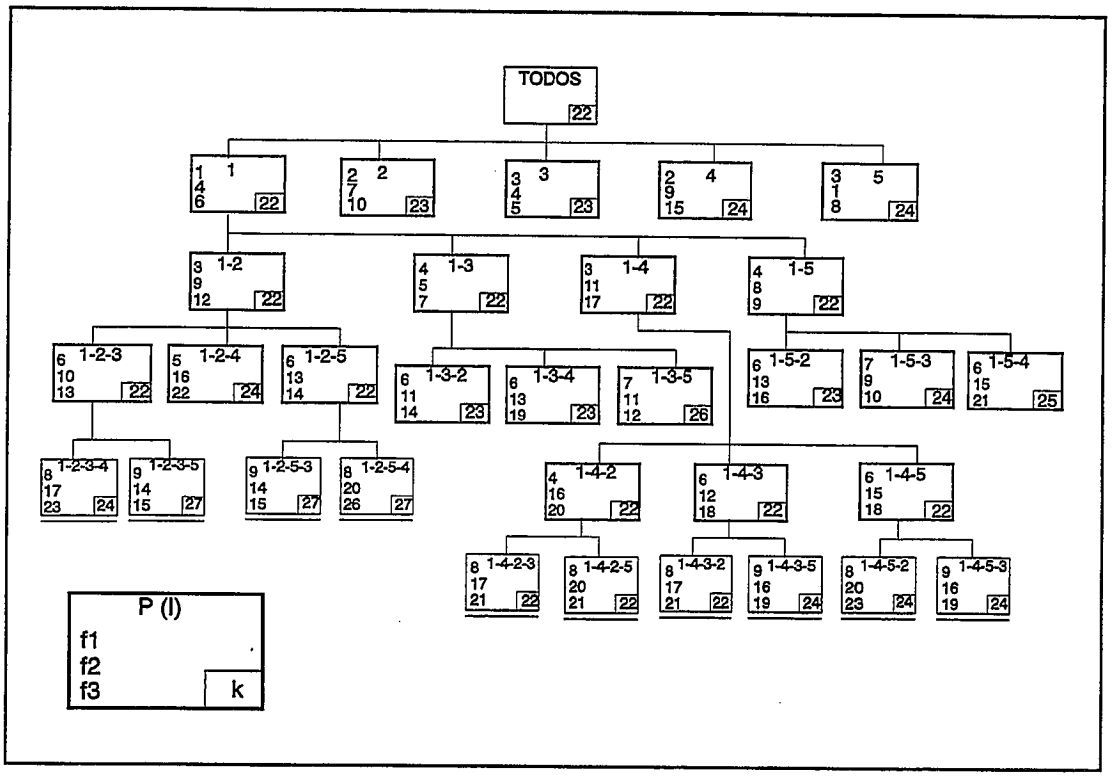


Fig. 6.1.5.44 Árbol correspondiente a la aplicación del método de Lomnicki al problema 1972

Cada paquete está asociado a un valor *k(P(I))*, cota inferior del valor F_{max} de las soluciones contenidas en el paquete. Esta cota puede obtenerse mediante un procedimiento de acotación similar al indicado anteriormente. Los condicionantes principales del procedimiento de acotación son: ningún hijo debe tener una cota inferior a la de su padre y, cuando el vértice es terminal, la cota debe coíncidir con el valor de la única solución que contiene el paquete. Naturalmente cuanto más ajustada sea la cota más eficiente será el algoritmo.

Inicialmente existe un único paquete (que suele etiquetarse "TODOS") cuya cota es la inicial del problema. Adicionalmente existe una estrategia de exploración que esencialmente sirve para elegir el paquete a explorar y subsidiariamente la parte del árbol que se conserva en memoria durante el desarrollo del algoritmo. Una estrategia básica muy utilizada consiste en elegir para su exploración el vértice pendiente con mejor cota y, en caso de empate, el más avanzado (es decir, con más piezas consideradas, con $|I|$ mayor). Una vez llegados a un vértice terminal y hallada una solución puede prescindirse de todos los vértices pendientes de cota superior al valor de F_{max} de la mejor solución hallada hasta el momento, e incluso de los de cota igual a dicho valor si nos contentamos

con una sola solución óptima en caso de que haya varias. Esta supresión de vértices limita considerablemente el volumen de información a conservar en el transcurso del algoritmo, por lo que habitualmente es útil determinar mediante un procedimiento heurístico rápido una primera solución satisfactoria antes de iniciar el algoritmo de exploración, o bien emplear inicialmente una estrategia de exploración que conduzca rápidamente a una primera solución (elegir el vértice pendiente más avanzado, mayor $|I|$, y en caso de empate el de mejor cota).

Diversos trabajos realizados en el Laboratori d'Organització de la Producció han permitido mejorar la eficiencia del algoritmo de Lomnicki, por un lado utilizando procedimientos de acotación más eficientes (algunos de ellos basados en las cotas transversales) y por otro diseñando un algoritmo denominado Lomnicki-pendular consistente en aplicar simultáneamente el algoritmo básico a dos problemas, el original y su inverso.

6.1.5.5 Problemas $n/m/P/F_{max}$, con $m > 3$

Los problemas $n/m/P/F_{max}$, con $m > 3$, son muy similares a los $n/3/F/F_{max}$. También aquí buscamos una de las $n!$ permutaciones de las n piezas. Las cotas de F_{max} constituyen una generalización de las ya vistas.

6.1.5.5.1 Cotas longitudinales (en los problemas $n/m/P/F_{max}$)

$$k_1 = \sum_{i=1}^{n} p_{1,i} + \min_i \{\sum_{j=2}^{m} p_{j,i}\}$$

$$k_j = \sum_{i=1}^{n} p_{j,i} + \min_{h,i} \{\sum_{k=1}^{j-1} p_{k,h} + \sum_{k=j+1}^{m} p_{k,i}\} \qquad (j = 2,3,\ldots,m-1 \;;\; h \neq i)$$

$$k_m = \sum_{i=1}^{n} p_{m,i} + \min_i \{\sum_{j=1}^{m-1} p_{j,i}\}$$

$$F_{max} \geq max \{ k_1, k_2, \ldots\ldots, k_m \}$$

6.1.5.5.2 Cotas transversales (en los problemas $n/m/P/F_{max}$)

$$h_i = \sum_{j=1}^{m} p_{j,i} + \sum_{s \neq i}^{n} \{p_{1,s}, p_{m,s}\}$$

$$F_{max} \geq \max \{ h_1, h_2, \ldots\ldots, h_n \}$$

$n = 5$ $m = 4$	Máquina				
	A	B	C	D	Σ
Pieza 1	9	14	15	1	39
Pieza 2	5	9	1	8	23
Pieza 3	11	5	7	8	31
Pieza 4	6	19	15	1	41
Pieza 5	16	3	4	5	28
Σ	47	50	42	23	
Cotas:	$k_A = 47 + 7 = 54$ $k_B = 50 + 5 + 9 = 64$ $k_C = 42 + 14 + 1 = 57$ $k_D = 23 + 15 = 38$		$h_1 = 39 + 18 = 57$ $h_2 = 23 + 26 = 49$ $h_3 = 31 + 12 = 43$ $h_4 = 41 + 19 = 60$ $h_5 = 28 + 15 = 43$		
$F_{max} \geq 64$					

Fig. 6.1.5.45 Datos y cotas del problema 9EF1

6.1.5.5.3 Métodos exactos para los problemas $n/m/P/F_{max}$ con $m>3$

Como en el caso $m=3$, sólo caben procedimientos basados en la programación lineal mixta o en la exploración implícita de todas las soluciones (métodos de *branch-and-bound*) como la extensión del de Lomnicki. Incluso con ordenador el consumo de tiempo es importante, lo que limita su utilización a valores reducidos de n y m.

6.1.5.5.4 Heurística de Campbell, Dudeck y Smith

La generalización a $m > 3$ de lo visto anteriormente es muy sencilla. Ahora deberemos calcular para $K = 1,2, \ldots, m-1$ los valores:

$$T_{1,i}(K) = \sum_{j=1}^{K} p_{j,i}$$

$$T_{2,i}(K) = \sum_{j=m-K+1}^{m} p_{j,i}$$

aplicar a cada una de las $m-1$ parejas de valores T el algoritmo de ordenación de Johnson, determinar F_{max} y guardar aquella permutación que haya proporcionado el mejor valor. En el problema 9EF1 las diferentes permutaciones obtenidas son poco consistentes entre sí

(en caso de ambigüedad hemos aplicado las reglas complementarias descritas anteriormente para Palmer/trapecios). La mejor solución se obtiene para $K=3$, con valor $F_{max}=83$:

2 - 4 - 1 - 3 - 5

	$K = 1$		$K = 2$		$K = 3$	
	$T_{1,i}$	$T_{2,i}$	$T_{1,i}$	$T_{2,1}$	$T_{1,i}$	$T_{2,1}$
Pieza 1	9	1	23	16	38	30
Pieza 2	5	8	14	9	15	18
Pieza 3	11	8	16	15	23	20
Pieza 4	6	1	25	16	40	35
Pieza 5	16	5	19	9	23	12
Orden $K = 1$: 2 - 3 - 5 - 4 - 1 $F_{max} = 88$						
Orden $K = 2$: 1 - 4 - 3 - 2 - 5 $F_{max} = 85$						
Orden $K = 3$: 2 - 4 - 1 - 3 - 5 $F_{max} = 83$						

Fig. 6.1.5.46 Aplicación de la heurística de Campbell, Durek y Smith al problema 9EF1

6.1.5.5.5 Heurísticas Palmer y trapecios

También aquí la generalización es muy simple:

$$S_{1,i} = (m-1).p_{1,i} + (m-2).p_{2,i} + \ldots\ldots + 1.p_{m-1,i}$$

$$S_{2,i} = p_{2,i} + 2.p_{3,i} + 3.p_{4,i} + \ldots\ldots + (m-1).p_{m,i}$$

$$S_{3,i} = S_{1,i} - S_{2,i}$$

La heurística Palmer ordena las piezas en orden creciente de $S_{3,i}$; la heurística trapecios aplica el algoritmo de ordenación de Johnson a $S_{1,i}$, $S_{2,i}$. En cada caso se utiliza la regla alternativa en caso de ambigüedad, y si ésta subsiste se da prioridad a la pieza con $p_{1,i}$ menor. En el problema 9EF1 la heurística Palmer nos proporciona la mejor solución hallada hasta el momento, con $F_{max} = 80$ (el valor óptimo es $F_{max} = 76$):

2 - 3 - 4 - 1 - 5

Las experiencias realizadas al respecto nos han mostrado que para valores reducidos de m ($m < 6$) la heurística trapecios suele dar, en promedio, mejores resultados que Palmer, pero al crecer m se invierte la relación.

	Máquina						
	A	B	C	D	$S_{1,i}$	$S_{2,i}$	$S_{i,3}$
Pieza 1	9	14	15	1	70	47	23
Pieza 2	5	9	1	8	34	35	-1
Pieza 3	11	5	7	8	50	43	7
Pieza 4	6	19	15	1	71	52	19
Pieza 5	16	3	4	5	58	26	32
Orden Palmer: 2 - 3 - 4 - 1 - 5 F_{max} = 80 Orden trapecios: 2 - 4 - 1 - 3 - 5 F_{max} = 83							

Fig. 6.1.5.47 Aplicación de las heurísticas Palmer y trapecios al problema 9EF1

	Piezas				
	2	3	4	1	5
Máquina A	5	16	22	31	47
Máquina B	14	21	41	55	58
Máquina C	15	28	56	71	75
Máquina D	23	36	57	72	80

Fig. 6.1.5.48 Valores $f_j(k)$ para la permutación Palmer en el problema 9EF1 ($F_{max}=80$)

6.1.5.6 Problemas $n/m/F/F_{max}$, con $m > 3$

Los problemas $n/m/F/F_{max}$, con $m > 3$, tienen características diferenciales respecto a los $n/3/F/F_{max}$, puesto que la solución óptima no tiene por qué conservar la secuencia en todas las máquinas. Solamente podemos garantizar el mismo orden en la primera y segunda máquinas por un lado y en la penúltima y última por otro. El número de conjuntos de secuencias a considerar tiene la cota superior:

$$(n!)^{m-2}$$

que puede tener un valor muy elevado. Las cotas de F_{max} constituyen, en la mayoría de los casos, una generalización de las ya vistas.

6.1.5.6.1 Cotas longitudinales (en los problemas $n/m/F/F_{max}$)

Son idénticas a las utilizadas para $n/m/P/F_{max}$

$$k_1 = \sum_{i=1}^{n} p_{1,i} + \min_i \{ \sum_{j=2}^{m} p_{j,i} \}$$

$$k_j = \sum_{i=1}^{n} p_{j,i} + \min_{h,i} \{ \sum_{k=1}^{j-1} p_{k,h} + \sum_{k=j+1}^{m} p_{k,i} \} \qquad (j = 2,3,\ldots,m-1 \; ; \; h \neq i)$$

$$k_m = \sum_{i=1}^{n} p_{m,i} + \min_i \{ \sum_{j=1}^{m-1} p_{j,i} \}$$

$$F_{max} \geq \max \{ k_1, k_2, \ldots\ldots, k_m \}$$

6.1.5.6.2 Cotas transversales (en los problemas $n/m/F/F_{max}$)

Aquí debemos tener en cuenta que una pieza puede "saltar" por encima de otra. En el caso de que no existan duraciones nulas ($p_{j,i} > 0$ a todo i y j), podemos calcular para cada pieza:

$$KTR_i = \min \{ p_{1,i} , p_{2,i}+p_{3,i} , p_{3,i}+p_{4,i} , \ldots , p_{m-2,i}+p_{m-1,i} , p_{m,i} \}$$

y establecer:

$$h_i = \sum_{j=1}^{m} p_{j,i} + \sum_{s \neq i} \{ KTR_s \}$$

$$F_{max} \geq \max \{ h_1 , h_2 , \ldots\ldots , h_n \}$$

si existe algún $p_{j,i}=0$, es decir, si alguna pieza "salta" una máquina, la determinación de la cota transversal es mucho más complicada. En primera aproximación puede tomarse:

$$KTR_i = \min_j \{ p_{j,i} \}$$

y aplicar la expresión anterior de h_i, pero los resultados no serán, en general, demasiado buenos, ya que el valor de *KTR* está muy subevaluado.

En el problema 4114, ya estudiado en una figura anterior, la cota obtenida es 12, que coincide con la solución óptima obtenida consistente en la secuencia 2-1 en las dos primeras máquinas y la 1-2 en la tercera y cuarta. Puede comprobarse que utilizando la misma secuencia en las cuatro máquinas, es decir, tratando el problema como uno $2/4/P/F_{max}$, la cota transversal sería 14, igual a los valores de F_{max} obtenido en cualquiera de las dos permutaciones posibles.

$n = 2$ $m = 4$	Máquina					
		A	B	C	D	Σ
	Pieza 1 Pieza 2	4 1	1 4	1 4	4 1	10 10
	Σ	5	5	5	5	
Cotas:	$k_A = 5+6 = 11$ $k_B = 5+1+5 = 11$ $k_C = 5+5+1 = 11$ $k_D = 5+6 = 11$ $F_{max} \geq 12$				$KTR_1 = 1+1 = 2$ $KTR_2 = 1$ $h_1 = 10+1 = 11$ $h_2 = 10+2 = 12$	

Fig. 6.1.5.49 Datos y cotas del problema 4114

6.1.5.6.3 Métodos exactos para los problemas $n/m/F/F_{max}$ con $m>3$

Como en el caso $m=3$, sólo caben procedimientos basados en la programación lineal mixta o en la exploración implícita de todas las soluciones (métodos de *branch-and-bound*). Incluso con ordenador el consumo de tiempo es importante, lo que limita su utilización a valores muy reducidos de n y m.

6.1.5.6.4 Métodos heurísticos para los problemas $n/m/F/F_{max}$ con $m>3$

Podemos utilizar dos vías:

- resolver el problema $n/m/P/F_{max}$ correspondiente, con lo que además de las deficiencias propias de las heurísticas añadimos la de restringir el conjunto de soluciones explorado, no teniendo en cuenta gran cantidad de las mismas entre las que puede hallarse la óptima. La utilización de un procedimiento exacto para los problemas $n/m/P/F_{max}$ constituirá aquí un método heurístico por dicha razón,

- utilizar alguno de los procedimientos "dinámicos" que describimos más adelante.

6.1.6 Problemas *flow-shop* semidinámicos y dinámicos

En esta sección vamos a liberarnos prácticamente de todas las restricciones que las hipótesis de Conway, Maxwell y Miller nos imponían, con lo que el acercamiento a la realidad industrial (por lo menos en el caso *flow-shop*) será mucho más estrecho.

6.1.6.1 Problemas semidinámicos: generalidades

Un problema semidinámico se caracteriza por el hecho de que no todas las máquinas están disponibles en el instante 0 y/o no todas las piezas están disponibles en el instante 0. Como el número de piezas es finito, y por tanto la ocupación del taller también podemos seguir utilizando los criterios tradicionales basados en c_{max}, F_{max} o F_{med}, exceptuando F_{max} en el caso de una sola màquina.

6.1.6.2 Problema $n/1/F/T_{max}$

Para el caso estático puro ya hemos indicado el teorema de Jackson que conduce a la secuencia EDD, conforme a la fecha contractual de vencimiento (6.1.4.2). Un algoritmo heurístico sencillo y razonablemente bueno para el caso dinámico, basado en ideas análogas y que denominaremos *algoritmo de Jackson*, es el siguiente (utilizamos la misma nomenclatura para duraciones y fechas comprometidas que en los casos 1 máquina ya vistos):

- sea $f=f_1$ el instante en que la máquina se halla disponible en cada uno de los pasos, en virtud de las operaciones ya programadas (inicialmente f será el instante de disponibilidad inicial de la máquina, normalmente, en tiempo relativo, el instante 0),

- en el transcurso del algoritmo las piezas (u operaciones, ya que solo existe una para cada pieza) se clasifican en las programadas P, y las elegibles o programables E. Inicialmente todas las piezas están en E y progresivamente las iremos pasando a P.

1) Determinar rp_i, instante en el que puede empezar la operación, para todas las piezas no programadas:

$$rp_i = \max \{ r_i , f \} \quad \text{para todo } i \in E,$$

2) Determinar fp, instante en que la máquina puede iniciar una nueva operación:

$$fp = \min \{ rp_i \} \quad \text{para todo } i \in E,$$

3) Determinar el conjunto de piezas que pueden dar un programa sin retrasos E' :

$$E' = \{ i \in E \text{ y } rp_i = fp \}$$

4) Si en E' existe una sola pieza se programa. Si existen varias se ordenan por d_i creciente y se programa la primera.

5) La pieza programada se pasa de E a P. Se actualiza el valor de f, si h es la pieza elegida:

$$f = rp_h + p_h$$

6) Si E está vacío hemos programado todas las piezas; en caso contrario volvemos al paso (1).

Una vez programadas todas las piezas podemos calcular c_i, L_i, T_i y T_{max}. Puede buscarse una mejora de la solución identificando la pieza que nos conduce al valor T_{max} e intentando adelantarla, si es posible, retrasando alguna de sus anteriores. En el ejemplo de las tablas adjuntas (figuras 6.1.6.1 y 6.1.6.2) no es posible mejorar la solución hallada directamente, pues toda permutación de la pieza 10 con las anteriores aumentaría T_{max}.

$f=0$											
Pieza	i	1	2	3	4	5	6	7	8	9	10
	r_i	12	7	9	18	28	0	37	25	40	32
	p_i	4	8	3	7	9	5	2	6	4	8
	d_i	23	32	17	25	41	12	43	39	50	45

Fig. 6.1.6.1 Datos del problema AX1SD

Secuencia Piezas		6	2	3	1	4	8	5	7	10	9
	c_i	5	15	18	22	22	35	44	46	54	58
	L_i	-7	-17	1	-1	4	-4	3	3	9	8
	T_i	0	0	1	0	4	0	3	3	9	
$T_{max} = 9$											

Fig. 6.1.6.2 Resultado de aplicación del algoritmo de Jackson al problema AX1SD

Un procedimiento sistemático para mejorar la solución encontrada mediante el algoritmo de Jackson lo detallamos en el apartado siguiente en el que presentamos el método de Carlier.

6.1.6.2.1 Algoritmo de Carlier

Vamos a dar otra forma al problema anterior. Consideraremos n piezas $I = \{1, 2, ..., n\}$, con instantes de disponibilidad r_i que deben sufrir una operación de duración p_i en una misma máquina, disponible desde el instante 0. Tras sufrir la operación en la máquina la pieza i debe permanecer inmovilizada durante un tiempo q_i ("duración de latencia").

Deseamos encontrar una programa (o una secuencia) que minimice:

$$\max \{ c_i + q_i \} = \max \{ t_i + p_i + q_i \}$$

siendo t_i el instante de comienzo de la operación de la pieza i en la máquina ($t_i \geq r_i$). Para simplificar hablaremos de duración refiriéndonos a $\max(c_i+q_i)$ y de duración óptima en el caso de la secuencia buscada, empleando los símbolos f y f^* para indicar dichos valores cuando sea oportuno. El ejercicio del apartado anterior se transforma fácilmente en el de la tabla de la figura 6.1.6.3, haciendo q_i igual a la diferencia entre d_i y un valor cualquiera, aquí 50 para evitar valores de q negativos.

$f=0$											
Pieza	i	1	2	3	4	5	6	7	8	9	10
	r_i	12	7	9	18	28	0	37	25	40	32
	p_i	4	8	3	7	9	5	2	6	4	8
	d_i	27	18	33	25	9	38	7	11	0	5

Fig. 6.1.6.3 Datos del problema AX1SD transformado

La única alteración al algoritmo de Jackson, para aplicarlo a este formato, consiste en:

4) Si en E' existe una sola pieza se programa. Si existen varias se ordenan por q_i decreciente y se programa la primera.

Secuencia piezas		6	2	3	1	4	8	5	7	10	9
	t_i	0	7	15	18	22	29	35	44	46	54
	c_i	5	15	18	22	29	35	44	46	54	58
	$c_i + q_i$	43	33	51	49	54	46	53	53	59	58
$max(c_i+q_i)=59$											

Fig. 6.1.6.4 Resultado de aplicación del algoritmo de Jackson al problema AX1SD transformado

Es posible representar mediante un grafo (tipo Roy) este problema; a una permutación o secuencia $S = \{ i_1, i_2, \ldots, i_n \}$ se puede asociar el grafo $G = (X, U)$ donde el conjunto de vértices X tiene $n+2$ elementos, los que coresponden a las n piezas más uno inicial, vértice 0, y uno final $n+1$. El conjunto de arcos puede partirse en tres U_1, U_2 y U_3, donde:

$$U_1 = \{ (0,i) \mid i \in I \} \text{ con valor asociado } r_i$$
$$U_2 = \{ (i_p, i_{p+1}) \mid 1 \leq p \leq n-1 \} \text{ con valor } p_{i_p}$$
$$U_3 = \{ (i, n+1) \mid i \in I \} \text{ con valor } p_i + q_i$$

El camino crítico de este grafo nos da el valor $\max(c_i+q_i)$. A cada secuencia corresponde

un conjunto de arcos U_2, solución posible de la ligadura disyuntiva existente entre los vértices I.

Sea I_1 un subconjunto cualquiera de I, la siguiente evaluación:

$$K(I_1) = \min_{i \in I_1} \{r_i\} + \sum_{i \in I_1} p_i + \min_{i \in I_1} \{q_i\}$$

es una cota de la duración óptima (ya que la duración es algún camino de alguna secuencia posible). Normalmente podría interesarnos:

$$V = \max \{ K(I_1) \} \text{ para todos los subconjuntos } I_1 \text{ de } I$$

que de hecho es la duración óptima del caso preemptivo (con interrupción de operaciones), fácil de calcular como veremos más adelante.

$f = 0$								
Pieza	i	1	2	3	4	5	6	7
	r_i	10	13	11	20	30	0	30
	p_i	5	6	7	4	3	6	2
	q_i	7	26	24	21	8	17	0

Fig. 6.1.6.5 Datos del problema 7X1SD

En el problema 7X1SD (figura 6.1.6.5) la secuencia de Jackson es la 6 - 1 - 2 - 3 - 4 - 5 - 7, representada en el grafo de la figura 6.1.6.6.

En lo que sigue supondremos que las piezas se han numerado según el orden de Jackson. Podemos definir, de acuerdo con Carlier:

pieza pivote inicial I, la de mayor número que pertenece a un camino crítico del grafo asociado,

pieza pivote terminal h, la de menor número tal que $(0, h, \ldots, I, n+1)$ es un camino crítico,

pieza crítica c, la de mayor número del intervalo $[h,I]$ tal que $q_c < q_i$; si no hay ninguna en dicho intervalo por convención haremos $c = h-1$ (aunque de hecho la solución hallada es óptima, y por tanto no procede hacer nada más),

conjunto crítico J, conjunto de las piezas del intervalo $[c+1, I]$

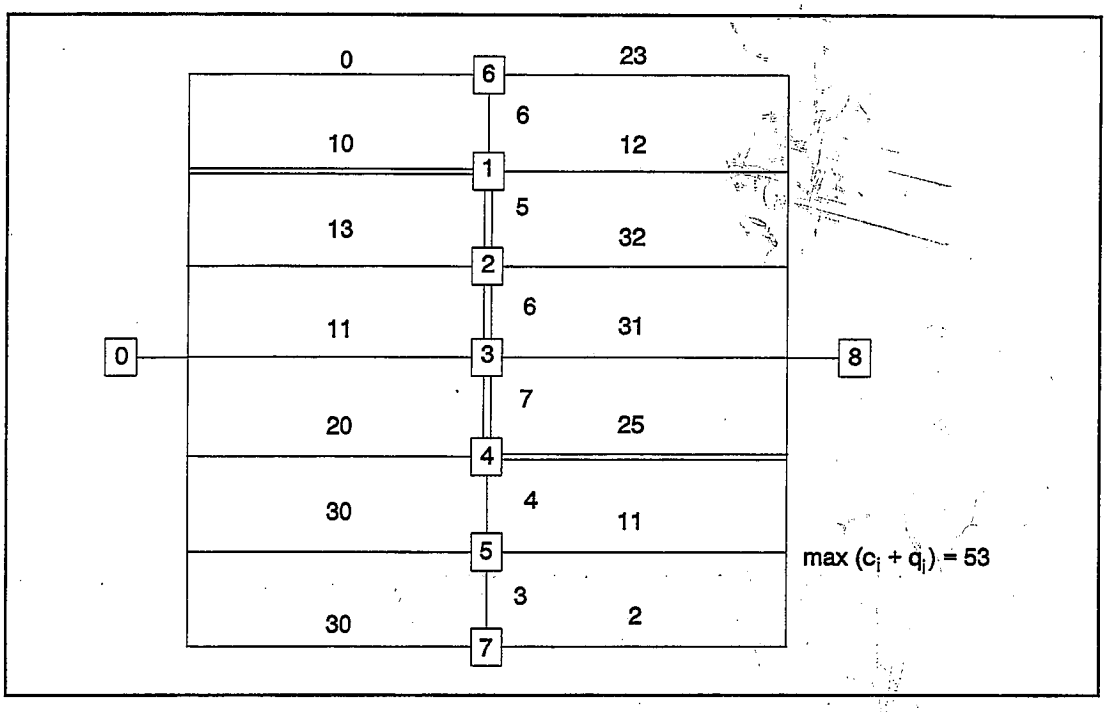


Fig. 6.1.6.6 Grafo de la solución de Jackson

En nuestro caso (utilizando las denominaciones originales de las piezas):

$$I = 4 \; ; \; h = 1 \; ; c = 1 \; ; J = \{ 2, 3, 4\}$$

Teorema de Carlier (1982)

La solución obtenida mediante el algoritmo de Jackson, si no es óptima, nos da una acotación de la solución óptima:

$$K(J) \leq f^* < K(J) + p_c$$

en efecto, dadas las definiciones anteriores:

$$t_{h-1} + p_{h-1} < t_h = r_h \quad \text{y} \quad r_h = \min_{i \in [h,I]} \{r_i\}$$

pueden darse dos casos:

Caso 1: $q_l \leq q_i$ a todo $i \in [h,l]$

entonces el programa es óptimo ya que el valor de la solución de Jackson, f_o, coincide con la cota inferior $K(J)$ (téngase presente que en este caso $c = h-1$ y $J = [h,l]$):

$$f_o = r_h + \sum_{i \in J} p_i + q_l = \min_{i \in J} \{r_i\} + \sum_{i \in J} p_i + \min_{i \in J} \{q_i\} = K(J)$$

que en cierta forma constituye una nueva demostración del teorema de Jackson en el caso de que todas las r_i son iguales.

Caso 2:

$$q_1 > \min_{i \in [h,l]} \{q_i\}$$

dados los condicionantes, tenemos que $q_c < q_i$ para todo i perteneciente al conjunto crítico J (aunque también $q_l \leq q_i$ para todo $i \in J$) ; así mismo $r_c \leq t_c < r_i$ para todo i perteneciente a J (de lo contrario en t_c además de la operación sobre c podrían empezar otras con una valor de q mayor, lo que está en contradicción con los supuestos del algoritmo de Jackson). Puesto que:

$$t_c = t_h + p_h + \ldots + p_{c-1} = r_h + p_h + \ldots + p_{c-1}$$

tenemos:

$$K(J) = \min_{i \in J} \{r_i\} + \sum_{i \in J} p_i + \min_{i \in J} \{q_i\} > t_c + \sum_{i \in J} p_i + q_1 =$$

$$= r_h + \sum_{i \in [h,l]} p_i + q_l - p_c = f_o - p_c$$

Por tanto:

$$f_o < K(J) + p_c$$

y como:

$$K(J) \leq f^* \leq f_o$$

tenemos:

$$K(J) \leq f^* < K(J) + p_c$$

y suponiendo enteras las duraciones:

$$K(J) \leq f^* \leq K(J) + p_c - 1$$

En nuestro ejemplo $K(J) = 11 + 17 + 21 = 49$; $p_c = 5$

$$49 \leq f^* \leq 49 + 5 - 1 = 53$$

Corolario. En una secuencia óptima *c* se ejecutará o bien *antes* que todas las piezas de *J*, o bien *después* de todas ellas. En caso contrario se formaría un camino de 0 a $n+1$ de longitud no inferior a $K(J) + p_c$ y la solución no podría ser óptima.

El teorema de Carlier muestra el interés de la solución de Jackson, que proporciona una cota inferior $K(J)$ y otra superior f_o, ambas excelentes. Además si se autoriza la preempción (la interrupción de una operación una vez empezada) el problema relajado se resuelve polinomialmente modificando ligeramente el algoritmo:

- una vez empezada una operación, la de la pieza *i* por ejemplo, se interrumpe si se hace disponible (en r_h) una pieza *h* con $q_h > q_i$; se inicia entonces *h* y se devuelve *i* a la lista de piezas pendientes con la duración restante $(p_i + t_i - r_h)$

Esta solución tiene el valor max $\{ K(I_1) \}$ con I_1 un subconjunto, no necesariamente propio, de *I*, y se calcula fácilmente. Este cálculo permite mejorar la cota inferior en una exploración arborescente. Para mejorar la cota superior f_o una técnica muy eficaz es la siguiente: se parte de la solución de Jackson, después se desplaza la pieza *c* ejecutando su operación después de todas las piezas de *J*, sin modificar el orden de ejecución del resto de piezas. Carlier afirma que en la mayoría de los casos una de las dos soluciones es óptima (aunque éste no es el caso en el ejemplo).

Propiedad de las soluciones de duración inferior a f_1

Sea el subconjunto $U = \{ u \in I - J \mid p_u > f_1 - K(J) \}$, la operación de cualquier pieza del mismo debe ejecutarse o bien *antes* o bien *después* que todas las operaciones de las piezas de J, para obtener una solución de valor inferior a f_1. La demostración es análoga a la que hemos visto para el corolario.

Proposición 1. Si

$$u \in U \quad \text{y} \quad r_u + p_u + \sum_{i \in J} p_i q_i \geq f_1$$

entonces, en una solución de duración *estrictamente* inferior a f_1, la operación de la pieza *u* se ejecutará después de todas las operaciones de las piezas de *J*. Se podrá ajustar la fecha de disponibilidad de *u* haciendo r_u igual a:

$$\max \{ r_u, \min_{i \in J} r_i + \sum_{i \in J} p_i \} \Rightarrow r_u$$

en efecto, si la operación de *u* se ejecutara antes que las operaciones de las piezas de *J* se produciría un camino de longitud superior a f_1 en el grafo asociado; la duración de la

solución sería, por tanto, superior a f_1.

Proposición 2. Si

$$u \in U \quad \text{y} \quad \min_{i \in J} r_i + p_u + \sum_{i \in J} p_i + q_u \geq f_1$$

entonces, en una solución de duración *estrictamente* inferior a f_1, la operación de u se ejecutará antes de las operaciones de todas las piezas de J. Se podría ajustar la duración de latencia de u haciendo q_u igual a:

$$\max \{q_u, \min_{i \in J} q_i + \sum_{i \in J} p_i\}$$

la demostración es idéntica a la anterior.

Método de exploración arborescente

Vamos a resolver el problema 1-máquina mediante una exploración arborescente (semejante al método de Lomnicki). Los vértices del árbol corresponden a aquellas soluciones del problema (secuencias o permutaciones de las n piezas) que cumplen ciertas condiciones suplementarias en cuanto al orden relativo de ciertas parejas de piezas. Se supone que el hecho de que las r_i sean diferentes introduce una fuerte distinción en cuanto a la calidad de las diferentes permutaciones, por lo que sólo conviene tener en cuenta explícitamente unas pocas. Por consiguiente, cierto punto de vista consiste en considerar que en cada vértice del árbol tenemos que resolver un problema deducido del inicial añadiéndole las restricciones adicionales.

El problema P asociado a un vértice S del árbol también es un problema 1-máquina del que escribimos (sin riesgo de ambigüedad) los datos como $[r_i, p_i, q_i]$ y $g(S)$ la cota inferior, que preferentemente se habrá obtenido a partir de la solución del problema preemptivo relajado, aunque también puede utilizarse $K(J)$, a pesar de que sea una cota menos fina.

Sistemáticamente tomaremos el vértice pendiente S del árbol con la menor cota $g(S)$ y aplicaremos al problema 1-máquina asociado la regla de Jackson, calculando c y J; luego compararemos f_o con el valor f-mejor. Si $f_o < f$-mejor haremos f-mejor $= f_o$; y si la solución de Jackson fuese localmente óptima ($c = h-1$) entonces cerraremos el vértice (el vértice es terminal, no tiene siguientes, y por tanto deja de estar pendiente). En caso contrario construiremos la solución obtenida ejecutando la operación de c después de las de J, conservando el orden del resto de operaciones; compararemos el valor f_1 con f-mejor, y si $f_1 < f$-mejor haremos f-mejor $= f_1$. Construiremos los dos vértices siguientes de S aplicando las proposiciones anteriores.

- vértice correspondiente al problema *antes*, en el que se impone que la operación de c se

ejecute antes que todas las operaciones de J haciendo q_c igual a:

$$\max \{q_c, \sum_{i \in J} p_i + q_l\}$$

- vértice correspondiente al problema *después*, en el que se impone que la operación c se ejecute después que todas las operaciones de J haciendo r_c igual a:

$$\max \{r_c, \min_{i \in J} r_i + \sum_{i \in J} p_i\}$$

Este método de separación es convergente pues conduce a arbitrar el conjunto de las disyunciones $[c,j]$ $(j \epsilon J)$ en el sentido (c,j) en el problema "antes" y en el sentido (j,c) en el problema "después". Se construye así, paso a paso, una secuencia (con ligaduras potenciales) en la máquina.

Determinamos las cotas inferiores de ambos problemas, $g(S_1)$ y $g(S_2)$; si $g(S_1) \geq f$-mejor, cerramos el vértice S_1, y análogamente con S_2. Cada vez que obtengamos un nuevo valor de f-mejor cerramos todos los vértices pendientes S tales que $g(S) \geq f$-mejor. Cuando todos los vértices estén cerrados, el algoritmo se termina y hemos hallado la solución óptima. Inicialmente podemos tomar como f-mejor infinito o la solución de Jackson del problema inicial.

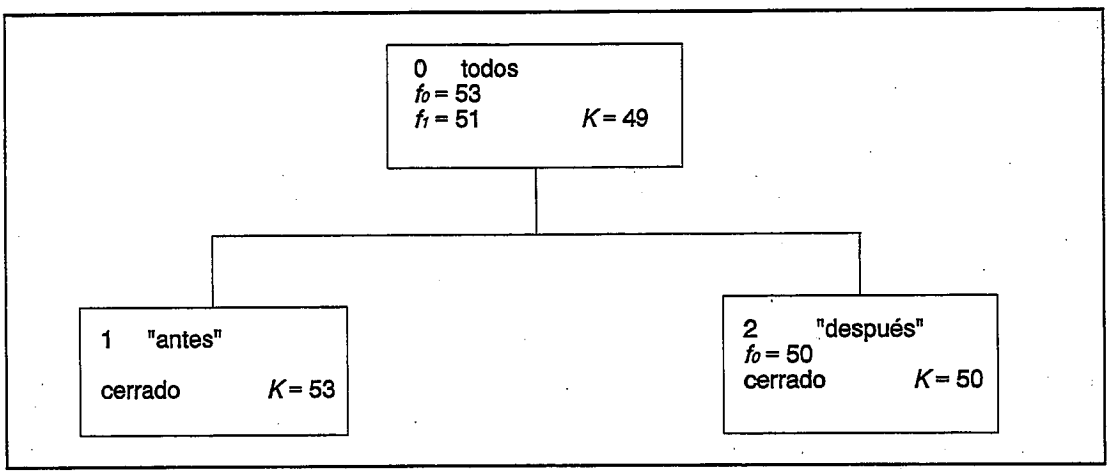


Fig. 6.1.6.7 Árbol de resolución del problema

En la figura 6.1.6.7 hemos representado el árbol de resolución del problema. El vértice inicial (numerado 0 con etiqueta "todos") corresponde al problema de partida, del que conocemos la cota $K(J) = 49$, y el valor $f_o = 53$. Por tanto inicialmente f-mejor $= 53$.

Situando la pieza crítica $c=1$ detrás de las de J obtenemos:

Secuencia	6	2	3	4	1	5	7
r_i	0	13	11	20	10	30	30
t_i	0	13	19	26	30	35	38
p_i	6	6	7	4	5	3	2
c_i	6	19	26	30	35	38	40
q_i	17	26	24	21	7	8	0
$c_i + q_i$	23	45	50	51	42	46	40

por tanto $f_1=51$, y f-mejor$=51$. En el problema "antes", vértice 1, hacemos $q_7 = \max\{7, 6+7+4+21\} = 38$ (con lo cual adoptando $I_1 = \{1\}$ tenemos $K(I_1) = 10 + 5 + 38 = 53 > f$-mejor; debemos cerrar el vértice). En el problema "después" hacemos $r_1 = \max\{10, 11+6+7+4\} = 28$. Para determinar su cota efectuaremos el cálculo de la solución relajada.

Los datos vigentes los hemos reunido en la tabla 6.1.6.8, en la que hemos ordenado las piezas por r_i creciente, y en caso de empate por q_i decreciente.

i	6	3	2	4	1	5	7
r_i	0	11	13	20	28	30	30
p_i	6	7	6	4	5	3	2
q_i	17	24	26	21	7	8	0

Fig. 6.1.6.8 Datos del problema 7X1SD en el vértice 2

La programación de la máquina será como sigue:

de 0 a 6 operación de la pieza 6 ($c_6=6$; $c_6+q_6=23$)
de 11 a 13 operación de la pieza 3 (interrumpida)
de 13 a 19 operación de la pieza 2 ($c_2=19$; $c_2+q_2=45$)
de 19 a 24 operación de la pieza 3 ($c_3=24$; $c_3+q_3=48$)
de 24 a 28 operación de la pieza 4 ($c_4=28$; $c_4+q_4=49$)
de 28 a 30 operación de la pieza 1 (interrumpida)
de 30 a 33 operación de la pieza 5 ($c_5=33$; $c_5+q_5=41$)
de 33 a 36 operación de la pieza 1 ($c_1=36$; $c_1+q_1=43$)
de 36 a 38 operación de la pieza 7 ($c_7=38$; $c_7+q_7=38$)

de donde $\max\{ K(I_1) \mid I_1 \in I \} = 49$ (basta tomar, como antes, $I_1 = \{2,3,4\}$). Por tanto, inicialmente, la cota del vértice 2 es 49, dicho vértice queda abierto, es el único pendiente y procedemos a explorarlo. La secuencia de Jackson es:

6 - 3 - 2 - 4 - 1 - 5 - 7

representada en el grafo de la figura 6.1.6.9

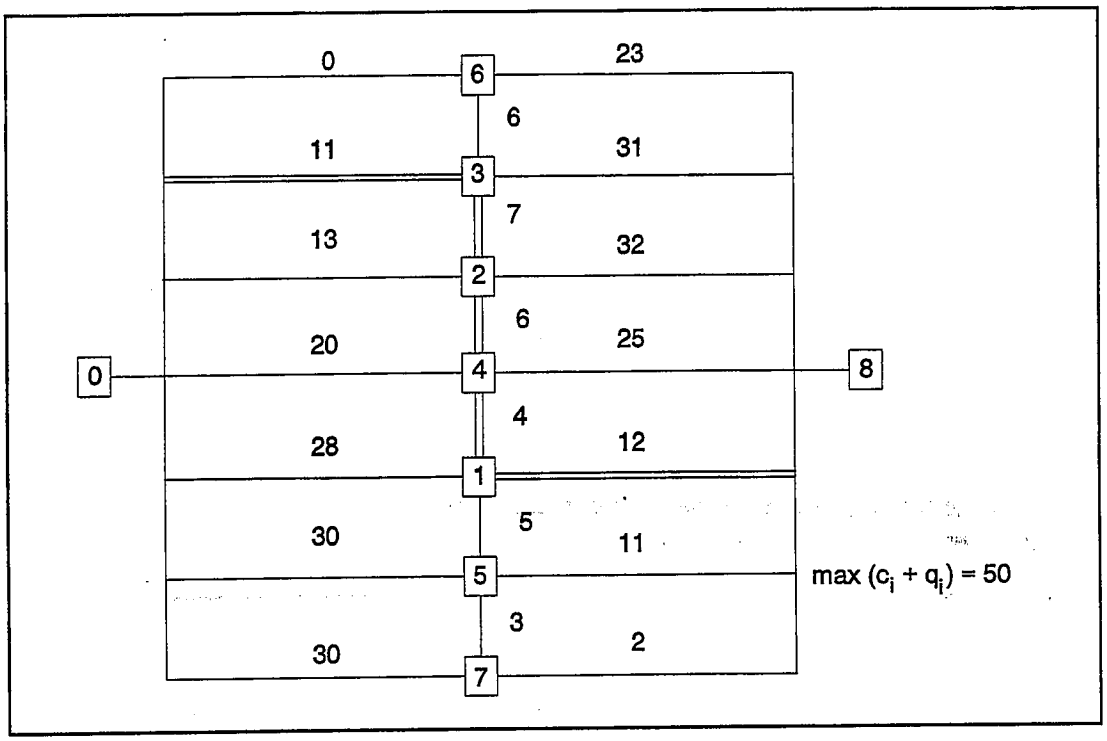


Fig. 6.1.6.9 Grafo de la solución de Jackson para el problema del vértice 2

Tenemos ahora: $f_o = 50$; $h = 3$; $l = 1$; $c = 6$; $J = \{ 3, 2, 4, 1 \}$; $K(J) = 50$ y f-mejor = 50; por tanto, disponemos de la solución óptima. Podemos cerrar el vértice.

El procedimiento de Carlier, a pesar de recurrir a la exploración arborescente, alcanza la solución óptima en un número de pasos y en un tiempo razonables.

6.1.6.3 Problema $n/m/F/c_{max}$: trapecios dinámicos

En el caso $n/m/F/c_{max}$ (recuérdese que c_{max} es el instante de terminación de la última pieza; si 0 es el inicio de la primera operación, c_{max} marca el tiempo total de funcionamiento del taller, y hace el mismo papel que F_{max} en los problemas estáticos) con valores de $r_i = r_{1,i}$ no todos nulos podemos utilizar una variante de las heurísticas Palmer y trapecios vistas anteriormente. Los valores $S_{1,i}$, $S_{2,i}$ y $S_{3,i}$ serán para cada pieza idénticos a los ya considerados:

$$S_{1,i} = \sum_{j=1}^{m-1} (m - j) \cdot p_{j,i}$$

$$S_{2,i} = \sum_{j=2}^{m} (j-1) \cdot p_{j,i}$$

$$S_{3,i} = S_{1,i} - S_{2,i}$$

a partir de los cuales determinaremos unos valores dinámicos $SD_{1,i}$, $SD_{2,i}$ y $SD_{3,i}$ que variarán a lo largo de la aplicación del algoritmo. Como de costumbre, f_j es el instante de disponibilidad de la máquina j (inicialmente tomará el valor correspondiente al instante en que la máquina puede empezar a tratar las operaciones de las piezas consideradas, progresivamente variará al ir programando operaciones). Las operaciones se dividen en tres clases: P, las de la primera máquina ya programadas y E las todavía pendientes de programar de dicha máquina. En la tercera clase, N, estarán las operaciones en las demás máquinas, no consideradas por el momento (de la 2 a la m). Los pasos son los siguientes:

1) Determinar $rp_{1,i}$, instante en que puede empezar la operación $(1,i)$, para todas las operaciones pendientes:

$$rp_{1,i} = \max \{ r_{1,i}, f_1 \} \text{ para todo } (1,i) \in E$$

2) Determinar fp_j, instante en que la máquina 1 puede iniciar una nueva operación:

$$fp_1 = \min \{ rp_{1,i} \} \text{ para todo } (1,i) \in E$$

3) Determinar ff_j, instante más temprano en que puede quedar libre la máquina 1 si lanzamos una nueva operación:

$$ff_1 = \min \{ rp_{1,i} + p_{1,i} \} \text{ para todo } (1,i) \in E$$

4) Determinar el conjunto E' de operaciones que pueden conducir a un programa activo (sin huecos de tiempo muerto aprovechables por una operación pendiente):

$$E' = \{ (1,i) \in E \text{ y } rp_{1,i} < ff_1 \}$$

5) Si en E' existe una sola operación se programa, si existe más de una calcular para cada una de ellas los valores dinámicos:

$$SD_{1,i} = S_{1,i} + m.[rp_{1,i} - fp_1]$$

$$SD_{2,i} = S_{2,i} - m.[rp_{1,i} - fp_1]$$

$$SD_{3,i} = SD_{1,i} - SD_{2,i}$$

para todo $(1,i) \in E'$

Aplicar el algoritmo de ordenación de Johnson a $SD_{1,i}$ y $SD_{2,i}$ para determinar la primera operación; en caso de ambigüedad se da preferencia al valor $SD_{3,i}$ menor, y si subsiste el empate se elige la operación con $rp_{1,i}$-fp_1 menor, y posteriormente con $p_{1,i}$ menor.

(esta regla, al ser la tercera de las diseñadas para este caso recibió el nombre de *regla C*; las pruebas realizadas con problemas generados aleatoriamente han mostrado su adecuado comportamiento en una proporción estimable de casos).

6) Pasar la operación programada de *E* a *P*. Actualizar el valor f_1, si *(1,h)* es la operación elegida:

$$f_1 = c_{1,h} = rp_{1,h} + p_{1,h}$$

siendo, por tanto, $c_{1,h}$ el instante de terminación de la operación *(1,h)*, y por consiguiente, como veremos más adelante el de disponibilidad para la operación *(2,h)*.

7) Si *E* está vacío hemos terminado con la programación de la máquina 1. En caso contrario volver a (1).

Si la secuencia hallada debe mantenerse en todas las máquinas, una dificultad que puede presentarse es la relativa a la posición de aquellas piezas, por ejemplo la *h*, que no tienen operación en la primera máquina, es decir, $p_{1,h}=0$. Pueden utilizarse dos procedimientos en el caso de secuencia única para todas las máquinas:

i) Considerar una operación ficticia, *(1,h)*, de duración nula, y situarla donde corresponda según las reglas anteriores,

ii) Considerar para definir la posición de h los instantes de disponibilidad de las operaciones en la segunda máquina, es decir, los valores $r_{2,i}$, para los que tendremos:

$$r_{2,i} = c_{1,i} \quad \text{si } p_{1,i} > 0;$$
$$r_{2,h} = r_{1,h} \quad \text{si } p_{1,h} = 0.$$

y secuenciarla en la segunda máquina (e implícitamente en la primera) según un orden FIFO.

Normalmente utilizaremos el segundo procedimiento. Puesto que las duraciones nulas pueden no limitarse a la primera máquina, la extensión de la regla FIFO para establecer la secuencia en las máquinas 2, 3, ..., *m* a partir de la secuencia en la primera máquina aparece como un procedimiento razonable, y no sólo en los problemas semidinámicos.

Sin embargo, si no existe la limitación de emplear la misma secuencia en todas las

máquinas se presenta la posibilidad de no limitar el uso de las reglas de los "trapecios dinámicos" a la secuenciación de la primera máquina. En efecto, una vez establecida la secuencia en la primera máquina podemos definir el instante de comienzo y terminación de las operaciones en la misma, y por tanto el instante de disponibilidad de las operaciones en la segunda máquina; la situación será, respecto a la segunda máquina, semejante en todo a la que teníamos anteriormente respecto a la primera (salvo que ahora las máquinas que falta por programar son *m-1* y la reestructuración de las duraciones de las operaciones). El procedimiento descrito anteriormente puede aplicarse a la secuenciación de todas las máquinas (con la excepción de la última). Actuaremos máquina a máquina, es decir, una vez establecida la secuencia para la máquina *j* tendremos los instantes de disponibilidad de las piezas para su operación en la máquina $j+1$, $r_{j+1,i}=c_{j,i}$, donde $c_{j,i}$ es el instante de terminación de la operación en la máquina *j* de la pieza *i*, y procederemos a determinar la secuencia para dicha máquina. Se exceptúa la máquina *m*, cuya secuencia coincidirá con la de la máquina *m-1* (este sería el resultado de aplicar el procedimiento). Visto el establecimiento de la secuencia en la primera máquina, el procedimiento para las máquinas 2, 3, ..., *m-1* será idéntico, salvo la consideración de que existe una máquina menos para secuenciar. Si una determinada pieza nó tiene operación en una máquina $p_{j,i}=0$, entonces no se tendrá en cuenta en la máquina j dicha pieza y tomaremos:

$$r_{j+1,i} = r_{j,i} = c_{j-1,i}$$

Los valores $S_{1,i}$, $S_{2,i}$ y $S_{3,i}$ cuando estemos programando la máquina j serán:

$$S_{1,i} = \sum_{l=j}^{m-1} (m-l) \cdot p_{l,i}$$

$$S_{2,i} = \sum_{l=j+1}^{m} (l-j) \cdot p_{l,i}$$

$$S_{3,i} = S_{1,i} - S_{2,i}$$

Los pasos ahora son los siguientes:

1) Empezar por la primera máquina ($j=1$). Situar en *E* todas las operaciones $(1,i)$, en *N* el resto y ajustar f_1.

2) Determinar $rp_{j,i}$, instante en que puede empezar la operación (j,i), para todas las operaciones pendientes:

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

3) Determinar fp_j, instante en que la máquina *j* puede iniciar una nueva operación:

$$fp_j = \min \{ rp_{j,i} \} \text{ para todo } (j,i) \in E$$

4) Determinar ff_j, instante más temprano en que puede quedar libre la máquina j si lanzamos una nueva operación:

$$ff_j = \min \{ rp_{j,i} + p_{j,i} \} \text{ para todo } (j,i) \in E$$

5) Determinar el conjunto de operaciones que pueden conducir a un programa activo E':

$$E' = \{ i \in E \text{ y } rp_{j,i} < ff_j \}$$

6) Si en E' existe una sola operación se programa, si existe más de una se calcular para cada una de ellas los valores dinámicos:

$$SD_{1,i} = S_{1,i} + (m-j+1).[rp_{j,i} - fp_j]$$

$$SD_{2,i} = S_{2,i} - (m-j+1).[rp_{j,i} - fp_j]$$

$$SD_{3,i} = SD_{1,i} - SD_{2,i}$$

para todo $(j,i) \in E'$

Aplicar el algoritmo de ordenación de Johnson a $SD_{1,i}$ y $SD_{2,i}$ para determinar la primera operación; en caso de ambigüedad se da preferencia al valor $SD_{3,i}$ menor, y si subsiste el empate se elige la operación con $rp_{j,i}$-fp_j menor, y posteriormente con $p_{j,i}$ menor.

7) Pasar la operación programada de E a P. Actualizar el valor f_j, si (j,h) es la operación elegida:

$$f_j = rp_{j,h} + p_{j,h}$$

Si todavía quedan operaciones en E, pasar a (2).

8) Si E está vacío hemos terminado la programación de la máquina j. Si esta máquina era la penúltima ($j = m-1$) pasar a (10), en caso contrario, ($j < m-1$) pasar a (9).

9) Tomar como j el valor siguiente $j+1$. Pasar las operaciones (j,i) de N a E, reestructurar los valores f_j, $r_{j,i}$, $S_{1,i}$, $S_{2,i}$ y $S_{3,i}$ y volver a (2).

10) Secuenciar las operaciones en la máquina m según la regla FIFO, es decir, en el orden de disponibilidad $r_{m,i}$. El algoritmo ha terminado.

Máquinas j disponibles f_j		A 0	B 4	C 8	D 10	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_i = r_{1,i}$							
1	4	2	4	3	5	17	25	-8
2	45	6	4	8	2	34	26	8
3	0	7	6	2	5	35	25	10
4	6	2	3	12	4	24	39	-15
5	10	3	4	2	5	19	23	-4
6	22	8	10	2	6	46	32	14
7	16	7	2	3	4	28	20	8
8	14	4	3	7	2	25	23	2
9	35	6	5	4	3	32	22	10
10	0	3	5	7	6	26	37	-11

Fig. 6.1.6.10 Datos del problema AX4SD

La aplicación manual de este algoritmo, aunque posible, no es cómoda; sin embargo, a través de ordenador no ofrece la menor dificultad. Los resultados numéricos que se adjuntan (figura 6.1.6.11) se han obtenido mediante la utilización del programa TRADIN32.

Obsérvese que el mismo procedimiento descrito puede utilizarse en problemas "estáticos", es decir, con todas las máquinas y piezas disponibles en el instante 0; naturalmente solo será interesante hacerlo cuando $m > 3$. La aplicación del procedimiento al problema 4114 (ver figuras 6.1.5.4 y 6.1.6.12) conduce a la solución óptima con $F_{max} = 12$.

Máq. A	Sec. f 0	10 3	3 10	1 12	5 15	4 17	8 21	6 30	7 37	9 43	2 51
Máq. B	Sec. f 4	10 8	3 16	5 20	1 24	4 27	8 30	6 40	7 42	9 48	2 55
Máq. C	Sec. f 8	10 15	3 18	5 22	1 27	4 39	6 42	7 45	8 52	9 56	2 64
Máq. D	Sec. f 10	10 21	3 26	5 31	1 36	4 43	6 49	7 53	8 55	9 59	2 66

Fig. 6.1.6.11 Solución dada por Tradin32 al problema AX4SD (c_{max} = 66)

Máquinas j disponibles f_j		A 0	B 0	C 0	D 0	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_i=r_{1,i}$							
1 2	0 0	4 1	1 4	1 4	4 1	15 15	15 15	0 0

Fig. 6.1.6.12 Datos del problema 4114

Máquina $j=1$

Al ser $r_1 = r_2 = 0$ el método de los trapecios dinámicos se reduce al procedimiento original de los trapecios ($ff=1$, por tanto $E' = \{ 1,2 \}$, $fp_1=rp_{1,1}=rp_{1,2}=0$); existe empate entre ambas piezas tanto respecto al criterio principal, como respecto al secundario, debiendo recurrir al terciario que da ventaja a la pieza 2, ya que $p_{1,2}<p_{1,1}$. En consecuencia:

Máq. A	Sec. f	2 1	1 5

Máquina $j=2$

El problema es ahora tal como indica la tabla de la figura 6.1.6.13; puesto que $ff=5$, inicialmente no existe conflicto pues $E' = \{ 2 \}$, y una vez programada la pieza 2, solo queda la 1 para seguir a continuación.

Máquinas j disponibles f_j		B 0	C 0	D 0	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_i=r_{2,i}$						
1 2	5 1	1 4	1 4	4 1	3 12	9 6	-6 6

Fig. 6.1.6.13 Datos del problema 4114 para $j=2$

Máq. B	Sec. f	2 5	1 6

Máquina $j=3$

Los datos del problema adoptan ahora la forma de la tabla de la figura 6.1.6.14; $ff=7$ por lo que $E' = \{ 2 , 1 \}$ y debemos determinar los valores dinámicos:

$$SD_{1,1} = 1 + 2\cdot(6-5) = 3 \; ; SD_{1,2} = 4$$
$$SD_{2,1} = 4 - 2\cdot(6-5) = 2 \; ; SD_{2,2} = 1$$

por consiguiente la pieza 1 se programa antes que la 2

Máquinas j disponibles f_j		C 0	D 0	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_i=r_{2,i}$					
1	6	1	4	1	4	-3
2	5	4	1	4	1	3

Fig. 6.1.6.14 Datos del problema 4114 para $j=3$

Máq. C	Sec. f	1 7	2 11

Máquina $j=4$

Como es la última máquina, se conserva el orden de la penúltima:

Máq. D	Sec. f	1 11	2 12

Este valor de c_{max} es óptimo ya que coíncide con el mejor valor de una cota transversal.

En un ejemplo más complejo (5X4SD, figura 6.1.6.15) podemos ver las dificultades del cálculo manual. Como todas las piezas y máquinas están disponibles en el instante 0, la secuencia en la primera máquina vendrá dada por el orden trapecios:

2 - 4 - 1 - 3 - 5

Máquinas j disponibles f_j		A 0	B 0	C 0	D 0	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_i=r_{1,i}$							
1	0	9	14	15	1	70	47	23
2	0	5	9	1	8	34	35	-1
3	0	11	5	7	8	50	43	7
4	0	6	19	15	1	71	52	19
5	0	16	3	4	5	58	26	32

Fig. 6.1.6.15 Datos del problema 5X4SD

Si mantuviésemos dicho orden en las cuatro máquinas obtendríamos $F_{max} = 83$. La situación después de la primera máquina es la que corresponde a la figura 6.1.6.16.

Máquinas j disponibles f_j		B 0	C 0	D 0	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_{2,i}$						
2	5	9	1	8	19	17	2
4	11	19	15	1	53	17	36
1	20	14	15	1	43	17	26
3	31	5	7	8	17	23	-6
5	47	3	4	5	10	14	-4

Fig. 6.1.6.16 Datos del problema 5X4SD en la máquina B

Máquina B			$m - j + 1 = 3$								
f	fp	ff	E'	r	rp	tm	p	$SD_{1,i}$	$SD_{2,i}$	$SD_{3,i}$	elegida
0	5	14	2	5	5	0	9	19	17	2	→ 2
			4	11	11	6	19	35	-1	36	
14	14	33	4	11	14	0	19	53	17	36	→ 4
			1	20	20	6	14	61	-1	62	
			3	31	31	17	5	68	-28	96	
33	33	38	1	20	33	0	14	43	17	26	→ 3
			3	31	33	0	5	17	23	-6	
38	38	50	1	20	38	0	14	43	17	26	→ 1
			5	47	47	9	3	37	-13	50	
52	52	55	5	47	52	0	3				→ 5

Fig. 6.1.6.17 Cálculo de la secuencia en la máquina B

Los cálculos los desarrollamos con la ayuda de la tabla de la figura 6.1.6.17 que está dividida en bandas horizontales, cada una de las cuales corresponde a la selección de una pieza. Los datos indicados se corresponden a los descritos en la exposición anterior, siendo:

$$tm_{j,i} = rp_{j,i} - fp_j$$

que nos permite determinar el valor de *SD*. La situación delante de la tercera máquina, C, es la indicada en la figura 6.1.6.18 y los cálculos se desarrollan en la 6.1.6.19.

Máquinas j disponibles f_j		C 0	D 0	$S_{1,i}$	$S_{2,i}$	$S_{3,i}$
i	$r_{2,i}$					
2	14	1	8	1	8	-7
4	33	15	1	15	1	14
3	38	7	8	7	8	-1
1	52	15	1	15	1	14
5	55	4	5	4	5	-1

Fig. 6.1.6.18 Datos del problema 5X4SD en la máquina C

Máquina C			$m - j + 1 = 2$								
f	fp	ff	E'	r	rp	tm	p	$SD_{1,i}$	$SD_{2,i}$	$SD_{3,i}$	elegida
0	14	15	2	14	14	0	1				→ 2
15	33	45	4 3	33 38	33 38	0 5	15 7	15 17	1 -2	14 19	→ 4
48	48	55	3 1	38 52	48 52	0 4	7 15	7 23	8 -7	-1 30	→ 3
55	55	59	1 5	52 55	55 55	0 0	15 4	15 4	1 5	14 -1	→ 5
59	59	74	1	52	59	0	15				→ 1

Fig. 6.1.6.19 Cálculo de la secuencia en la máquina C

Máq. A	Sec. f 0	2 5	4 11	1 20	3 31	5 47
Máq. B	Sec. f 0	2 14	4 33	3 38	1 52	5 55
Máq. C	Sec. f 0	2 15	4 48	3 55	5 59	1 74
Máq. D	Sec. f 0	2 23	4 49	3 63	5 68	1 75

Fig. 6.1.6.20 Solución obtenida del problema 5S4SD

Para la máquina D repetimos la secuencia hallada para la máquina C, y la solución hallada se resume en la figura 6.1.6.20.

6.1.7. Problemas de flujo general *n/m/G* estáticos

Podemos visualizar intuitivamente este tipo de problemas como una especie de rompecabezas gráfico consistente en situar un conjunto de rectángulos, representativos de las operaciones, dentro del marco de un diagrama de Gantt, satisfaciendo ciertas ligaduras (potenciales y disyuntivas). Las operaciones correspondientes a cada pieza *i* constituyen *g(i)* rectángulos, uno para cada una de ellas, de longitud proporcional a la duración de la operación. Cada rectángulo u operación está asociado a tres identificadores *k/i/j*; siendo *i* el número (o identificador) de la pieza, *k* el número de orden de la operación en la secuencia de operaciones de la pieza y *j* el identificador de la máquina en la que se realiza la operación (función de *i* y de *k*).

En la figura 6.1.7.1 hemos representado un problema *5/3/G* (las duraciones, representadas por las longitudes, son pieza 1: 3-6-3, pieza 2: 2-5-2-7. pieza 3: 5-7-3, pieza 4: 4-6-7-4, pieza 5: 2-6). En cada fila tenemos las operaciones correspondientes a una misma pieza. En la figura 6.1.7.2 hemos reproducido las mismas operaciones, pero en este caso clasificadas por máquinas; cada fila corresponde a una máquina. Aquí la disposición de las operaciones no aparece tan clara puesto que el orden definitivo, la secuencia en cada máquina es lo que estamos buscando, mientras que el orden de las operaciones de una pieza está fijado.

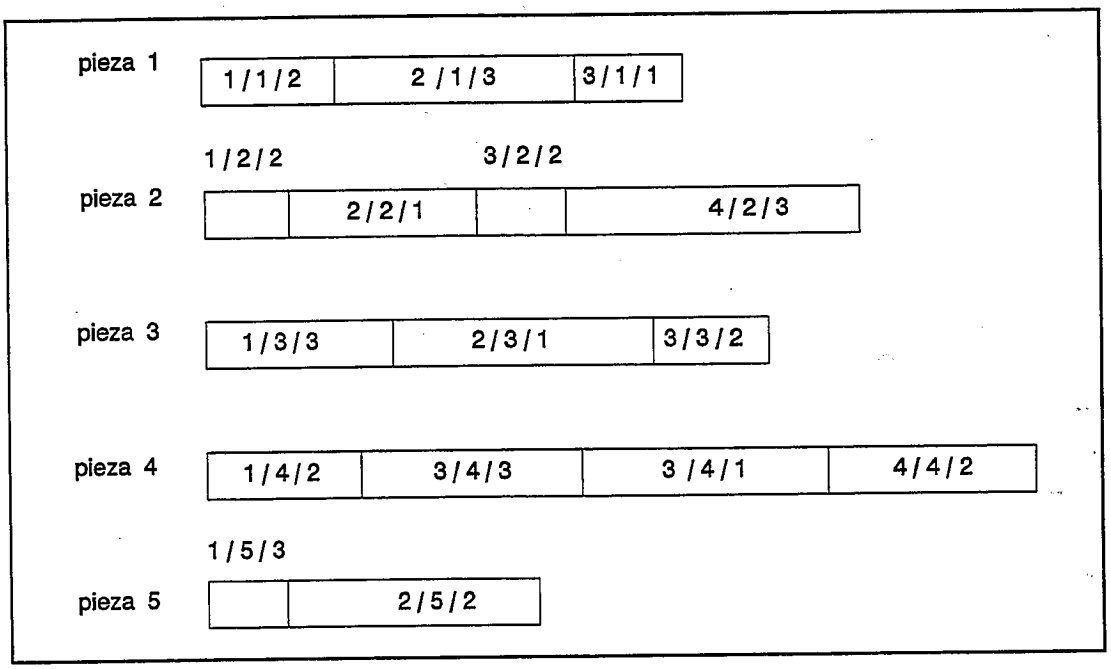


Fig. 6.1.7.1 Problema 5/3/G. Datos ordenados por piezas

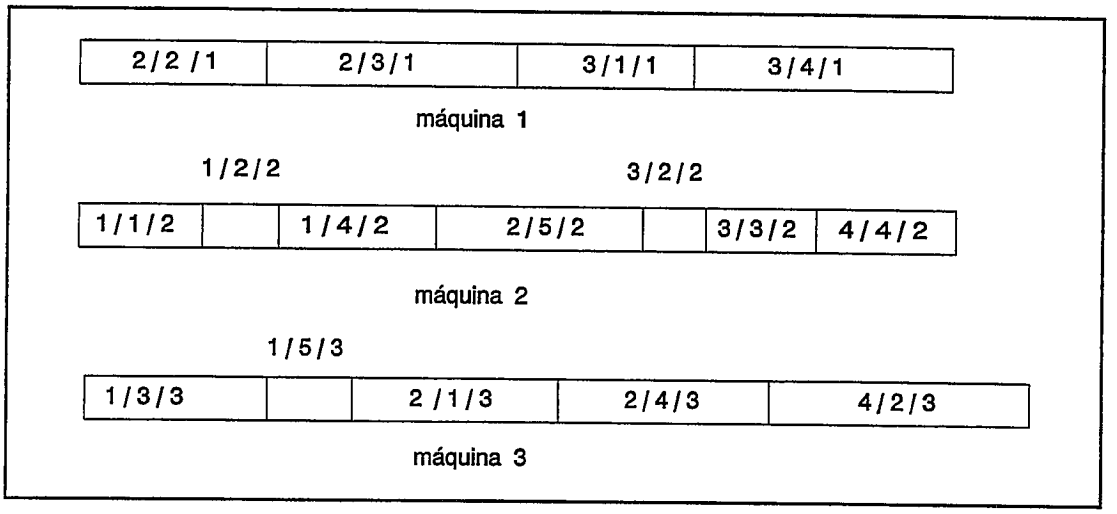


Fig. 6.1.7.2 Problema 5/3/G. Datos ordenados por máquinas

No debe confundirse el diagrama de la figura 6.1.7.2 con un programa; dicha figura *no es un diagrama de Gantt*, puesto que dos operaciones sobre una misma pieza pueden perfectamente tener una parte solapada en la misma vertical (e incluso si una pieza tiene dos operaciones en la misma máquina el orden en que se dispongan no tiene por qué ser el que corresponde a la ruta de la pieza). Si eliminamos los solapamientos y tenemos en cuenta las prioridades, pasamos de la figura 6.1.7.2 a la 6.1.7.3 que *sí es un programa*, aunque no necesariamente muy bueno. El de la figura 6.1.7.4 es una alternativa, óptimo si se busca minimizar F_{max}.

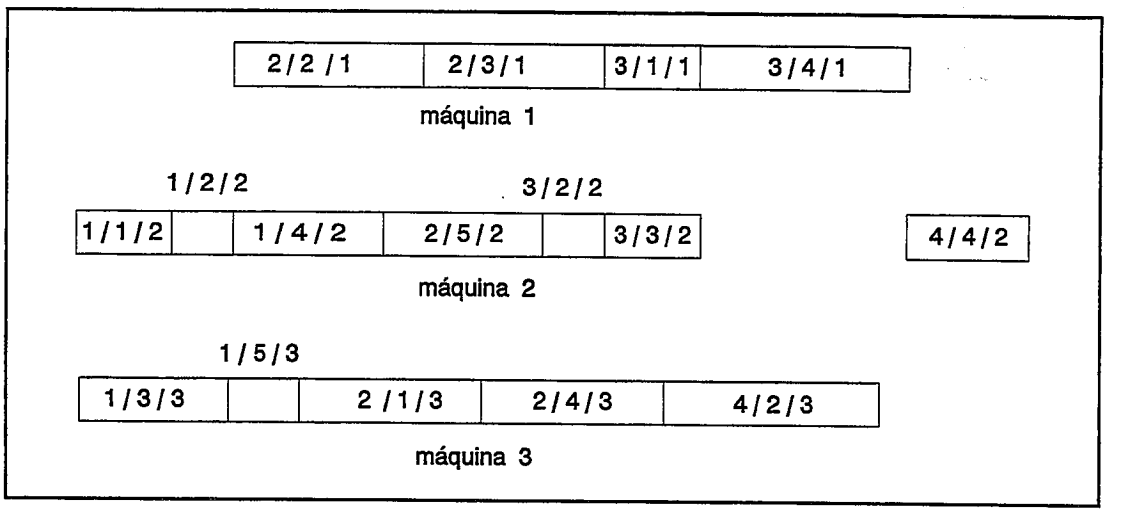


Fig. 6.1.7.3 Problema 5/3/G. Programa correspondiente a la secuencia de la figura 6.1.7.2

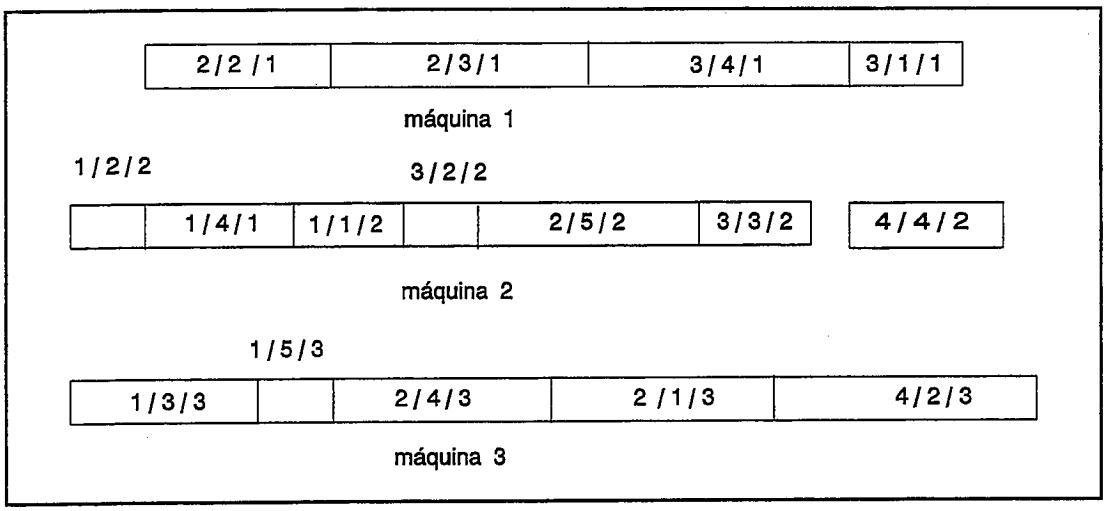


Fig. 6.1.7.4 Problema 5/3/G. Programa óptimo respecto a la minimización de F_{max}

En forma tabular los datos para definir un problema n/m/G tendrán un formato basado en la siguiente notación:

$g(i)$: número de operaciones de la pieza i
$m_{k,i}$: máquina en la que se realiza la operación k-ésima de la pieza i ($k=1,2,....,g(i)$)
$p_{k,i}$: duración de la operación k-ésima de la pieza i ($k=1,2,...,g(i)$)
r_i : instante en que la pieza i entra en el taller,
f_j : instante en que está disponible la máquina j,

En un problema de este tipo la distinción entre el caso estático y el semidinámico no tiene gran trascendencia, por lo que hemos introducido los valores r_i y f_j. En el ejemplo indicado podemos formalizar los datos como se indica en la figura 6.1.7.5.

	Pieza 1 $r_1 = 0$		Pieza 2 $r_2 = 0$		Pieza 3 $r_3 = 0$		Pieza 4 $r_4 = 0$		Pieza 5 $r_5 = 0$	
Op.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur
1	2	3	2	2	3	5	2	4	3	2
2	3	6	1	5	1	7	3	6	2	6
3	1	3	2	2	2	3	1	7		
4			3	7			2	4		

Fig. 6.1.7.5 Datos del ejemplo en el formato definido

Los problemas *n/m/G* son, en general, más complejos que los *n/m/F*, y vamos a ver dos

casos triviales en los que es posible hallar una solución óptima con algoritmos sencillos (el segundo caso es más discutible).

6.1.7.1 Problema $n/2/G/F_{max}$

El único problema del que se conoce una solución estricta sencilla es el problema $n/2/G/F_{max}$ con la restricción especial de que cada pieza puede tener a lo sumo dos operaciones y de que tanto las piezas como las máquinas están disponibles en el instante 0 inicial. Jackson (1956) demostró que podía utilizarse el algoritmo de Johnson con algunas modificaciones de detalle. El algoritmo comienza estableciendo una partición de las n piezas:

- clase A: formada por las piezas que tienen una única operación en la máquina 1;
- clase B: una única operación en la máquina 2;
- clase AB: dos operaciones, primero en la máquina 1, después en la máquina 2;
- clase BA: dos operaciones, primero en la máquina 2, después en la máquina 1;

Se ordenan las piezas de las clases AB y BA utilizando el algoritmo de Johnson, como si fuesen las únicas piezas existentes. En cuanto a las piezas A y B, se ordenan como se quiera. En estas condiciones, las secuencias óptimas para cada máquina serán:

máquina 1: piezas AB + piezas A + piezas BA

máquina 2: piezas BA + piezas B + piezas AB

La razón de estas secuencias es intuitivamente evidente. El programa óptimo podrá no ser único (dada la libertad en las piezas A y B, entre otras cosas).

6.1.7.2 Problema $2/m/G/F_{max}$

El procedimiento gráfico que sigue es aplicable tanto al caso $2/m/G/F_{max}$ como al $2/m/F/F_{max}$. Puede proporcionar una buena solución en los problemas de dimensión adecuada, pero el procedimiento no es ni eficiente, ni tan interesante y realista como el anterior. La realización de ambas piezas se representará mediante una línea que una dos vértices opuestos de un rectángulo (O y Z). Los lados de este rectángulo tienen una longitud proporcional a la duración de las operaciones de cada pieza, $\Sigma\, p_{j,1}$ en horizontal, por ejemplo, y $\Sigma\, p_{j,2}$ en vertical. La línea que une O con Z puede estar compuesta únicamente de segmentos horizontales, verticales o inclinados 45°. Un segmento horizontal indica que se realizan operaciones exclusivamente en la primera pieza, uno vertical que únicamente se trabaja en la segunda, y uno inclinado que simultáneamente se

realizan operaciones en las dos piezas, naturalmente en máquinas distintas. Para impedir que el tramo corresponda a realizar simultáneamente dos operaciones en la misma máquina se disponen unos obstáculos o zonas prohibidas como se puede observar en la figura 6.1.7.7. Para minimizar F_{max} interesa que la mayor parte de la línea a trazar, sino toda, corresponda a trazos a 45° a fin de evitar tiempos muertos en una u otra pieza, o, lo que es lo mismo, que la línea tenga la menor longitud posible, lo que obliga a definir la forma más apropiada de sortear los obstáculos. Sea el ejemplo correspondiente a la figura 6.1.7.7 cuyos datos aparecen en la tabla 6.1.7.6.

	Pieza 1		Pieza 2	
	Máquina	Duración	Máquina	Duración
Operación 1	A	3	B	2
Operación 2	B	4	D	3
Operación 3	C	3	C	4
Operación 4	D	2	A	3

Fig. 6.1.7.6 Datos del problema 2/4/G/F_{max}

Ambas piezas tienen un tiempo total de operación de 12, por lo que si logramos la simultaneidad de todas las operaciones habremos minimizado F_{max}. Sin embargo, en la figura podemos observar que figuran 4 obstáculos, uno correspondiente a cada máquina (ya que cada operación tanto de la pieza 1 como de la 2 se realiza en una máquina distinta, y sólo se produce conflicto entre las parejas de operaciones, una de cada pieza, que se realizan en la misma máquina). Todos los obstáculos, salvo el C, están apartados de la dirección a 45°, pero éste interrumpe forzosamente la diagonal, puede contornearse de dos formas PTUZ o bien PQRSZ, siendo la segunda mejor que la primera (el tramo horizontal SZ es menor que el PT).

En esta forma obtenemos $F_{max}=14$, con 2 unidades de tiempo muerto situadas entre la segunda y la tercera operación de la pieza 1 (y después de la 4ª operación de la pieza 2) tal como puede comprobarse en la figura 6.1.7.8.

En el caso de *flow-shop* (problema *2/m/F/*F_{max}), se procede análogamente, con la particularidad de que las zonas prohibidas ahora quedarán dispuestas preferentemente sobre la diagonal, lo que puede complicar más el problema de encontrar la forma de contornear los obstáculos, ya que crece el número de alternativas a considerar. Naturalmente, se puede generalizar el procedimiento a más dimensiones, pero el problema se visualiza penosamente. En tres dimensiones cabría utilizar un sistema diédrico de representación.

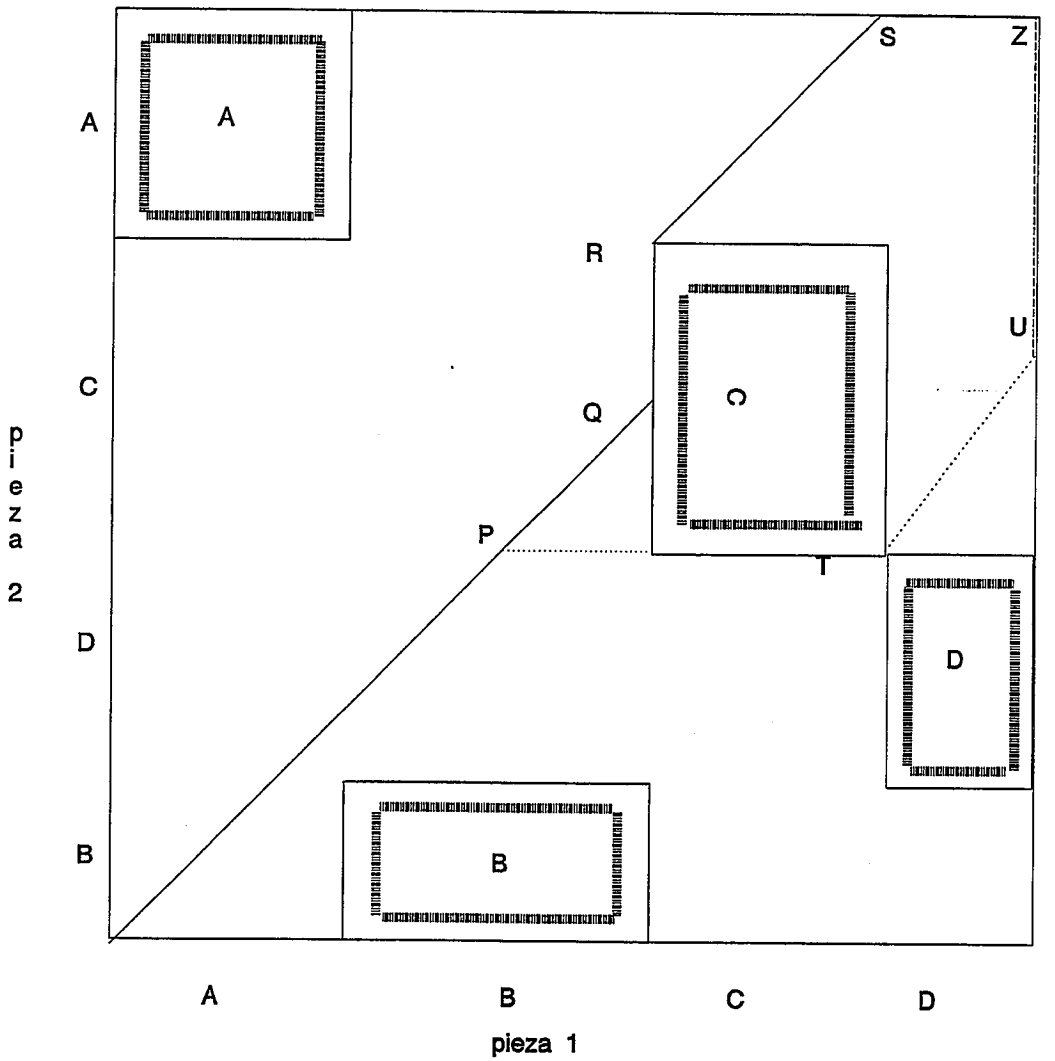


Fig. 6.1.7.7 Determinación gráfica de la solución del problema 2/4/G/F_{max}

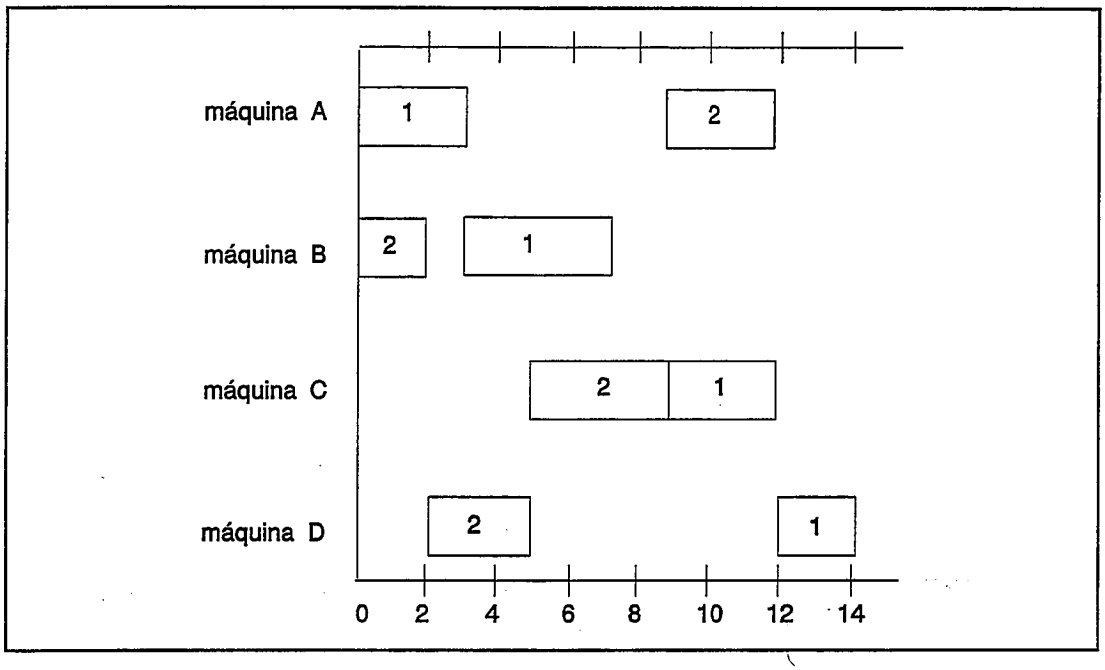


Fig. 6.1.7.8 Diagrama de Gantt de la solución del problema 2/4/G/F_{max}

6.1.7.3 Formulación mediante la programación lineal en variables mixtas

Se han propuesto diversos modelos, debidos (por orden cronológico y sencillez creciente) a Bowman, Wagner, Manne, etc. Vamos a describir el modelo de Manne y, para simplificar la exposición, supondremos que cada pieza pasa por cada máquina una vez y sólo una (lo que permite ahorrar índices y punteros). Sea la siguiente nomenclatura que difiere ligeramente de la utilizada en los otros apartados:

$p_{j,i}$ la duración de la operación de la pieza *i* en la máquina *j*,
$\alpha_{k,i,j}$ = 1 si la operación *k*-ésima de la pieza *i* tiene lugar en la máquina *j*
= 0 en caso contrario (dato)
$f_{j,i}$ instante de finalización de la operación de la pieza *i* en la máquina *j*

Puesto que en una misma máquina, y al mismo tiempo, no pueden desarrollarse dos operaciones, existen ligaduras disyuntivas entre dos piezas cualesquiera *i* y *h* en todas las máquinas:

$$f_{j,i} - f_{j,h} \geq p_{j,i} \text{ o bien } f_{j,h} - f_{j,i} \geq p_{j,h}$$

Este tipo de ligadura para tratarse mediante un modelo lineal precisa de variables binarias (cero/uno). Sea:

$y_{j,i,h}$ = 1 si i precede a h en la máquina j, no necesariamente de forma inmediata,
$y_{j,i,h}$ = 0 en caso contrario; (evidentemente, dada la complementaridad, no necesitamos a la vez $y_{j,i,h}$ e $y_{j,h,i}$),

En consecuencia, podemos escribir:

$$M \cdot y_{j,i,h} + (f_{j,i} - f_{j,h}) \geq p_{j,i}$$

$$M \cdot (1 - y_{j,i,h}) + (f_{j,h} - f_{j,i}) \geq p_{j,h}$$

donde M es una constante suficientemente grande para que sólo una de las dos ligaduras potenciales en las que se descompone la disyuntiva sea activa para $y_{j,i,h}=0$ o 1; por ejemplo:

$$M = \sum_i \sum_j p_{j,i}$$

En cuanto a las ligaduras potenciales tradicionales debidas al orden de las operaciones relativas a la misma pieza, se imponen mediante:

$$\sum_j a_{k,i,j} \cdot p_{j,i} + \sum_j a_{k-1,i,j} \cdot f_{j,i} \leq \sum_j a_{k,i,j} \cdot f_{j,i}$$

para $k=1,2,\ldots,m$ tomando $a_{0,i,j}=0$ a todo i y j.

Con m máquinas y n piezas existen:

$m \cdot n$ variables $f_{j,i}$
$m \cdot n \cdot (n-1)/2$ variables $y_{j,i,h}$
$m \cdot n$ inecuaciones originadas por las ligaduras potenciales
$m \cdot n \cdot (n-1)$ inecuaciones originadas por las disyuntivas

Con 4 máquinas y 10 piezas tenemos 220 variables y 400 inecuaciones.

Si el objetivo es minimizar F_{med}, bastará adoptar como función objetivo:

$$[MIN] \sum_i \sum_j a_{m,i,j} \cdot f_{j,i}$$

si se trata de minimizar F_{max}, deberemos añadir n inecuaciones de la forma:

$$\sum_j a_{m,i,j} \cdot f_{j,i} \leq F_{\max} \qquad i = 1, 2, \ldots, n$$

y adoptar como función objetivo:

$$[MIN]\ F_{max}$$

Aparentemente los intentos concretos de utilización de la programación lineal en variables mixtas, ya reseñados anteriormente, de Wagner & Story y de Wagner & Giglio fracasaron debido al comportamiento caprichoso de los paquetes informáticos de PLM existentes y su consumo de tiempo. Como todo ello ocurría hacia 1960, cabría esperar que treinta años después, con nuevas generaciones de ordenadores y de paquetes las cosas hubiesen cambiado. Sin embargo, parece que los problemas solubles por esta vía, o son muy particulares, o son casos triviales. Probablemente es más interesante el empleo *directamente* sobre la estructura del problema de las técnicas de exploración arborescente (como hace Balas), usuales en los paquetes de PLM, sin transformar dicha estructura para adaptarla al formato de la PLM.

6.1.7.4 Tipos de programas

Vamos a ver a continuación procedimientos, principalmente heurísticos, para obtener soluciones, generalmente aproximadas, de los problemas $n/m/G/c_{max}$. En primer lugar, es conveniente que discutamos qué características de los programas asociamos con la eficacia de los mismos.

Evidentemente existen una infinidad de programas solución posible de un problema del taller mecánico, puesto que pueden insertarse arbitrariamente tiempos muertos de infinitas formas, con lo que se satisfacen las condiciones básicas (ligaduras potenciales y disyuntivas). Naturalmente muchos de estos programas no son interesantes, y únicamente poseerán buenas características de eficiencia aquéllos que sean razonablemente compactos. Consideremos inicialmente todos aquellos programas que tienen el mismo orden o secuencia de operaciones en cada máquina; dichos programas difieren entre sí únicamente por la cantidad de tiempo muerto entre operaciones; existe uno de los programas que domina a todos los demás que llamaremos programa "semiactivo" (esta dominancia debe entenderse respecto a cualquier medida regular de la eficacia). Pasamos de un programa cualquiera al programa semiactivo correspondiente mediante un *desplazamiento limitado a la izquierda*; esto correspondería a desplazar en el diagrama de Gantt todas las operaciones hacia la izquierda hasta que quedasen bloqueadas, sea por la operación anterior en la misma máquina, sea por la operación anterior de la pieza en una máquina distinta. El desplazamiento global limitado a la izquierda puede obtenerse mediante desplazamientos parciales, operación a operación, también limitados a la izquierda; cualquiera que sea el orden elegido para mover las operaciones, cuando ningún

movimiento más sea posible, habremos alcanzado el mismo programa semiactivo. Por consiguiente, un programa semiactivo está asociado a un conjunto de secuencias de las operaciones en cada máquina y recíprocamente. El número de programas semiactivos es por tanto finito, pero desgraciadamente puede ser muy grande, lo que desaconseja la enumeración directa.

Es difícil dar una buena cota superior del número de programas semiactivos. Habitualmente es mucho menor que *G!*, donde *G* es el número total de operaciones de las n piezas:

$$g(1) + g(2) + ... + g(i) + ... + g(n) = G$$

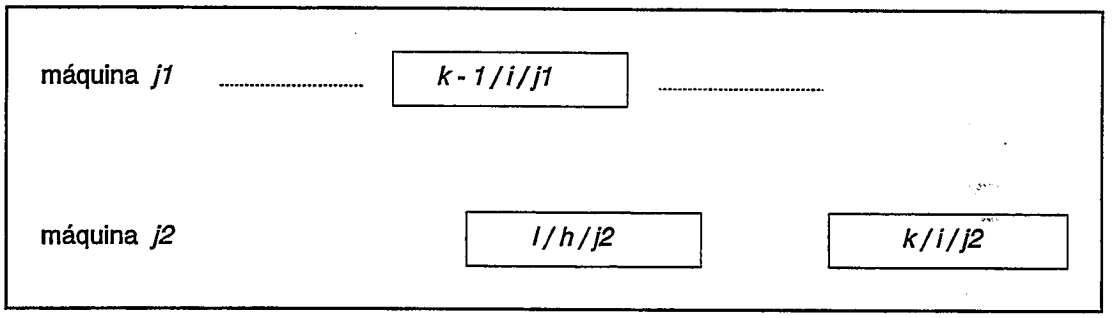


Fig. 6.1.7.9 (a) Programa inadmisible. La operación $(k/i/j_2)$ puede sufrir un desplazamiento limitado a izquierda

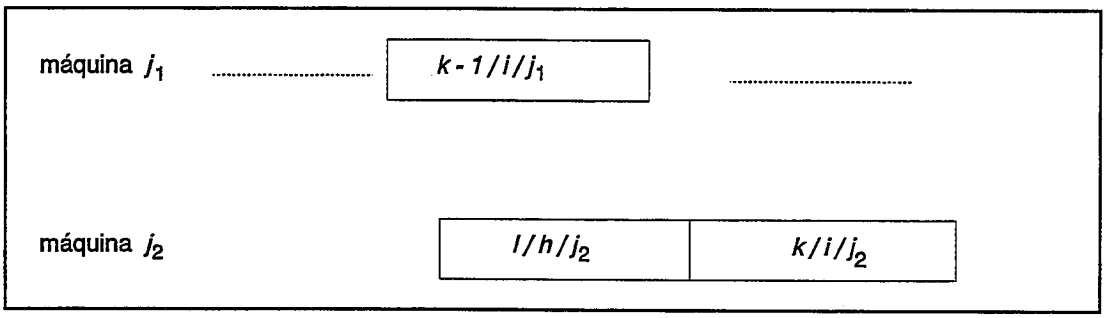


Fig. 6.1.7.9 (b) La operación $(k/i/j_2)$ se ha desplazado a la izquierda, hasta que la operación anterior en la misma máquina le ha cerrado el paso

Si *h(1), h(2), ..., h(j), ..., h(m)* son los números de operaciones a realizar en cada una de las m máquinas, entonces:

$$(h(i)!) \cdot (h(2)!) \cdot ..... \cdot (h(m)!)$$

es una cota superior del número de programas semiactivos. Se alcanza esta cota si cada pieza tiene una sola operación; si no, el número de programas semiactivos es menor. En efecto, supongamos que dos piezas, con dos operaciones cada una, deben elaborarse en las máquina A y B. La primera pieza tiene la primera operación en A y la segunda en B; la segunda pieza exactamente en orden contrario, primero en B luego en A. Por tanto, cada máquina realiza dos operaciones y (2!)·(2!) = 4; sin embargo solo hay tres programas semiactivos:

Programa 1		Programa 2		Programa 3	
Máq. 1 2	Orden 1-2 1-2	Máq. A B	Orden 2-1 2-1	Máq. A B	Orden 1-2 2-1

la cuarta combinación (Maq. A orden 2-1; Máq. B orden 1-2) no es factible. En efecto, para que A pueda ejecutar su operación sobre 2, esta pieza debe haber sufrido su operación en B. Pero en B antes de efectuar la operación en la pieza 2 tiene (según el orden elegido) que efectuarse la de la pieza 1, que antes debe haber sido tratada en A. En lenguaje de grafos diríamos que entre las ligaduras potenciales, y la solución dada a las disyuntivas, hemos creado un circuito de longitud total positiva, por lo que el programa es irrealizable. Con más piezas y operaciones por pieza, las combinaciones de órdenes de las operaciones en las máquinas que son incompatibles aumentarán; por lo que los programas semiactivos (factibles) pueden ser notablemente menos que el valor de la cota.

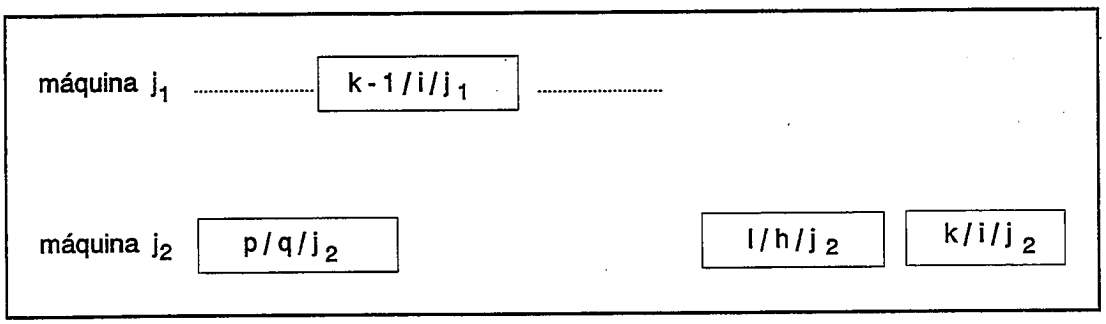


Fig. 6.1.7.10 (a) Programa inadmisible. La operación $(k/i/j_2)$ puede sufrir un desplazamiento a la izquierda

Una categoría más restringida de programas es la de los *programas activos*, en los que no es posible realizar ningún *desplazamiento a la izquierda* (limitado o general) de ninguna operación. Un desplazamiento a la izquierda es la disminución del instante de comienzo de una operación que no requiere aumento del instante de comienzo de ninguna otra; es decir, en este tipo de desplazamiento se admiten saltos por encima de otras operaciones para ocupar huecos (tiempos muertos) de longitud suficiente. Un desplazamiento limitado a la izquierda es un desplazamiento a la izquierda, pero no todos los desplazamientos son

desplazamientos limitados. Por tanto, un programa activo también es semiactivo, pero no necesariamente lo recíproco. Por otra parte, es de esperar que el programa óptimo sea un programa activo según muchas medidas de eficacia, puesto que todo programa semiactivo pero no activo será dominado por uno activo. Podemos limitarnos a considerar en nuestra búsqueda de soluciones los programas activos, pero aunque son menos que los semiactivos, su número puede ser tan grande que desaconseje la enumeración directa. Al contrario que en los programas semiactivos, el orden de los movimientos parciales desde un programa no activo puede influir en cuál sea el programa activo alcanzado.

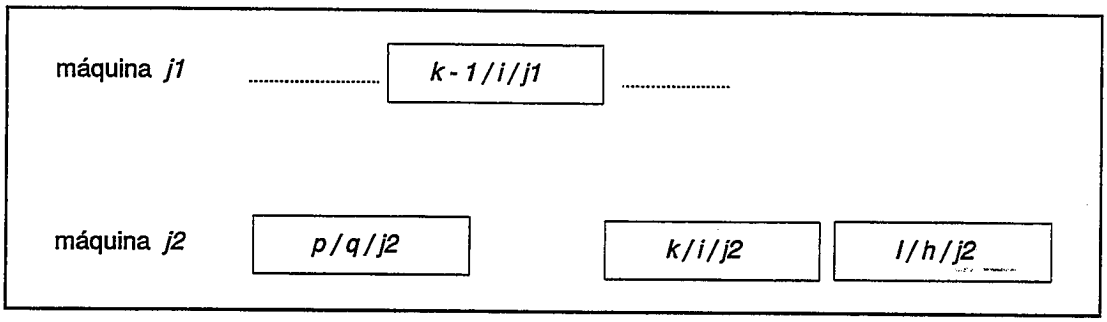


Figura 6.1.7.10 (b): La operación $(k/i/j_2)$ se ha desplazado a la izquierda, saltando por encima de la $(l/h/j_2)$ hasta que la operación anterior de la pieza en la máquina j_1 le ha cerrado el paso

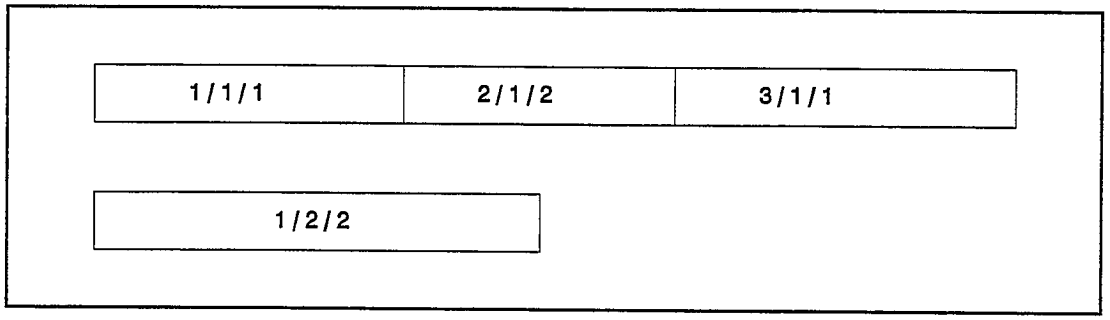


Fig. 6.1.7.11 (a) Datos del ejemplo 22G

Diremos que una operación es *realizable en la máquina j en el instante t* si *t* es mayor o igual que los instantes de terminación de las operaciones de la misma pieza que preceden a la operación considerada. Si para cada instante *t* de tiempo muerto y para cada máquina *j*, no existe ninguna operación realizable en *j* en el instante *t*, el programa correspondiente se llama *programa sin retrasos*. Hablando en forma sencilla, un programa de este tipo es tal que ninguna operación se retrasa cuando la máquina que debe realizarla está libre. Los programas sin retrasos son por definición un subconjunto de los programas activos, pero

no existe ninguna evidencia de que el programa óptimo figure entre ellos. En el ejemplo presentado en la figura 6.1.7.11 (a), los dos programas activos existentes corresponden a las figuras 6.1.7.11 (b) y (c). De ellos (c) es el único programa sin retrasos, puesto que en el (b) la máquina 2 está libre inicialmente cuando podría estar realizando la operación (1/2/2). Sin embargo, respecto muchas medidas de eficacia (b) será considerado mejor que (c).

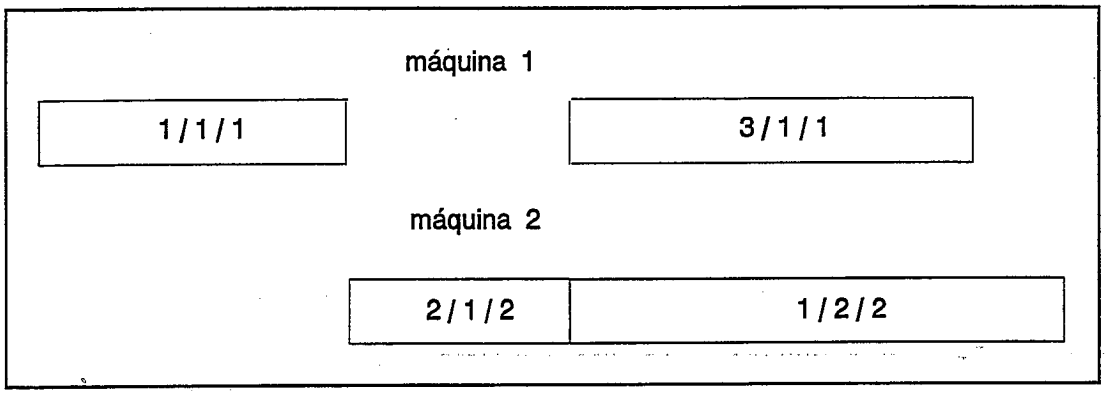


Fig. 6.1.7.11 (b) Programa activo del ejemplo 22G

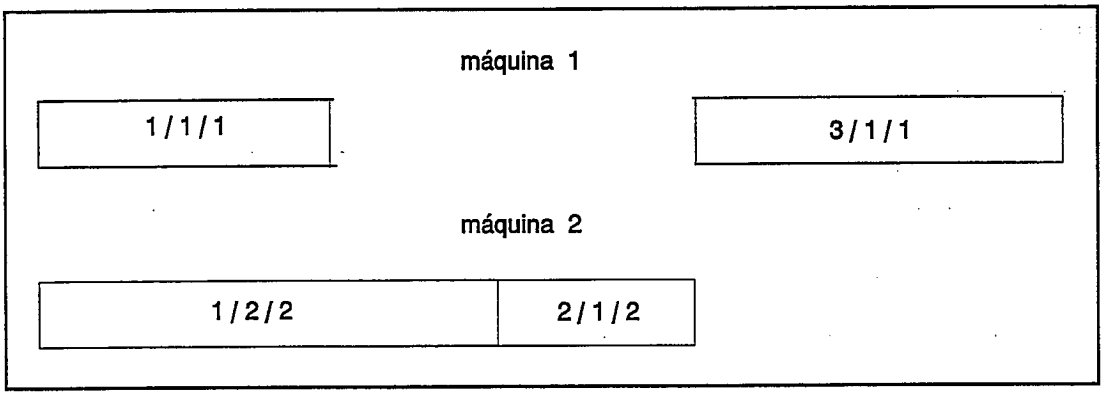


Fig. 6.1.7.11 (c) Programa sin retrasos del ejemplo 22G

No se pueden derivar los programas sin retrasos de los programas activos de la misma forma que éstos se derivan de los programas semiactivos, pero se pueden generar directamente. La clase de programas sin retrasos es muy importante pues existe una evidencia concreta y experimental de que en conjunto son mejores que el resto de los programas activos (aunque entre ellos esté el óptimo). Ello sugiere que si no existe un procedimiento para construir el programa óptimo directamente, y hay que recurrir a la exploración heurística o aleatoria, es ventajoso concentrarse en los programas sin retrasos en vez de en los activos, aunque perdamos la oportunidad de alcanzar el óptimo.

6.1.7.5 Generación de programas

En este apartado vamos a estudiar los principios fundamentales de la generación de programas. Todos los procedimientos actúan sobre el conjunto de las $G = \Sigma\ g(i)$ operaciones, seleccionando cada vez una operación y asignándole una fecha de comienzo. La forma en que se seleccionan las operaciones y se asigna la fecha de comienzo caracteriza cada método. En general, en primer lugar se establece una lista de las operaciones elegibles, *E*, de la cual se extrae una operación que se programa (asignándole fechas de comienzo y fin) pasándola a la lista de operaciones programadas *P*.

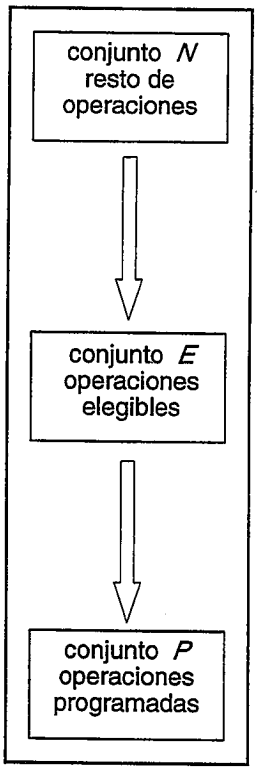


Fig. 6.1.7.12 Subconjuntos o listas de operaciones

La primera distinción a hacer estriba en clasificar los métodos en directos, o en un solo paso, y métodos con reajuste. Los métodos en un solo paso actúan sin vuelta atrás, es decir, una vez se ha asignado una fecha de comienzo a una operación no se modifica para facilitar asignaciones posteriores (una vez una operación está en *P* ni se vuelve a considerar, ni regresa a *E*). En los métodos con reajuste todas las fechas son provisionales hasta llegar al final y pueden sufrir modificaciones. Normalmente los métodos existentes son directos, no porque sean mejores, ni mucho menos, sino a causa del consumo de tiempo y espacio de memoria de los métodos con reajuste.

Evidentemente, sea cual sea el programa óptimo existe una forma de construirlo mediante un método directo; pero generalmente los métodos directos no poseen la misma eficacia en todos los casos.

Todos los métodos combinan una *estrategia* con una *táctica* o *regla*:

- estrategia: determinación de los conflictos a resolver (competencia entre varias operaciones para disponer de un recurso)

- regla: forma de resolución de los conflictos

Una clase de métodos se centran en las piezas u órdenes de fabricación. La estrategia consiste en clasificar las piezas, elegiendo la primera (o más prioritaria) y programando todas sus operaciones; a continuación se elige la segunda y así sucesivamente. La regla se centrará en la forma de colocación de las operaciones de la pieza considerada respecto a las operaciones ya programadas. Las ventajas de esta clase de métodos se centran en el buen control de las piezas importantes (respecto, por ejemplo, a las fechas de vencimiento) y en la facilidad de tener en cuenta los solapes. Entre las desventajas figuran el descontrol de las piezas poco prioritarias, la consideración difícil de las particularidades de centros de carga especiales y la también difícil reacción frente a incidencias.

Otra clase de métodos, que denominaremos de orden cronológico o de *dispatching*, adoptan la estrategia de resolver los conflictos en el orden en que se presentarían en la ejecución real. En un método de *dispatching* las operaciones de cualquier máquina se asignan en orden creciente de sus instantes de comienzo, lo que equivale a decir que, para cualquier máquina se toman las decisiones en el mismo orden en que serán ejecutadas. La clase de los métodos *dispatching* es suficiente, en el sentido de que dado cualquier programa activo se puede generar mediante uno de dichos métodos. Las reglas se refieren, en general, a la elección de una cola, explícita o implícita, de operaciones y de la operación concreta de la cola que va a programarse. En lo que sigue, nos centraremos en esta clase de métodos únicamente.

En el transcurso de la aplicación de un método dispatching *el conjunto de las operaciones programables o elegibles, E*, en un momento dado está constituido por el subconjunto de las G operaciones que tienen sus precedentes en P, pues ya están programadas y se conoce su instante de inicio, y por tanto el de fin. El subconjunto E va modificándose a lo largo de la aplicación del método; en el instante inicial en un problema n/m, habrá en E n operaciones, la inicial de cada pieza, estando las G-n operaciones restantes en N (y P vacío). Cada vez que una operación de una pieza, excepto la última, se programe (pase de E a P) será substituida en E por la siguiente operación de la pieza. Puede compararse útilmente el subconjunto E con el concepto de "lista de espera". Las operaciones que preceden a las de E han sido programadas, las que siguen a las de E no se consideran

todavía. Un procedimiento tipo *dispatching* realiza implícitamente una simulación, con un reloj ficticio para medir el paso del tiempo. En un instante dado t marcado por el reloj ficticio los conjuntos N, E y P se encuentran en cierto estado.

Normalmente el E se subdivide a su vez en subconjuntos. Una posibilidad es la de establecer subconjuntos por piezas, pero el procedimiento más utilizado es la partición de E por máquinas (o colas) en m subconjuntos, E_j, algunos de los cuales pueden ser vacíos en un momento determinado:

$$E = E_1 \, U \, E_2 \, U \, ... \, U \, E_m$$

En E_j existen aquellas operaciones de E que tienen que efectuarse en la máquina j. Consecuentemente, la elección en E de la operación a programar se realiza en dos fases. Primero se divide E en los subconjuntos E_j, y se escoge uno de ellos (no vacío), y luego se elige una operación del mismo. Podemos decir que primero se elige la máquina en la que se va a programar una operación, y luego, si hay varias posibilidades, la operación concreta ("la pieza", ya que sólo puede haber una operación de cada pieza en E).

En un instante de aplicación del algoritmo tendremos las máquinas ocupadas hasta cierto momento en virtud de las operaciones ya programadas, y las operaciones de E disponibles desde cierto instante, en virtud de la terminación de su operación anterior. Vamos a especificar la nomenclatura utilizada (aunque sacrificaremos el rigor a la sencillez de la notación, no haciendo aparecer, por ejemplo, el instante t correspondiente al paso de aplicación del algoritmo). Llamemos:

f_j : instante en que está disponible la máquina j para una nueva operación (inicialmente coincide con los valores de disponibilidad de las máquinas dados en los condicionantes del problema),

$r_{k,i}$: instante en que está disponible la operación (k,i) situada en la cola de la máquina $j = m_{k,i}$, es decir en el subconjunto E_j,

$rp_{k,i}$: instante más temprano en que puede empezar la operación (k,i) en espera en la máquina j,

$$rp_{k,i} = \max \{ r_{k,i} , f_j \mid (k,i) \in E_j \}$$

fp_j : el mínimo de $rp_{k,i}$ para las operaciones de E_j, si éste no es vacío; si E_j es vacío fp_j valdrá infinito,

$$fp_j = \min \{ rp_{k,i} \mid (k,i) \in E_j \} \quad \text{si } E_j \text{ no es vacío}$$
$$fp_j = \infty \quad \text{si } E_j \text{ es vacío}$$

Es el instante más temprano en que puede empezar una operación en la máquina *j*; si E_j es vacío (no hay operaciones en espera en la máquina j) fp_j no está definido y por conveniencia, como se verá más adelante, diremos que es infinito,

ff_j : instante más temprano en que puede quedar libre la máquina *j* si lanzamos una operación de E_j,

$$ff_j = \min \{ rp_{k,i} + p_{k,i} \mid (k,i) \in E_j \} \quad \text{si } E_j \text{ no es vacío}$$

$$ff_j = \infty \quad \text{si } E_j \text{ es vacío}$$

La situación es análoga a la que se produce en ajedrez cuando analizamos varias jugadas. ff_j es el posible nuevo valor de f_j. Si E_j es vacío ff_j no está definido, pero por conveniencia como se verá más adelante, diremos que es infinito.

Puesto que estos valores van modificándose en el transcurso del algoritmo en rigor deberíamos añadir otro índice (por ejemplo, la *t* de la hora del reloj ficticio) que hiciera referencia al paso o fase del mismo, pero en aras a la sencillez antes apuntada hemos prescindido de dicho índice.

Elección de la máquina

A partir de lo anterior se nos presentan dos reglas posibles, ambas coherentes, para la primera elección, la de la máquina:

Regla 1. Se elige la máquina con menor fp_j,

$$fp\text{-min} = \min_j \{ fp_j \}$$

si además la operación elegida es tal que puede empezar en el instante fp-min (regla adicional 1A):

$$(j,k) \in E_j \quad \text{y} \quad rp_{j,k} = fp\text{-}min$$

obtendremos un programa sin retrasos (*dispatching de programas sin retrasos*),

Regla 2. Se elige la máquina con menor ff_j,

$$ff\text{-min} = \min_j \{ ff_j \}$$

si además la operación elegida puede empezar antes de *ff-min* (regla adicional 2A):

$$(j,k) \in E_j \quad \text{y} \quad rp_{j,k} < ff\text{-}min$$

obtendremos un programa activo (*dispatching de programas activos*).

Lo anterior debe entenderse de la siguiente forma: si aplicamos la regla 1, más la 1A, cualquiera que sea la forma en que resolvamos la elección de la operación cuando haya varios candidatos, el programa obtenido es sin retrasos; si aplicamos la regla 2, más la 2A, cualquiera que sea la forma en que resolvamos la elección de la operación cuando haya varios candidatos, el programa obtenido es activo.

Como entre los programas activos debe encontrarse el óptimo, y entre los sin retrasos tal vez no, utilizaremos de preferencia la regla 2, con 2A.

Elección de la operación

Una vez elegida la máquina, la regla adicional puede reducir el número de candidatos a programar. De hecho, debemos considerar el conjunto de candidatos:

- programas sin retrasos:

$$E_j' = \{ (k,i) \in E_j \quad \text{y} \quad rp_{j,k} = fp\text{-}min \}$$

- programas activos:

$$E_j' = \{ (k,i) \in E_j \quad \text{y} \quad rp_{j,k} < ff\text{-}min \}$$

E_j' no puede estar vacío pues, de lo contrario no habríamos podido determinar $fp_j = fp\text{-}min$ o $ff_j = ff\text{-}min$, según el caso. Si en E_j' hay un solo elemento, se programa y no aparece ninguna dificultad adicional.

Si en E_j' hay más de un elemento debemos elegir uno de ellos, insistiendo de nuevo que *cualquiera que elijamos* nos conducirá a un programa sin retrasos o simplemente activo, según el caso. Si no fuera por la explosión de posibilidades que proprociona la combinatoria podríamos realizar la elección aleatoriamente, aplicar el procedimiento repetidas veces, y guardar siempre la solución más eficiente, con menor F_{max} o valor de la medida de eficacia elegida. Cabría esperar que si el número de ensayos fuese suficientemente grande, en alguno de ellos lográramos acceder a una solución óptima. Alternativamente podríamos pensar en enumerarlos todos. Dadas las dificultades prácticas vamos a intentar construir uno, no demasiado malo.

Las reglas propuestas para la elección de la operación suelen basarse en criterios muy sencillos,

- *Regla a (FCFS)* : elegir la operación que ha llegado antes a la máquina ("primero llegado,

primero servido"),

- *Regla b (SPT)* : elegir la operación más corta, es decir, con menor $p_{k,i}$; las operaciones cortas son las que comprometen menos los recursos productivos y tienden a reducir el stock de piezas en curso,

- *Regla c* : elegir la operación que puede terminar antes, es decir, que minimiza $rp_{k,i} + p_{k,i}$, o bien que cumple:

$$rp_{k,i} + p_{k,i} = ff_j$$

se trata de una modificación de la regla anterior,

- *Regla d (TSPT)* : elegir la operación con menor duración excepto en aquellos casos de las operaciones que hayan esperado más de cierto límite, las cuales tienen una prioridad según la regla FCFS.

- *Regla e (LPT)* : elegir la operación más larga.

- *Regla f* : elegir la operación que minimiza el tiempo muerto, es decir, que minimiza la diferencia:

$$rp_{k,i} - f_j$$

o bien la:

$$rp_{k,i} - fp_j$$

lo que contribuirá a la mejor utilización de los medios productivos, interesante si constituyen un "cuello de botella",

- *Regla g* : elegir la operación que tiene mayor duración pendiente $DP_{k,i}$, donde la duración pendiente es la suma de las duraciones de todas las operaciones de la pieza que siguen a la operación considerada:

$$DP_{k,i} = \sum_{l=k+1}^{g(i)} p_{l,i}$$

Esta regla tiene por objeto dar prioridad a aquellas piezas a las que les falta más para terminar,

- *Regla h (EDD)* : elegir la operación que pertenezca a la pieza con fecha comprometida menor. Esta regla intenta dar prioridad a los trabajos que deben entregarse antes,

- *Regla i (LS o DS)* : elegir la operación con menor margen, el margen se define como la diferencia entre los días que quedan hasta la fecha comprometida y la suma de la duración de la operación inmediata más la duración pendiente,

- *Regla j ((DS/RD)* : elegir la pieza con menor cociente margen total por número de operaciones pendientes (incluyendo la inmediata).

- *Regla k (COVERT)* : elegir la operación con mayor coste esperado de retraso dividido por duración.

- *Regla l (CR)* : elegir la operación con menor fracción crítica, cociente entre el tiempo disponible entre la fecha actual y la comprometida y el plazo estimado para terminar la pieza.

Una regla que hemos utilizado regularmente (en ausencia de fechas comprometidas de terminación de los trabajos) consiste en la combinación de las reglas *c* y *g*. Elegimos la pieza con mayor índice o cociente crítico:

$$\frac{DP_{k,i}}{p_{k,i} + (rp_{k,i} - fp_j)}$$

donde $(rp_{k,i} - fp_j)$ representa el tiempo muerto de espera de la máquina causado por la programación de la operación (k,j) en lugar de programar otra operación (existe por lo menos una operación en E_j para la cual esta diferencia es nula).

En caso de empate, utilizaremos como reglas complementarias:

- dar prioridad a la operación con menor tiempo muerto:

$$(rp_{k,i} - fp_j)$$

- dar prioridad a la operación con menor duración $p_{k,i}$.

La regla indicada suele penalizar la última operación de cada pieza, para la que $DP_{k,i}=0$, y por tanto su índice siempre es menor que el de cualquier otra operación no final. En el prototipo informático TALLER2 hemos utilizado, con resultados aceptables, la siguiente variante del cociente crítico:

$$\frac{DP_{k,i} + (g(i) - k) \cdot K_1 + K_2}{p_{k,i} + \alpha (rp_{k,i} - fp_j)}$$

donde:

$g(i)$-k es el número de operaciones de la pieza que siguen a la operación (k,i) considerada,

K_1, K_2 son constantes (hemos obtenido buenos resultados con valores de K_1 y K_2 del orden de la duración media de las operaciones).

$\alpha \geq 1$ es un coeficiente de penalización del tiempo muerto.

La primera expresión dada del índice corresponde a $K_1 = K_2 = 0$ para todas las piezas y $\alpha = 1$; es la que utilizaremos en la resolución de los problemas.

6.1.7.6 Aplicación del procedimiento propuesto

Para montar un determinado subconjunto (tiempo de montaje, 20 horas) se requiere disponer simultáneamente de cuatro piezas que se elaboran en una misma planta. Cada pieza, para su elaboración, debe ser objeto de cierto número de operaciones (entre 3 y 5) cuyas características se detallan en la tabla de la figura 6.1.7.13.

Pieza	1		2		3		4	
Nº operac.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur.
1	A	6	A	4	B	8	A	3
2	B	4	C	10	C	10	B	10
3	C	8	D	6	E	12	D	9
4	D	7	-	-	D	6	C	8
5	-	-	-	-	-	-	E	8

(duración expresada en horas)

Fig. 6.1.7.13 Datos de las operaciones en el ejemplo 4/5/G/Fmin

Deseamos determinar el programa de realización de las operaciones que minimice el plazo desde el momento inicial hasta la iniciación del montaje, suponiendo que las máquinas, de las que hay un solo ejemplar de cada tipo, están disponibles para dicha fabricación a partir de los siguientes instantes indicados en la figura 6.1.7.14, mientras que la materia prima para la elaboración de las piezas ya se encuentra disponible en el almacén en el instante inicial.

Máquina	A	B	C	D	E
Disponible a la hora	6	0	8	14	22

Fig. 6.1.7.14 Instantes de disponibilidad de las máquinas

Este problema pertenece a la categoría 4/6/G/c_{max} (o bien dado que la materia prima de las piezas está disponible en el instante 0, 4/6/G/F_{max}).

Manualmente aplicamos el algoritmo mediante la ayuda de unas tablas, en las que disponemos una parte para cada una de las máquinas. Relativamente a las mismas en forma global indicamos dos valores, la fecha de disponibilidad f_j y el valor ff_j, si generamos programas activos (alternativamente indicamos f_j y fp_j si generamos solamente programas sin retrasos). A continuación disponemos los valores relativos a las operaciones en cada máquina, los cuales ocupan filas. Por columnas, dichos valores son:

i : número o nombre de la pieza,
k : número de operación de la pieza,
p : duración de la operación $p_{k,i}$,
DP : duración pendiente $DP_{k,i}$,
LL : instante de llegada $r_{k,i}$
EM : instante (programado) de comienzo de la operación,
TM : instante (programado) de fin de la operación,
MS : número o nombre de la máquina en que tiene lugar la siguiente operación de la pieza.

En la primera tabla hemos indicado la situación inicial. Existen cuatro operaciones (las primeras de cada pieza) en espera (subconjunto *E*), tres en la primera máquina y una en la segunda; estas máquinas son las únicas para las que se ha definido ff_j; los valores en blanco deben considerarse como infinito. En el transcurso del algoritmo los valores de *f* y *ff* irán variando, pero para facilitar el seguimiento no substituiremos los anteriores, sino que escribiremos los nuevos al lado separados por una raya inclinada (/). Por tanto, el valor válido es el situado más a la derecha salvo, en el caso de *ff* (igual ocurriría con *fp*), si en dicha posición aparece una raya inclinada, o un espacio en blanco el valor debe considerarse infinito (o que en la máquina no hay cola).

No retiraremos tampoco las operaciones programadas (que han pasado ya al subconjunto *P*); las distinguiremos muy fácilmente pues serán aquéllas para las que están definidos los instantes de comienzo y fin.

De acuerdo con nuestras reglas elegiremos en primer lugar la máquina B, que tiene el menor *ff*. Como sólo hay una operación en cola la programaremos para ser realizada entre los instantes 0 y 8. En dicha máquina *f* pasa a valer 8 y *ff* infinito, mientras que en la máquina C aparece la operación (2/3), con duración 10, duración pendiente 18 e instante

de llegada 8. En consecuencia, la máquina C tiene un valor *ff* de 18.

Pasamos ahora a la máquina A, cuya *ff* es la menor. Existen tres candidatos para programar (1/1), (1/2) y (1/4). Estudiemos los cocientes críticos de cada uno:

$$\frac{19}{6} = 3{,}17 \; ; \qquad \frac{16}{4} = 4 \; ; \qquad \frac{35}{3} = 11{,}67$$

y como consecuencia damos prioridad a (1/4) que comenzará en el instante 6 y terminará en el nueve. En consecuencia, en la máquina A, el valor de *f* será 9, y el de *ff* 13. La operación (2/4) aparece en cola en la máquina B, y conduce a que el valor *ff* de la misma sea 19. Nuevamente deberemos programar una operación en la máquina A, y esta vez será la (1/2) la elegida. La fecha de comienzo será 9 y la de terminación 13, con lo que en la máquina A *f* pasa a valer 13 y *ff* 19. La operación (2/2) aparece en la cola de la máquina C, con duración 10, duración pendiente 6 e instante de llegada 13. A continuación corresponde programar una operación en la máquina C, dado que su *ff* vale 18 (contra 19 las de las máquinas A y B, e infinito del resto). Hay dos candidatos (2/2) y (2/3) cuyos cocientes críticos son:

$$\frac{6}{10+(13-8)} = 0{,}4 \; ; \qquad \frac{18}{10+0} = 1{,}8$$

por lo que damos prioridad a (2/3). Una vez programada esta operación con inicio en 8 y final en 18, y actualizados los valores de *f* y *ff* de la máquina C, la operación (3/3) aparece en la máquina *E*, con lo que su *ff* pasa de infinito a 34. A continuación se produce la alternativa de programar en la máquina A o en la B, puesto que ambas tienen el valor *ff* igual a 19; la elección que realicemos *es indiferente* pues conducirá forzosamente al mismo resultado. Por ello comenzamos por la máquina A, y programamos la única operación de la cola, la (1/1) entre 13 y 19. Después de actualizar la operación (2/1) se sitúa en cola de la máquina B, que es la elegida a continuación, pero entre los candidatos no figura *ya que su instante de llegada no es estrictamente inferior a ff*. Proseguimos de esta forma hasta que todas las operaciones han sido programadas (véase la figura 6.1.7.16). Los instantes de terminación de las cuatro piezas son:

pieza 1 : 51
pieza 2 : 34
pieza 3 : 40
pieza 4 : 44

Por tanto, disponemos de las cuatro piezas en el instante (hora) 51, y podemos comenzar entonces el montaje, que terminará en el instante 71.

Máquina A
$f_1 = 6$
$ff_1 = 9$

i	k	p	DP	LL	EM	TM	MS
1	1	6	19	0			B
2	1	4	16	0			C
4	1	3	35	0			B

Máquina B
$f_2 = 0$
$ff_2 = 8$

i	k	p	DP	LL	EM	TM	MS
3	1	8	28	0			C

Máquina C
$f_3 = 8$
$ff_3 =$

i	k	p	DP	LL	EM	TM	MS

Máquina D
$f_4 = 14$
$ff_4 =$

i	k	p	DP	LL	EM	TM	MS

Máquina E
$f_5 = 22$
$ff_5 =$

i	k	p	DP	LL	EM	TM	MS

Fig. 6.1.7.15 Tabla inicial para la aplicación del procedimiento

Máquina A
$f_1 = 6/9/13/19$
$ff_1 = 9/13/19/$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	6	19	0	13	19	B
2	1	4	16	0	9	13	C
4	1	3	35	0	6	9	B

Máquina B
$f_2 = 0/8/19/23$
$ff_2 = 8/19/23$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	4	15	19	19	23	C
3	1	8	28	0	0	8	C
4	2	10	25	9	9	19	D

Máquina C
$f_3 = 8/18/28/36/44$
$ff_3 = 18/28/36/44$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	8	7	23	36	44	D
2	2	10	6	13	18	28	D
3	2	10	18	8	8	18	E
4	4	8	8	28	28	36	E

Máquina D
$f_4 = 14/28/34/40/51$
$ff_4 = 28/34/40/51/$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	7	0	44	44	51	-
2	3	6	0	28	28	34	-
3	4	6	0	34	34	40	-
4	3	9	16	19	19	28	C

Máquina E
$f_5 = 22/34/44$
$ff_5 = 34/44/$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
3	3	12	6	18	22	34	D
4	5	8	-	36	36	44	-

Fig. 6.1.7.16 Tabla final del ejemplo 4/5/G/F_{max}

Aunque el método es heurístico, en este caso la solución es óptima, como puede comprobarse trazando el diagrama de Gantt, en el que se observará la rigidez de la carga de la máquina C sin tiempos muertos intermedios, lo mismo que en la pieza 4, y el conflicto casi permanente entre las piezas 4 y 1.

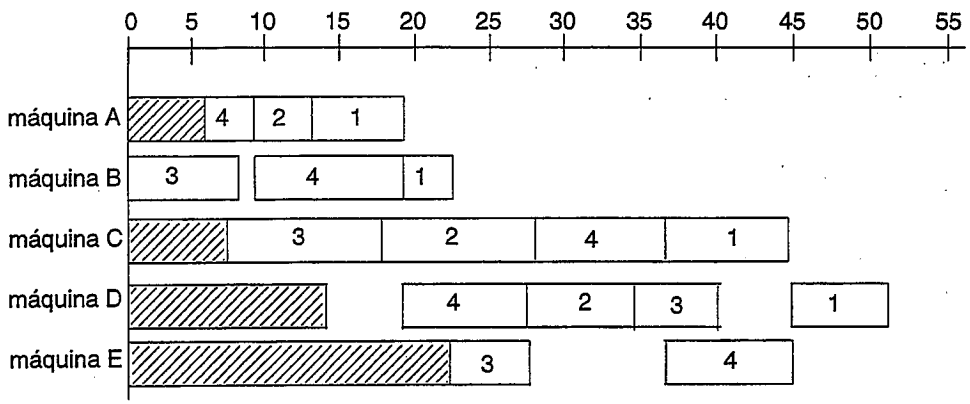


Fig. 6.1.7.17 Diagrama de Gantt del ejemplo 4/5/G/F_{max}

Periódico LV
f_1 = 0/70/105/110/130
fp_1 = 0/90/75/105/110/

i	*k*	*p*	*DP*	*LL*	*EM*	*CM*	*MS*
A	1	70	35	0	0	70	AV
B	2	20	25	90	110	130	MD
C	2	30	20	75	75	105	MD
D	2	5	10	90	105	110	EP

Periódico EP
f_2 = 0/90/110/115/125
fp_2 = 30/90/110/120/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
A	3	20	5	85	90	110	MD
B	2	60	45	30	30	90	LV
C	4	5	0	120	120	125	-
D	3	5	5	110	110	115	AV

Periodico AV
f_3 = 0/75/85/120/155
fp_3 = 30/75/115/150/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
A	2	10	25	70	75	85	EP
B	4	5	0	150	150	155	-
C	1	45	50	30	30	75	LV
D	4	5	0	115	115	120	-

Periódico MD
f_4 = 0/90/120/125/150
fp_4 = 60/105/120/130/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
A	4	5	0	110	120	125	-
B	3	20	5	130	130	150	AV
C	3	15	5	105	105	120	EP
D	1	30	15	60	60	90	LV

Fig. 6.1.7.18 Tabla final del "Los 4 amigos" 4/4/G/F_{max}

Generación de programas sin retrasos

En la figura 6.1.7.18 se indica el resultado de los cálculos para la obtención de un programa sin retrasos en el problema "Los 4 amigos". Se ha conseguido un valor F_{max} = 155, mejor que el del programa semiactivo (de hecho es un programa activo) de la figura 6.1.3.3.

Comparación

Hemos determinado un programa sin retrasos y un programa activo utilizando los procedimientos descritos para el problema *44G_95* cuyos datos se indican en la figura 6.1.7.19

	Pieza 1		Pieza 2		Pieza 3		Pieza 4	
Op.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur.	Máq.	Dur.
1	A	4	B	5	A	5	B	6
2	B C	5	A	3	B	2	C	4
3	C D	3	D	5	C	4	D	3
4	D B	5	C	3 2	D	2	A	2

todas las piezas y las máquinas disponibles en el instante 0

Fig. 6.1.7.19 Datos del problema 44G_95

Máquina A
f_1 = 0/4/9/12/22
fp_1 = 0/4/9/20

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	4	13	0	0	4	C
2	2	3	7	5	9	12	D
3	1	5	8	0	4	9	B
4	4	2	0	20	20	22	-

Máquina B
f_2 = 0/5/11/13/18
fp_2 = 0/5/11/13/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	5	0	12	13	18	-
2	1	5	10	0	0	5	A
3	2	2	6	9	11	13	C
4	1	6	9	0	5	11	C

Máquina C
f_3 = 0/9/15/19/21
fp_3 = 4/11/15/19/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	5	8	4	4	9	D
2	4	2	0	17	19	21	-
3	3	4	2	13	15	19	D
4	2	4	5	11	11	15	D

Máquina D
f_4 = 0/12/17/20/22
fp_4 = 9/12/17/20/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	3	5	9	9	12	B
2	3	5	3	12	12	17	C
3	4	2	0	19	20	22	-
4	3	3	2	15	17	20	A

Fig. 6.1.7.20 Tabla final del problema 44G_95 (programa sin retrasos)

Ambos programas conducen al mismo valor de F_{max} (ver figuras 6.1.7.20 y 6.1.7.21):

Programa sin retrasos		Programa activo	
$c_1 = 18$ $c_2 = 21$ $c_3 = 22$ $c_4 = 22$	$F_{max} = 22$ $F_{med} = 20{,}75$	$c_1 = 20$ $c_2 = 21$ $c_3 = 22$ $c_4 = 22$	$F_{max} = 22$ $F_{med} = 21{,}25$

Ambos resultados parecen equivalentes, aunque ninguno de ellos es óptimo. Permutando en el programa activo las piezas 1 y 2 en la máquina D obtendremos otro programa activo con los valores:

Programa óptimo	
$c_1 = 21$ $c_2 = 21$ $c_3 = 21$ $c_4 = 21$	$F_{max} = 21$ $F_{med} = 21$

Máquina A
f_1 = 0/4/8/13/22
ff_1 = 4/9/8/13/22/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	4	13	0	0	4	C
2	2	3	7	5	5	8	D
3	1	5	8	0	8	13	B
4	4	2	0	20	20	22	-

Máquina B
f_2 = 0/5/11/15/20
ff_2 = 5/11/17/15/20/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	5	0	12	15	20	-
2	1	5	10	0	0	5	A
3	2	2	6	13	13	15	C
4	1	6	9	0	5	11	C

Máquina C
f_3 = 0/9/15/19/21
ff_3 = 9/15/19/21/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	5	8	4	4	9	D
2	4	2	0	17	19	21	-
3	3	4	2	15	15	19	D
4	2	4	5	11	11	15	D

Máquina D
f_4 = 0/12/17/20/22
ff_4 = 13/12/17/20/22/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	3	5	9	9	12	B
2	3	5	3	8	12	17	C
3	4	2	0	19	20	22	-
4	3	3	2	15	17	20	A

Fig. 6.1.7.21 Tabla final del problema 44G_95 (programa activo)

Según las circunstancias puede ser mejor la solución obtenida buscando un programa activo que buscando simplemente uno sin retrasos, o bien al contrario; el resultado depende de la eficacia de la regla utilizada para elegir la operación. Obsérvese que en el

problema 44G_95 el programa óptimo no es un programa sin retrasos; por tanto, cualquiera que sea la regla de elección de la operación nunca podremos encontrar en este caso el programa óptimo, generando programas sin retrasos.

6.1.7.7 Utilización de técnicas interactivas

Como ya hemos visto, no todas las aplicaciones del procedimiento descrito son tan afortunadas como en el caso *45G*. En algunas ocasiones una sencilla permutación de operaciones puede mejorar la solución. Por ejemplo, en el programa sin retrasos del problema "Los 4 amigos" (figura 6.1.7.18) permutando en LV los lectores D y B obtendremos un programa óptimo con F_{max} = 150.

Vamos a analizar la mejora "a posteriori" de un programa. Para ello consideremos el siguiente ejercicio *4/4/G/*F_{max} con todas las máquinas y piezas disponibles en el instante 0, cuyos datos son los de la figura 6.1.7.22.

	Pieza 1		Pieza 2		Pieza 3		Pieza 4	
op.	máq.	dur.	máq.	dur.	máq.	dur.	máq.	dur.
1	A	20	A	26	A	36	B	21
2	B	30	C	15	B	19	A	18
3	C	35	B	30	D	40	D	16
4	D	15	D	29	C	5	C	45

Fig. 6.1.7.22 Datos del problema 4/4/G/F_{max}

La figura 6.1.7.23 muestra el detalle de los cálculos para la determinación de un programa activo, siendo los resultados:

c_1 = 175 c_2 = 204 c_3 = 165 c_4 = 160	F_{max} = 204 F_{max} = 176

No parece que el valor F_{max} = 204 sea el mejor obtenible (buscando un programa sin retrasos habríamos llegado a un mejor resultado), dadas algunas de las elecciones a que nos ha conducido la aplicación del criterio de la fracción crítica; mediante un procedimiento interactivo podemos transformar la solución hallada en otra mejor.

Máquina A
f_1 = 0/20/39/65/101
ff_1 = 20/46/39/65/101/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	20	80	0	0	20	B
2	1	26	74	0	39	65	C
3	1	36	64	0	65	101	B
4	2	18	61	21	21	39	D

Máquina B
f_2 = 0/21/51/120/150
ff_2 = 21/51/110/150/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	30	50	20	21	51	C
2	3	30	29	80	120	150	D
3	2	19	45	101	101	120	D
4	1	21	79	0	0	21	A

Máquina C
f_3 = 0/80/115/160/165
ff_3 = 86/80/115/160/165/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	35	15	51	80	115	D
2	2	15	59	65	65	80	B
3	4	5	0	160	160	165	-
4	4	45	0	55	115	160	-

Máquina D
f_4 = 0/55/160/175/204
ff_4 = 55/160/130/175/204/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	15	0	115	160	175	-
2	4	29	0	150	175	204	-
3	3	40	5	120	120	160	C
4	3	16	45	39	39	55	C

Fig. 6.1.7.23 Tabla final del problema 4/4/G/F_{max}

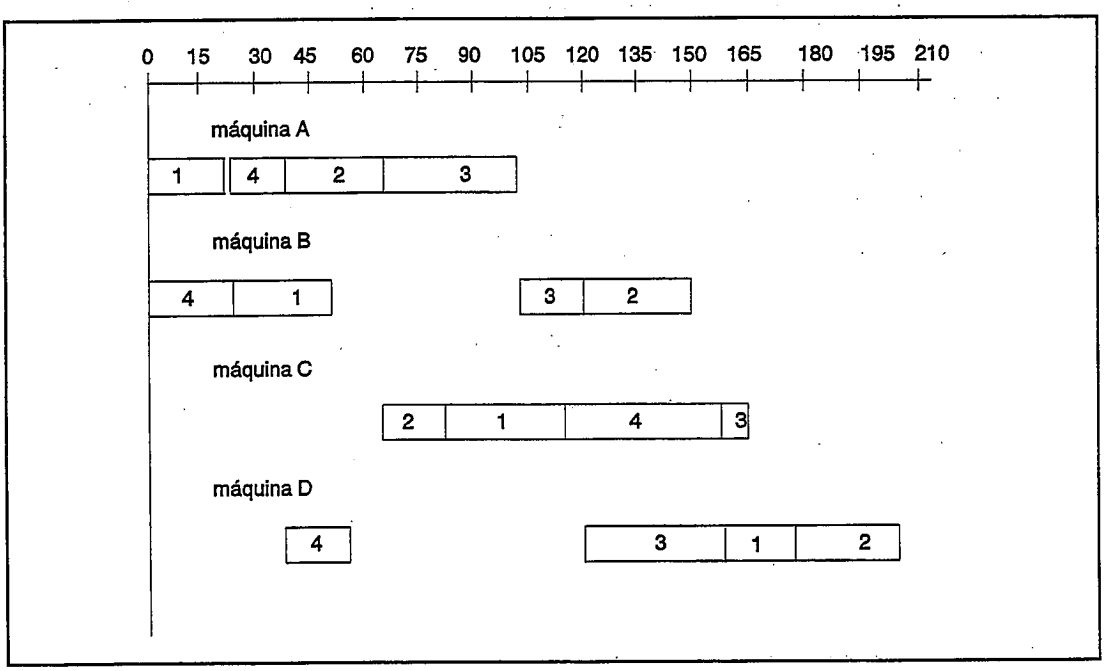


Fig. 6.1.7.24 Diagrama de Gantt inicial (F_{max} = 204)

En primer lugar dibujamos el diagrama de Gantt (que es el de la figura 6.1.7.24). En él observamos que para reducir la ocupación de la máquina D las piezas 3, 1 y 2 deben poder iniciarse antes. Una primera posibilidad consiste en permutar 3 y 1 en D con lo que

ganaremos 5 horas (a costa de retrasar el final de 3 en C 10 horas). Si conjugamos la permutación anterior con la de las piezas 2 y 1 en la máquina C, teniendo en cuenta que la operación siguiente de 2 en B no queda afectada, obtendremos una nueva solución que evaluamos en la tabla de la figura 6.1.7.25.

Dicha tabla es análoga a las anteriores, pero hemos colocado las operaciones en cada máquina en el orden de realización; ya que el orden está prefijado aquí, no cabe cálculo del índice o cociente crítico para la determinación de qué operación va a realizarse (incluso puede prescindirse del cálculo de *ff* para elegir el orden de las máquinas *si estamos seguros de que no ha habido error y que el orden de las operaciones conduce a un programa activo)*.

Máquina A
f_1 = 0/20/39/65/101
ff_1 = 20/46/39/65/101/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	20	80	0	0	20	B
4	2	18	61	21	21	39	D
2	1	26	74	0	39	65	C
3	1	36	64	0	65	101	B

Máquina B
f_2 = 0/21/51/120/150
ff_2 = 21/51/110/150/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
4	1	21	79	0	0	21	A
1	2	30	50	20	21	51	C
3	2	19	45	101	101	120	D
2	3	30	29	101	120	150	D

Máquina C
f_3 = 0/86/101/146/165
ff_3 = 86/80/101/146/165/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	35	15	51	51	86	D
2	2	15	59	65	86	101	B
3	4	45	0	55	101	146	-
4	4	5	0	160	160	165	-

Máquina D
f_4 = 0/55/101/160/189
ff_4 = 55/101/160/189/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
4	3	16	45	39	39	55	C
1	4	15	0	86	86	101	-
3	3	40	5	120	120	160	C
2	4	29	0	150	160	189	-

Fig. 6.1.7.25 Tabla del problema 4/4/G/F_{max} en el orden prefijado

La solución obtenida es ahora:

c_1 = 101 c_2 = 189 c_3 = 165 c_4 = 146	F_{max} = 189 F_{max} = 150,25

El valor F_{max} = 189 representa una mejora de 15 horas (es decir, del 7,94 %), y el F_{med} = 150,25 de 15,75 (10,48 %). El diagrama de Gantt correspondiente es el de la figura 6.1.7.26, en el que toda posible reducción pasa por acelerar las piezas 3 y 2, lo que puede lograrse permutando 2 y 3 en la máquina A. Los cálculos se hallan en la tabla de la figura 6.1.7.27 y conducen a:

$c_1 = 101$ $c_2 = 175$ $c_3 = 166$ $c_4 = 161$	$F_{max} = 175$ $F_{max} = 150{,}75$

A una ganancia de 14 horas en F_{max} (8 %) corresponde un incremento de 0,5 horas en F_{med}. El diagrama de Gantt es el de la figura 6.1.7.28; en él sólo observamos una pequeña posibilidad de mejora, si todo encaja, como es el caso, permutando en A las piezas 4 y 3. La tabla de los cálculos es la de la figura 6.1.7.29; el diagrama de Gantt está en la figura 6.1.7.30 y los resultados son:

$c_1 = 105$ $c_2 = 174$ $c_3 = 165$ $c_4 = 160$	$F_{max} = 174$ $F_{max} = 151$

No parece posible una nueva disminución de F_{max}.

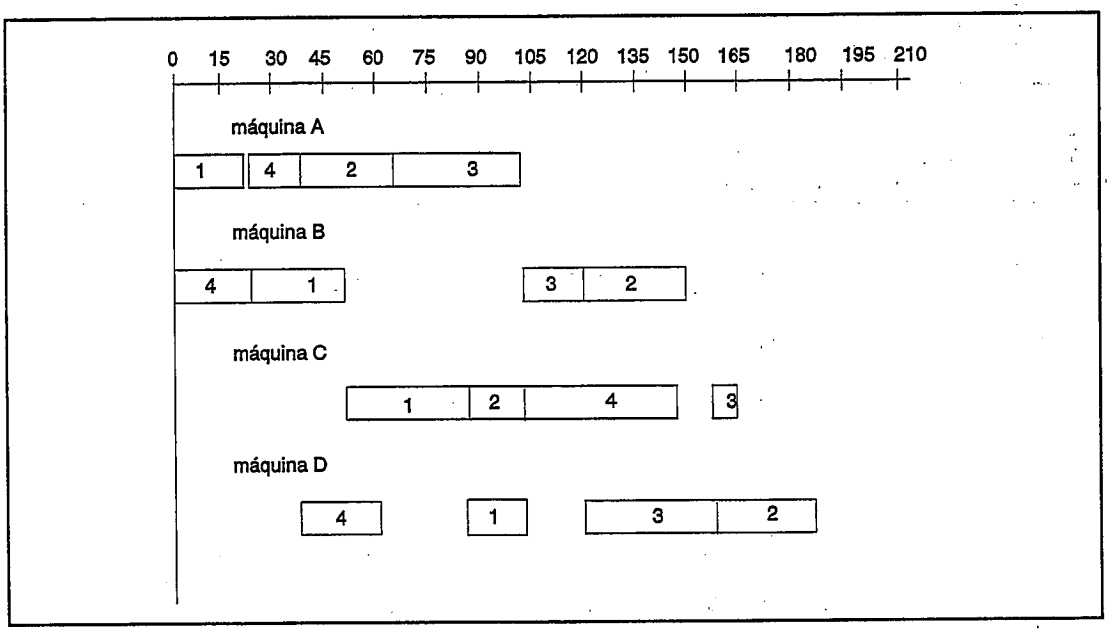


Fig. 6.1.7.26 Diagrama de Gantt después de la permutación de 1 y 2 en C más la de 1 y 3 en D ($F_{max} = 189$)

Máquina A
f_1 = 0/20/39/75/101
ff_1 =

i	k	p	DP	LL	EM	TM	MS
1	1	20	80	0	0	20	B
4	2	18	61	21	21	39	D
3	1	36	64	0	39	75	B
2	1	26	74	0	75	101	C

Máquina B
f_2 = 0/21/51/94/146
ff_2 =

i	k	p	DP	LL	EM	TM	MS
4	1	21	79	0	0	21	A
1	2	30	50	20	21	51	C
3	2	19	45	75	75	94	D
2	3	30	29	116	116	146	D

Máquina C
f_3 = 0/86/116/161/166
ff_3 =

i	k	p	DP	LL	EM	TM	MS
1	3	35	15	51	51	86	D
2	2	15	59	101	101	116	B
4	4	45	0	55	116	161	-
3	4	5	0	141	161	166	-

Máquina D
f_4 = 0/55/101/141/175
ff_4 =

i	k	p	DP	LL	EM	TM	MS
4	3	16	45	39	39	55	C
1	4	15	0	86	86	101	-
3	3	40	5	94	101	141	C
2	4	29	0	146	146	175	-

Fig. 6.1.7.27 Tabla del problema 4/4/G/F_{max} en el orden prefijado

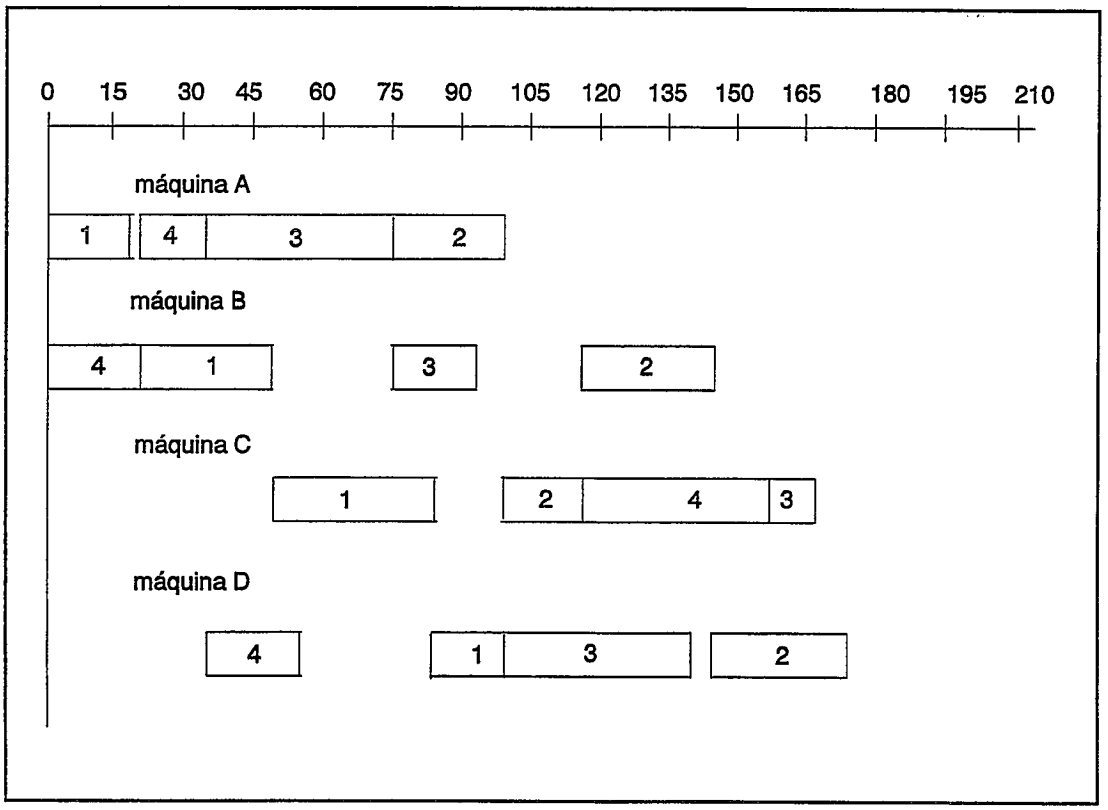


Fig. 6.1.7.28 Diagrama de Gantt después de la permutación de 2 y 3 en A (F_{max} = 175)

Máquina A
f_1 = 0/20/56/74/100
ff_1 =

i	k	p	DP	LL	EM	TM	MS
1	1	20	80	0	0	20	B
3	1	36	64	0	20	56	B
4	2	18	61	21	56	74	D
2	1	26	74	0	74	100	C

Máquina B
f_2 = 0/21/51/75/145
ff_2 =

i	k	p	DP	LL	EM	TM	MS
4	1	21	79	0	0	21	A
1	2	30	50	20	21	51	C
3	2	19	45	56	56	75	D
2	3	30	29	115	115	145	D

Máquina C
f_3 = 0/86/115/160/165
ff_3 =

i	k	p	DP	LL	EM	TM	MS
1	3	35	15	51	51	86	D
2	2	15	59	100	100	115	B
4	4	45	0	90	115	160	-
3	4	5	0	145	160	165	-

Máquina D
f_4 = 0/90/105/145/174
ff_4 =

i	k	p	DP	LL	EM	TM	MS
4	3	16	45	74	74	90	C
1	4	15	0	86	90	105	-
3	3	40	5	75	105	145	C
2	4	29	0	145	145	174	-

Fig. 6.1.7.29 Tabla del problema 4/4/G/F_{max} en el orden prefijado

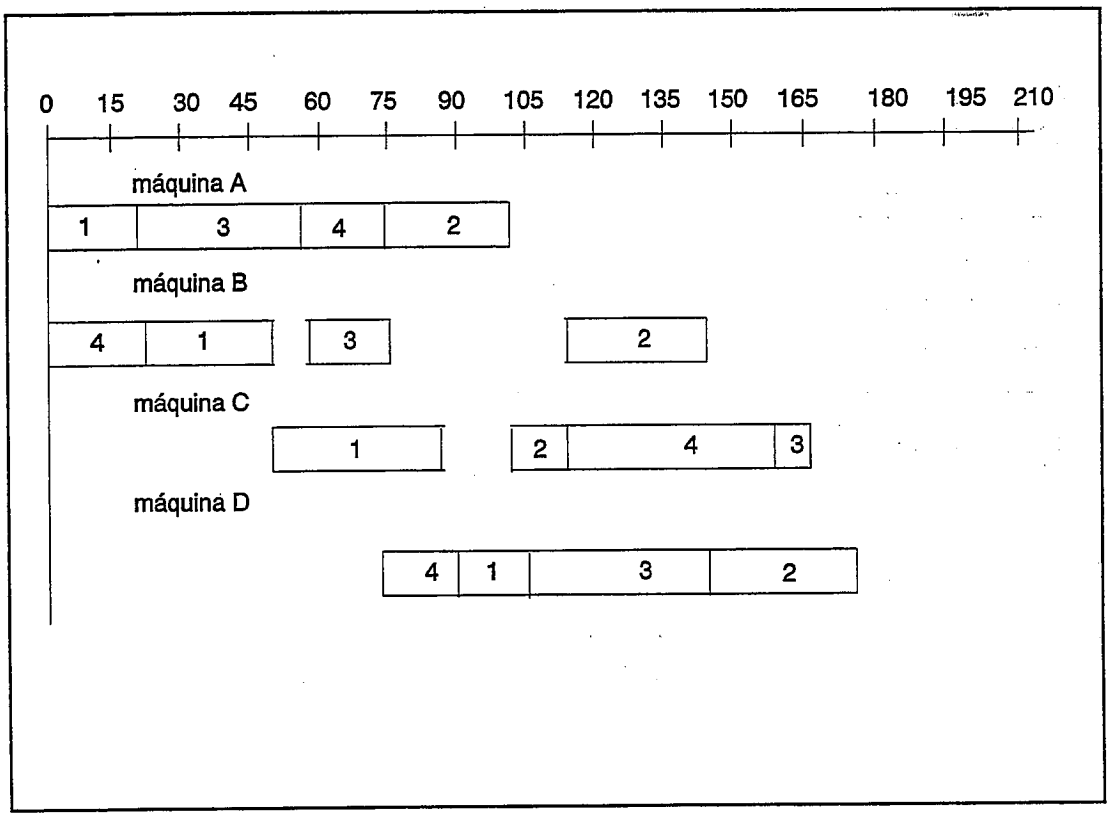


Fig. 6.1.7.30 Diagrama de Gantt después de la permutación de 3 y 4 en A (F_{max} = 174)

6.1.7.4 Método *greedy* a través de la obtención de cotas

Una de las carencias del procedimiento analizado anteriormente es la no disponibilidad de cotas de F_{max} a lo largo del proceso. La obtención de cotas puede lograrse mediante un esquema sencillo de la siguiente forma. En la tabla de la figura 6.1.7.31 hemos repetido la tabla inicial de cálculo del problema $4/4/G/F_{max}$, pero hemos añadido cuatro filas más a la información correspondiente a cada máquina (una para cada operación que debe tratarse). Dichas filas están divididas en seis columnas, en las que las tituladas *i, k, p* y *DP* tienen el mismo significado que antes.

Máquina A
$f_1 = 0$
$ff_1 = 20$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	20	80	0			B
2	1	26	74	0			C
3	1	36	64	0			B
4	2	18	61				D

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	1	0	20	80	100
2	1	0	26	74	120
3	1	0	36	64	146
4	2	21	18	61	161

Máquina B
$f_2 = 0$
$ff_2 = 21$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	30	50				C
2	3	30	29				D
3	2	19	45				D
4	1	21	79	0			A

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	2	20	30	50	101
2	3	41	30	29	129
3	2	26	19	45	115
3	1	0	21	79	100

Máquina C
$f_3 = 0$
$ff_3 =$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	35	15				D
2	2	15	59				B
3	4	5	0				-
4	4	45	0				-

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	3	50	35	15	100
2	2	26	15	59	100
3	4	95	5	0	135
4	4	55	45	0	130

Máquina D
$f_4 = 0$
$ff_4 =$

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	15	0				-
2	4	29	0				-
3	3	40	5				C
4	3	16	45				C

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	4	85	15	0	139
2	4	71	29	0	124
3	3	55	40	5	100
4	3	39	16	45	100

Fig. 6.1.7.31 Tabla inicial (con cotas) del problema $4/4/G/F_{max}$

Los dos títulos nuevos corresponden a:

LLE : llegada esperada; de hecho se trata de una estimación de $rp_{k,i}$ efectuada de la forma siguiente: se toma el valor $rp_{k',i}$, con $k' \leq k$, de la mayor k' de la pieza i cuya rp sea

resultante de una programación y se le suman las duraciones de las operaciones siguientes hasta la *k* exclusive (operaciones *k', k'+1, ..., k-1*):

$$re_{k,i} = rp_{k',i} + \sum_{l=k'}^{k-1} p_{l,i}$$

Si este valor es inferior al valor f_j de la máquina que debe realizar la operación, $j = m_{k,i}$, se toma f_j, en caso contrario $re_{k,i}$:

$$LLE(k,i) = \max \{ f_j , re_{k,i} \}$$

FEF : fecha esperada de fin; para cada máquina consideramos el problema 1-máquina en el que *LLE* marca el instante de disponibilidad de cada pieza, *p* la duración de la operación y *DP* la duración latente. Obtenemos la secuencia que minimiza max $\{ t + p + DP \}$, siendo *t* el instante de inicio de cada operación, en este problema, y *FEF* indica el valor de $(t + p + DP)$. El mayor *FEF* es una cota de F_{max} del problema original. La secuencia óptima puede obtenerse mediante el algoritmo de Carlier, aunque en el caso tratado, con un máximo de cuatro operaciones por máquina, bastará una simple enumeración.

La aplicación de lo anterior a nuestro problema conduce a una cota de 161. Como veremos, esta evaluación es algo optimista, ya que sólo tiene en cuenta los conflictos en *una máquina* y no los posibles encadenamientos de los mismos (por ejemplo, cuando dos piezas tengan aproximadamente la misma ruta) que implican una resolución coordinada de los mismos.

Vamos a utilizar las cotas para crear un programa activo. Suponiendo que inicialmente $ff\text{-}min = ff_A = 20$, empezamos intentando programar una operación en la primera máquina, siendo los candidatos tres (1/1), (1/2) y (1/3). En la tabla de la figura 6.1.7.32 hemos analizado las repercusiones que en la cota representan cada una de las elecciones. Las cotas obtenidas en los dos primeros casos conservan el valor 161, mientras que en el tercero es 171. Por tanto, elegiremos una de las dos primeras, y adoptando, por ejemplo, el criterio del cociente crítico para deshacer el empate, daremos prioridad a la operación (1/1).

Máquina A				Máquina B				Máquina C				Máquina D			
LLE	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*FP*	*FEF*
(1/1) en máquina A															
				20	30	50	101	50	35	15	111	85	15	0	130
20	26	74	120	61	30	29	134	46	15	59	120	91	29	0	159
20	36	64	146	56	19	45	120	115	5	0	146	75	40	5	120
21	18	61	161	0	21	79	100	55	45	0	141	39	16	45	100
(1/2) en máquina A															
26	20	80	126	46	30	50	126	76	35	15	155	111	15	0	155
				41	30	29	154	26	15	59	100	71	29	0	100
26	36	64	146	62	19	45	140	121	5	0	145	81	40	5	145
26	18	61	161	0	21	79	100	60	45	0	105	44	16	45	105
(1/3) en máquina A															
36	20	80	136	56	30	50	130	86	35	15	136	121	15	0	154
36	26	74	156	77	30	29	145	62	15	59	136	107	29	0	139
				36	19	45	100	95	5	0	171	55	40	5	115
36	18	61	161	0	21	79	100	70	45	0	166	54	16	45	115

Fig. 6.1.7.32 Análisis de las posibilidades, ff-min = ff_A = 20

Ahora *ff-min* = ff_B = 21, siendo los candidatos (2/1) y (1/4) aunque el primero, a causa del tiempo muerto a que obligaría, no es muy interesante. El análisis de las cotas figura en la tabla de la figura 6.1.7.33 de la que deducimos la prioridad de (1/4).

Máquina A				Máquina B				Máquina C				Máquina D			
LLE	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*
(2/1) en máquina B															
								50	35	15	111	85	15	0	146
20	26	74	120	61	30	29	149	46	15	59	120	91	29	0	175
20	36	64	146	56	19	45	125	115	5	0	155	75	40	5	120
71	18	61	161	50	21	79	150	105	45	0	150	89	16	45	176
(1/4) en máquina B															
				21	30	50	101	51	35	15	111	86	15	0	130
20	26	74	120	61	30	29	134	46	15	59	120	107	29	0	159
20	36	64	146	56	19	45	120	115	5	0	146	75	40	5	120
21	18	61	161					55	45	0	141	39	16	45	100

Fig. 6.1.7.33 Análisis de las posibilidades, ff-min = ff_B = 21

Ahora *f-min* = ff_A = 39, con tres candidatos: (1/2), (1/3) y (2/4), y se recoge el análisis de las cotas para cada una de las elecciones en 6.1.7.34. Las cotas son en este caso 170, 174 y 170 respectivamente, lo que significa que el valor, mantenido hasta ahora, de 161

era ilusorio. Deshaciendo el empate como antes, mediante el cociente crítico, damos prioridad a (2/4).

Máquina A				Máquina B				Máquina C				Máquina D			
LLE	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*
(1/2) en máquina A															
				21	30	50	101	51	35	15	111	86	15	0	101
				61	30	29	120	46	15	59	120	91	29	0	170
46	36	64	146	72	19	45	155	131	5	0	146	91	40	5	146
46	18	61	161					80	45	0	141	64	16	45	125
(1/3) en máquina A															
				21	30	50	101	51	35	15	101	86	15	0	145
56	26	74	156	97	30	29	156	82	15	59	160	127	29	0	174
				56	19	45	120	115	5	0	151	75	40	5	135
56	18	61	161					90	45	0	146	74	16	45	135
(2/4) en máquina A															
				21	30	50	101	51	35	15	101	86	15	0	101
39	26	74	139	80	30	29	153	65	15	59	160	110	29	0	170
39	36	64	165	75	19	45	139	134	5	0	151	94	40	5	146
								55	45	0	146	39	16	45	100

Fig. 6.1.7.34 Análisis de las posibilidades, ff-min = ff_A = 39

A continuación *ff-min* = ff_B = 51, con un solo candidato (2/1), que programamos en consecuencia; lo mismo que (3/4) en D. La situación consecuente es la de la tabla de la figura 6.1.7.35. El nuevo conflicto se presenta en la máquina A, ya que *ff-min* = ff_A = 65 siendo los candidatos (1/2) y (1/3). En la figura 6.1.7.36 analizamos las cotas, de dar preferencia a una u otra operación, que resultan ser 184 y 175. Consecuentemente programamos (1,3). Seguidamente con *ff-min* = ff_C = 86 dirimimos el conflicto entre (3/1) y (4/4) a favor de la primera. Sin conflictos se presenta la programación de (2/3) en B y de (1/2) en A.

Ahora *ff-min* = ff_D = 101 con dos candidatos, (4/1) y (3/3), dando el análisis de las cotas 175 y 178 respectivamente, por lo que programamos la primera. *ff-min* = ff_C = 116 con candidatos (2/2) y (4/4), cotas 175 y 205, lo que nos lleva a programar la primera. No hay conflicto en programar (3/3) en D ni (3/2) en B; pero si, con *ff-min* = ff_C = 146, entre (4/3) y (4/4), más aparente que real. Las cotas son 191 y 175, por lo que programamos (4/4). La programación de las dos últimas operaciones, (4/3) y (4/2) no presenta conflicto, por lo que se obtiene una solución con F_{max} = 175 (vease la tabla de la figura 6.1.7.37). Puede comprobarse que esta solución es idéntica a la hallada en la tabla de la figura 6.1.7.29 en forma interactiva y que se ha dibujado en la figura 6.1.7.30.

El procedimiento utilizado consiste, por tanto, en elegir la máquina mediante la regla habitual (*ff-min*); restringir los candidatos a aquéllos tales que *rp* < *ff-min*; si hay más de

un candidato analizar la evolución de las cotas al lanzar cada uno de ellos y elegir aquél cuyo lanzamiento conduzca a la mejor cota de F_{max}; en caso de empate aplicar el criterio del cociente crítico (y, si persiste, los subsidiarios). En el caso presente se ha llegado directamente a la solución $F_{max} = 175$.

Máquina A
f_1 = 0/20/39
ff_1 = 20/46/39/65

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	20	80	0	0	20	B
2	1	26	74	0			C
3	1	36	64	0			B
4	2	18	61	21	21	39	D

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	1	programada			
2	1	39	26	74	139
3	1	39	36	64	165
4	2	programada			

Máquina B
f_2 = 0/21/51
ff_2 = 21/51/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	30	50	20	21	51	C
2	3	30	29				D
3	2	19	45				D
4	1	21	79	0	0	21	A

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	2	programada			
2	3	80	30	29	153
3	2	75	19	45	139
4	1	programada			

Máquina C
f_3 = 0
ff_3 = 86

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	35	15	51			D
2	2	15	59				B
3	4	5	0				-
4	4	45	0	55			-

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	3	51	35	15	101
2	2	65	15	59	160
3	4	134	5	0	151
4	4	55	45	0	146

Máquina D
f_4 = 0/55
ff_4 = 55/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	15	0				-
2	4	29	0				-
3	3	40	5				C
4	3	16	45	39	39	55	C

i	*k*	*LLE*	*p*	*DP*	*FEF*
1	4	86	15	0	101
2	4	110	29	0	170
3	3	94	40	5	146
4	3	programada			

Fig. 6.1.7.35 Tabla (con cotas) del problema 4/4/G/F_{max} tras programar 5 operaciones

No es desdeñable tener en cuenta que cada vez que hemos procedido a una elección hemos determinado una cota para *cada una de las alternativas*, lo que nos permite dibujar el árbol de la exploración (figura 6.1.7.38) que es en todo idéntico al utilizado en el método de Lomnicki. Dicho árbol tiene la raíz "todos" con valor de la cota 161, a la que hemos atribuido el numeral 0. Los demás vértices del árbol han sido numerados en el orden de aparición, y dentro del rectángulo correspondiente aparece, además, el valor de la cota. Para reconstruir la secuencia de elecciones en cada ramificación figura la máquina (podría inscribirse también el valor de *ff-min*) y al lado de cada rectángulo la operación cuyo lanzamiento produce la alternativa. Vemos que hemos seguido una trayectoria directa (exploración a lo largo) hasta el vértice terminal que nos ha dado el programa activo, pero

que han quedado inexploradas bastantes ramas con cotas mejores que el valor de dicha solución.

Máquina A				Máquina B				Máquina C				Máquina D			
LLE	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*	*LLE*	*p*	*DP*	*FEF*
(1/2) en máquina A															
65	36	64	165	80 101	30 19	29 45	139 174	51 65 160 55	35 15 5 45	15 59 0 0	101 160 165 146	86 110 120	15 29 40	0 0 5	101 139 184
(1/3) en máquina A															
75	26	74	175	116 75	30 19	29 45	175 139	51 101 134 55	35 15 5 45	15 59 0 0	101 175 166 161	86 146 94	15 29 40	0 0 5	101 175 146

Fig. 6.1.7.36 Análisis de las posibilidades,, ff-min = ff(A) = 65

Máquina A
f_1 = 0/20/39/75/101
ff_1 = 20/46/39/65/101/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	1	20	80	0	0	20	B
2	1	26	74	0	75	101	C
3	1	36	64	0	39	75	B
4	2	18	61	21	21	39	D

Máquina B
f_2 = 0/21/51/94/146
ff_2 = 21/51/94/146/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	2	30	50	20	21	51	C
2	3	30	29	116	116	146	D
3	2	19	45	75	75	94	D
4	1	21	79	0	0	21	A

Máquina C
f_3 = 0/86/116/161/166
ff_3 = 86/131/116/161/146/166/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	3	35	15	51	51	86	D
2	2	15	59	101	101	116	B
3	4	5	0	141	161	166	-
4	4	45	0	55	116	161	-

Máquina D
f_4 = 0/55/101/141/175
ff_4 = 55/101/141/141/175/

i	*k*	*p*	*DP*	*LL*	*EM*	*TM*	*MS*
1	4	15	0	86	86	101	-
2	4	29	0	146	146	175	-
3	3	40	5	94	101	141	C
4	3	16	45	39	39	55	C

Fig. 6.1.7.37 Tabla final del problema 4/4/G/F_{max} en el procedimiento de utilización de las cotas

La mejor cota de un vértice pendiente es 161 (vértice 2) seguida de 170 (vértice 6), 171 (vértice 3) y 174 (vértice 7). Por ello, si deseásemos una solución mejor que la hallada podríamos reemprender la exploración a partir de cualquiera de ellos. Del vértice 7 llegamos a *dos* programas activos con valor F_{max} = 174 (el hallado interactivamente, representado en la figura 6.1.7.30, y el que resulta a partir de este último permutando las operaciones (4/1) y (3/3) en D).

Del vértice 6 llegamos a dos programas activos con $F_{max} = 175$. Continuando a partir de los vértices 2 y 3, la degradación progresiva de las cotas no nos ofrece ningún valor mejor que el hallado. Por tanto, disponemos ya de las soluciones óptimas y sub-óptimas.

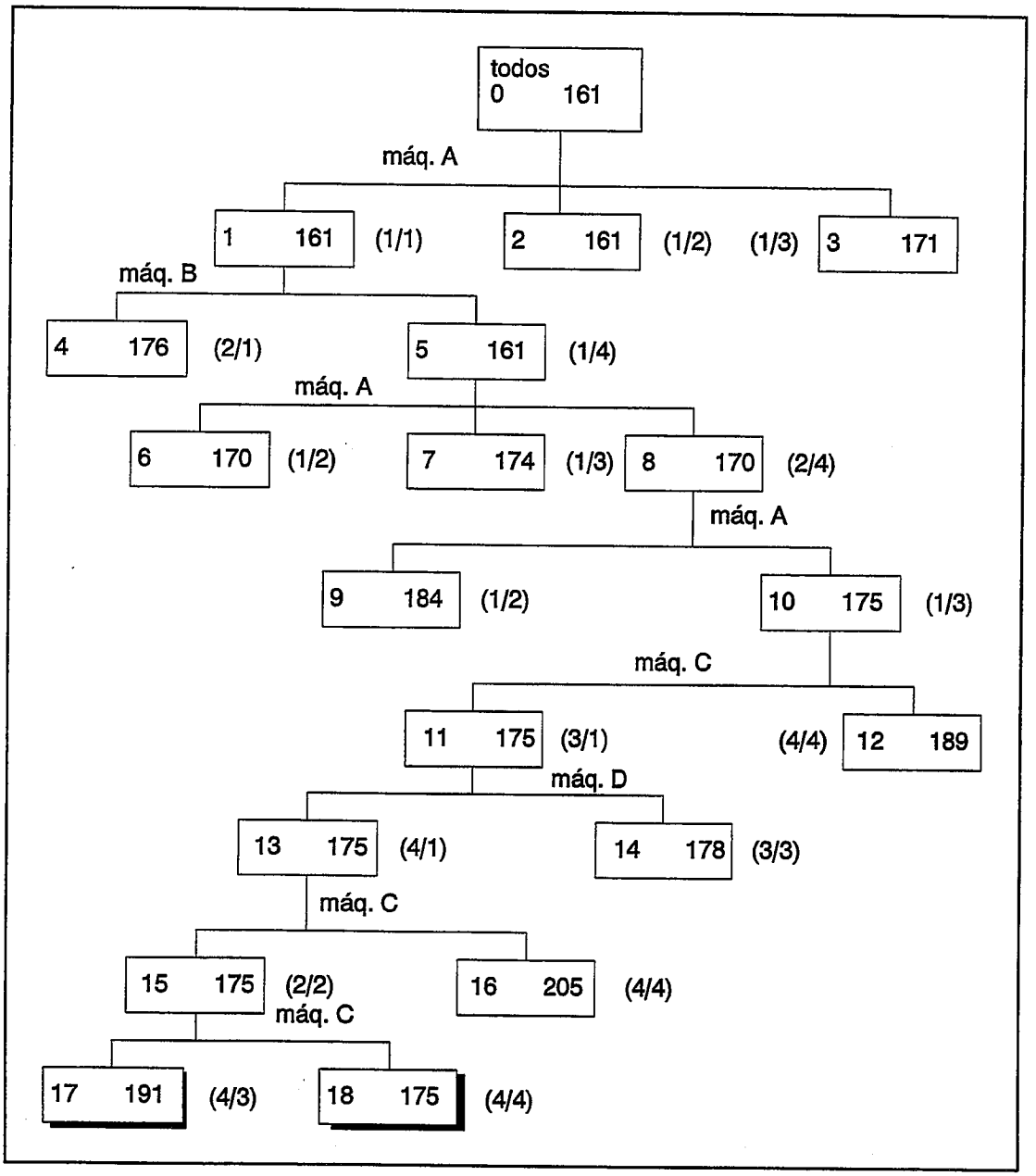


Fig. 6.1.7.38 Árbol de la exploración realizada con las cotas

Este procedimiento de exploración "a lo ancho" es idéntico, en cuanto a su concepción, al aplicado en el método de Lomnicki. Lo sugerimos tímidamente en 1975, con ciertos reparos, ya que era preciso disponer para su aplicación de un procedimiento cómodo para tratar el problema auxiliar 1-máquina, paso resuelto posteriormente por Carlier (1982). Variantes del mismo esquema básico han sido propuestas por el propio Carlier (1989) y por E. Balas (1988, las fechas corresponden a la publicación de artículos en *Management Science*). La solución óptima del famoso problema *10/10/G/*F_{max} (con 100 operaciones, ver los datos en la Tabla de la figura 6.1.7.39) propuesto por Muth y Thompson (1963) fue determinada en 1987 por Carlier tras horas de exploración del árbol en ordenador (su valor es 930); ánalogamente Balas con su variante también llega a dicho valor (aunque aparentemente su algoritmo no le garantiza que se trate del óptimo). El procedimiento del cociente crítico, propuesto anteriormente en el presente texto, obtiene en cuestión de segundos, mediante el programa TALLER (1984) funcionando sobre PC,una solución de valor 1054, fácilmente mejorable en forma interactiva hasta 985 (a menos de un 0,6 % del óptimo). Este valor 1054 es de los mejores que conocemos obtenido por un algoritmo directo, de un solo paso, sin parametrización ni vuelta atrás (*backtracking*) o bien exploración arborescente. Una versión construida por F. Blanco (1989), basada en otros índices con parametrización, ha conducido inicialmente al valor 1018, y posteriormente a una solución de valor 953 (a menos de un 0,25 % del óptimo).

	Operación																			
Piez.	1		2		3		4		5		6		7		8		9		10	
1	A	29	B	78	C	9	D	36	E	49	F	11	G	62	H	56	I	44	J	21
2	A	43	C	90	E	75	J	11	D	69	B	28	G	46	F	46	H	72	I	30
3	B	91	A	85	D	39	C	74	I	90	F	10	H	12	G	89	J	45	E	33
4	B	81	C	95	A	71	E	99	G	9	I	52	H	85	D	98	J	22	F	43
5	C	14	A	6	B	22	F	61	D	26	E	69	I	21	H	49	J	72	G	53
6	C	84	B	2	F	52	D	95	I	48	J	72	A	47	G	65	E	6	H	25
7	B	46	A	37	D	61	C	13	G	32	F	21	J	32	I	89	H	30	E	55
8	C	31	A	86	B	46	F	74	E	32	G	88	I	19	J	48	H	36	D	79
9	A	76	B	69	D	76	F	51	C	85	J	11	G	40	H	89	E	26	I	74
10	B	85	A	13	C	61	G	7	I	64	J	76	F	47	D	52	E	90	H	45

Fig. 6.1.7.39 Datos del problema 10X10

No es nuestra finalidad proceder a una comparación autocomplaciente de heurísticas, sino la de manifestar el interés de los procedimientos mixtos: heurísticas sencillas capaces de dar una buena solución en espacio de segundos combinadas con la intervención interactiva del programador.

Secuencia										
Máq.	1	2	3	4	5	6	7	8	9	10
A	2	5	7	9	8	1	4	10	3	6
B	7	5	4	6	9	10	3	8	2	1
C	5	8	6	2	7	4	9	10	3	1
D	7	5	9	6	2	3	1	4	10	8
E	5	2	4	8	7	1	9	6	10	3
F	5	6	7	9	8	2	1	10	3	4
G	7	5	9	4	10	2	8	6	1	3
H	5	7	9	4	2	3	8	1	6	10
I	5	7	6	4	10	3	8	9	2	1
J	7	2	5	9	6	10	8	3	3	1

Fig. 6.1.7.40 Solución de Blanco al problema 10 x 10 (F_{max} = 953)

6.1.8 Caso *G*, problemas semidinámicos y dinámicos

El procedimiento descrito para tratar los problemas $n/m/G/F_{max}$ estáticos se adapta perfectamente al caso en que las máquinas y/o las piezas no estén todas disponibles en el instante 0.

6.1.8.1 Problemas dinámicos: caso *G*

En un problema de tipo dinámico el funcionamiento del taller no está limitado en el tiempo; en un instante determinado el número de piezas en elaboración será finito, pero las que vayan acabándose y abandonen el taller serán rápidamente substituidas por otras nuevas. El número de piezas o carga de trabajo pendiente en un momento dado en el taller será una cantidad variable que deseablemente oscilará ligeramente alrededor de un valor medio, si la función planificación ha logrado mitigar las grandes diferencias de volumen de la demanda. Si no es así, las oscilaciones de la carga alrededor del valor medio serán importantes y repercutirán en los plazos de realización de los trabajos.

Aun con un número limitado de piezas distintas en catálogo, en la mayoría de los casos no será posible conocer de antemano cuáles van a presentarse para su realización, ni cuándo, salvo dentro de un horizonte reducido. De una forma global sólo podremos

conocer estimaciones de valores medios. El criterio de eficiencia en esta situación no podrá ser c_{max}, que carece de significado, sino una estimación de F_{max}, F_{med}, T_{max} o T_{med}, teniendo los dos primeros, a nuestro entender, el defecto de que son valores dependientes de la duración de las operaciones de cada pieza, además de la de las esperas. Si todas las piezas son muy homogéneas en cuanto a las operaciones, esto carecerá de importancia; en caso contrario, las piezas con operaciones globalmente cortas pueden resultar perjudicadas. Por ello nos inclinamos por un índice basado en los retrasos, lo que obliga, no siendo ello un problema en el mundo industrial, a tener una fecha de vencimiento comprometida para cada pieza.

Dada su complejidad sólo describiremos esquemáticamente las líneas de enfoque de este problema, y nos centraremos en el caso de flujo *G*, siendo fácilmente adaptable lo indicado al caso *F*.

A) Para cada pieza que llega al taller establecemos su fecha comprometida d_i, fijada exteriormente por la planificación o por acuerdos comerciales, o bien calculada interiormente mediante una expresión del tipo:

$$d_i = r_i + p_i + (K_1 + g(i) \cdot K_2)$$

B) Igualmente determinaremos el margen total, MT_i, existente para cada pieza, y su parte alícuota por operación, MPO_i.

$$MT_i = \max \{ d_i - p_i - r_i , 0 \}$$

$$MPO_i = \frac{MT_i}{g(i)}$$

C) Al programar por el método de lanzamiento o *dispatching* se calculará para cada operación elegible para la máquina *j*, es decir, del conjunto E_j', el plazo pendiente estimado y el margen relativo existente:

$$PPE_{k,i} = p_{k,i} + DP_{k,i} + (g(i)\text{-}k\text{-}1) \cdot MPO_i$$

$$MR_{k,i} = d_i - fp_{jp} - PPE_{k,i}$$

D) Se dará prioridad a las operaciones con margen relativo negativo (que se estima están ya retrasadas). Se resolverán los empates con otra regla, por ejemplo el índice señalado anteriormente.

6.1.8.2 Esquema de un sistema informático de programación de la producción

Dada la complejidad del proceso de programación, desarrollado a gran nivel de detalle, es conveniente la utilización de un soporte informático. En la figura 6.1.8.1 se han recogido en grandes líneas el esquema conceptual de un paquete informático de este tipo, que se desarrolla a través de tres módulos principales:

Módulo de actualización de datos. Este módulo tiene por objeto constituir una base de datos del sistema, que contiene todas las informaciones que interesan a éste salvo la situacion del taller. Por tanto, se trata de las informaciones con menor grado de volatilidad relativas a las operaciones tipo (si existen), equipos y sus modalidades de utilización, rutas, tiempos y plazos, tanto de las órdenes tipo como de las efectivas, cartera de órdenes, con fecha de emisión, fecha prevista de realización, prioridades (si se utilizan), datos de costes (id.), etc. La cartera de órdenes se irá alimentando de las nuevas que sean generadas por el sistema de cálculo de necesidades o los pedidos, y reduciendo en aquéllas terminadas que pasen bajo el control de otros sistemas informáticos de gestión

Módulo de establecimiento de la situación inicial. Puesto que el proceso de programación es cíclico, al iniciar un ciclo determinado la situación es tal que existe un programa anterior, unas acciones que se han desarrollado y se está desarrollando en virtud del mismo, y, posiblemente, se han producido incidentes no previstos y unas desviaciones respecto a lo programado. Por ello la definición del punto de partida de la reprogramación es crítica y delicada. Puede realizarse *off-line*, introduciendo el programador las informaciones consiguientes, o bien mediante una conexión directa con el taller.

Módulo de programación. Asociado a una base de datos en la que se guarde el programa vigente y las desviaciones, éste es el módulo que realiza las funciones y utiliza los procedimientos similares a los algoritmos que se han descrito en los últimos capítulos. Distinguimos a fines pragmáticos dos partes en este módulo: la que realiza la manipulación de la información relativa a las operaciones, garantiza la coherencia de las precedencias y prioridades, presenta los datos al programador y asume las modificaciones sugeridas por el mismo, etc. y la que contiene los procedimientos que permiten secuenciar las operaciones. Consideramos que la primera parte debe poder funcionar aunque el juego de reglas de secuenciación a utilizar se modifique, por ejemplo por la adición de nuevas o más complicadas reglas resultado de la experiencia en la utilización del sistema.

En forma complementaria incluimos un módulo monitor de control, que facilite los diálogos con el programador usuario, cuya intervención interactiva en el sistema es muy importante.

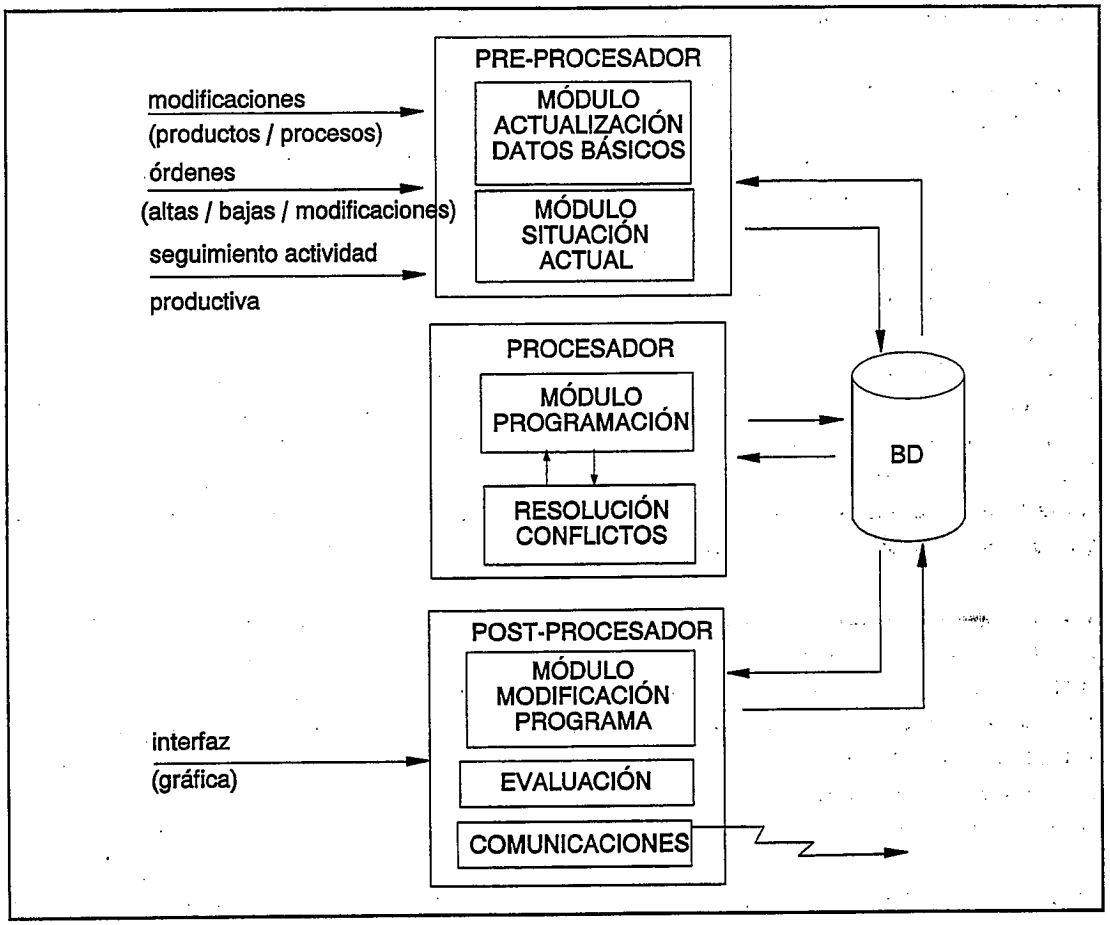
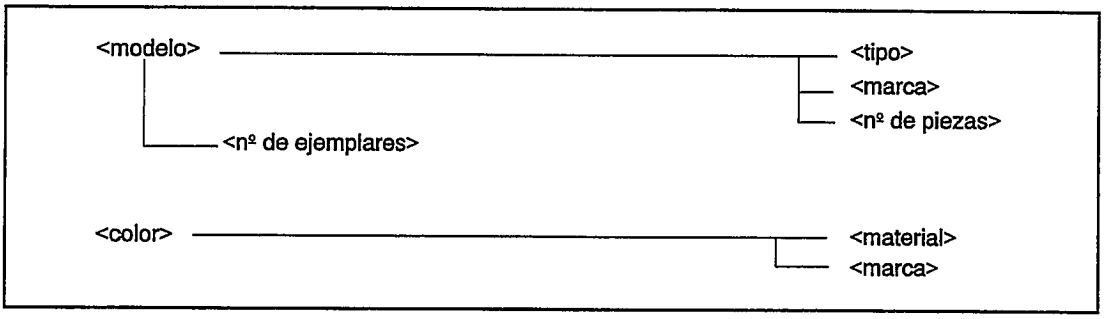


Fig. 6.1.8.1 Arquitectura de un sistema de ayuda a la programación de operaciones

6.1.8.3 El sistema *Berenice*

El sistema *Berenice* fue desarrollado para ayudar a la programación de la fabricación de pequeñas piezas de resinas sintéticas en una cadena automatizada a partir de una cartera de pedidos. Cada pieza se obtiene a partir de dos o tres obleas (según sean piezas bicapa o tricapa) del material sometidas a presión y temperatura en unos moldes adecuados de alta precisión. Cada molde se compone de una base y varios juegos de tapas y émbolos; emplearemos la palabra molde para referirnos al conjunto de todos estos elementos. Dado el alto coste de los moldes el número de ejemplares de cada uno es limitado (de 2 a 6). A causa de la pequeñez de las piezas un molde permite obtener de las obleas tres, cuatro o cinco piezas idénticas (según su tamaño) en el mismo proceso. La configuración del molde define una propiedad del producto obtenido asociada a su forma geométrica que denominaremos "modelo".

El material puede tener diferentes composiciones que se traducen en una propiedad del producto obtenido, que denominaremos "color". Esto nos permite definir el producto como la combinación de dos módulos:



Por imperativos comerciales los modelos y colores están agrupados en marcas, y sólo es factible la combinatoria de modelos y colores dentro de la marca; la marca define así mismo el número de capas del producto, y por tanto el número de obleas distintas necesarias en el proceso.

Las obleas tienen un período de maduración de una semana (5-7 días) y además sufren un proceso de degradación (el material está "vivo") aunque se conserven en condiciones adecuadas de refrigeración. Por consiguiente, es preciso programar adecuadamente su fabricación. Tanto por disponibilidad de dispositivos automáticos de suministro de obleas a la línea, como para facilitar operaciones manuales posteriores, el número de colores de los productos a fabricar un día determinado es limitado (por ejemplo a 4).

En la línea caben simultáneamente 60 moldes en diferentes fases del proceso. El tiempo de recorrido de un molde a lo largo de la línea es de 30 minutos, por lo que a dos turnos de siete horas y media efectivas diarias permite realizar un máximo de 1680 moldeos por día.

Las funciones de *Berenice* son, por consiguiente, dos:

- programación de la fabricación de las obleas con horizonte una semana,

- programación de la fabricación de la línea con horizonte diario.

En cada caso se tendrán en cuenta las restricciones impuestas por las limitaciones del sistema productivo y en el segundo específicamente la disponibilidad de obleas.

El núcleo fundamental del procedimiento empleado (tanto para la programación de obleas como de la línea) consiste en el establecimiento de una secuencia de moldeos; para ello

se establece una lista de los candidatos a ser secuenciados (productos con demanda pendiente cuyo color corresponde a los disponibles), lista que se actualiza a medida que se van secuenciando moldeos y por tanto reduciendo la producción pendiente y las obleas disponibles. A cada candidato se le asigna una nota que resulta de tener en cuenta tres aspectos:

- la antigüedad del pedido, *NA*, función lineal del retraso,

- el volumen del pedido, *NPE*, función convexa del tiempo minimo necesario para realizar todos los artículos (se favorecen los pedidos muy pequeños y los muy grandes),

- el volumen del producto, *NPR*, agrupando el porcentaje que representa el artículo dentro del pedido y el tiempo necesario para realizar el total de unidades del artículo correspondientes a todos los pedidos pendientes,

Estas notas conducen a una nota global parametrizada:

$$NG = \alpha \cdot NA + ß \cdot NPE + \gamma \cdot NPR$$

donde α, ß y γ son unos pesos, modificables por el usuario, tales que:

$$\alpha + ß + \gamma = 1$$

La nota global se mantiene constante a lo largo de la secuenciación de un día. En cada posición de la secuencia se programa el moldeo del artículo candidato posible (existe disponibilidad de moldes) con la mayor nota.

Mediante este procedimiento es posible establecer la secuencia de moldeos de los próximos cinco días, para los cuales se conoce la disponibilidad de obleas. Una adaptación permite determinar la secuencia del sexto día, que nos permite conocer el volumen y la calidad de las obleas necesarias para dicho día, y por tanto realizar la programación con horizonte una semana.

En primer lugar, cabe definir los colores correspondientes al sexto día, en función de la demanda pendiente estimada. Para ello se seleccionan los colores de acuerdo con dos criterios:

- el 50 % aproximadamente en función del volumen de producción pendiente de los mismos,

- el 50 % restante en función de la antigüedad de los mismos (producción pendiente de dicho color perteneciente a un pedido con mayor antigüedad).

En ambos casos se utiliza el criterio alternativo para resolver los empates. A fin de elegir colores susceptibles de conducir a moldeos que saturen la línea se emplea un procedimiento heurístico (inspirado en el problema combinatorio denominado de "los músicos"). Elegidos los colores se utiliza el procedimiento indicado anteriormente sin limitar el número de obleas de cada color; una vez conocida la secuencia se puede determinar el número de obleas necesarias y traducirlo a cantidades a fabricar.

6.1.8.5. El sistema *Palamedes*

El sistema *Palamedes* fue diseñado para una empresa que fabrica piezas metálicas cuya estructura productiva es del tipo taller cerrado (*job shop*), con grupos de máquinas idénticas o similares todas ellas capaces de realizar una determinada operación. Produce un número limitado de piezas para un grupo de clientes reducido y los pedidos de los mismos establecen una cartera de pedidos organizados por tipo de pieza: para cada pieza existe una lista de entregas a realizar formada por parejas cantidad/fecha. Esta cartera de pedidos se transforma en órdenes de fabricación, que la empresa denomina "lanzamientos", consistentes en una cantidad a fabricar con una fecha de vencimiento. Un lanzamiento puede servir para atender más de una de las entregas comprometidas por la empresa. Nos centraremos en el procedimiento de programación de los lanzamientos y no en el correspondiente a la transformación de la cartera de pedidos en éstos; dicho procedimiento posee gran similitud con los descritos anteriormente.

La unidad programable considerada es el lanzamiento (i). Cada lanzamiento corresponde a un tipo de pieza y cantidad, y está asociado a una fecha de emisión r_i y a una fecha de vencimiento (terminación planificada o *due date*, d_i). Cada lanzamiento se descompone en operaciones cuya naturaleza, secuencia y duraciones se obtiene a partir de la información técnica de la base de datos.

El procedimiento básico utilizado simula el comportamiento de los lanzamientos en el taller. Cada etapa del procedimiento consiste en la adopción de una decisión, fijando el instante de comienzo de una operación en una máquina de un grupo de máquinas. Diremos que se *ha programado* la operación cuando se ha fijado su instante de comienzo. Elegimos un procedimiento del tipo *dispatching*, es decir, las decisiones para cada grupo de máquinas se adoptan en el mismo orden en que las operaciones se realizarían de ejecutarse el programa tal como se ha establecido. Por tanto, en una fase dada del procedimiento las operaciones se dividen en tres categorías: operaciones ya programadas, operaciones programables (la operación precedente ha terminado o ha llegado al nivel que permite el solape) y operaciones no programables todavía.

Para elegir la operación programable cuyo instante de comienzo fijamos actuaremos en dos pasos:

Paso 1. Elegiremos el grupo de máquinas en el cual programaremos una operación,

Paso 2. Elegiremos la operación programable más adecuada,

Esto lleva a considerar que las operaciones programables se clasifican en diferentes colas "lógicas", una delante de cada grupo de máquinas. Llamamos Q_j a la lista de operaciones que constituye la cola frente al grupo de máquinas j. Algunas de las colas pueden ser vacías, y en consecuencia el grupo de máquinas correspondiente no podrá ser elegido en el paso 1.

Cuando llegamos al paso 2 disponemos de un origen de tiempos objetivo, instante "en curso" o actual, que coincide, aproximadamente, con el momento más temprano en que una de las máquinas del grupo queda disponible. Antes de elegir el grupo también podemos considerar un origen de tiempos, que será el instante más temprano en que una máquina está disponible en un grupo con *cola* de operaciones programables. Más adelante formalizaremos estos conceptos.

Para cada lanzamiento el tiempo que resta desde este instante en curso hasta d_i es el tiempo disponible (a_i). Por otra parte, en un momento dado, y teniendo en cuenta las operaciones realizadas o en curso, cada lanzamiento tiene un tiempo de elaboración (mínimo) pendiente, considerando los tiempos operatorios y los posibles solapes (TE_i). Con diversos juegos de hipótesis el tiempo pendiente puede adquirir diversos valores, pero en cualquier caso la comparación entre a_i y TE_i nos permite apreciar si el lanzamiento está adelantado, retrasado o dentro de programa.

Suponiendo que no se producen interrupciones, que una operación iniciada en una máquina prosigue en la misma hasta su terminación, dadas unas operaciones programadas podemos estimar que cada máquina quedará disponible cuando acabe la operación en curso (la última programada en ella). Llamaremos a este instante f.

Vamos a formalizar un conjunto de índices, asociados a una fase de aplicación del procedimiento:

- consideraremos diferentes grupos de máquinas j; si el grupo j consta de L máquinas las denominaremos :

$$j_1, j_2, \ldots\ldots, j_L$$

- llamaremos a la cola lógica frente a la máquina o grupo j, Q_j (conjunto o lista)

- a toda operación $k \in Q_j$ está asociado un instante de llegada o disponibilidad, r_k, que para la primera operación de un lanzamiento es la fecha de emisión del mismo, y para el resto de operaciones, salvo solapamientos y transportes, coincidirá con la fecha de terminación

de la operación anterior,

- la disponibilidad de j_i será f_{ji},

- la (primera) disponibilidad del grupo será:

$$f_j = \min_i \{ f_{ji} \}$$

- la fecha más temprana (posible) de comienzo de la operación k depende de la disponibilidad de la máquina y del instante de llegada:

$$rp_k = \max \{ r_k , f_j \} \quad , \quad k \in Q_j$$

- fecha más temprana (posible) de comienzo de un nuevo trabajo en el grupo j

$$fp_j = \min \{ rp_k \mid k \in Q_j \text{ y } Q_j \text{ no es vacío} \}$$

si Q_j es vacío fp_j carece de significado, pero un valor cómodo a asignarle es infinito,

(los valores fp_j son los que podemos utilizar como origen de tiempos)

- fecha más temprana (posible) de liberación de una máquina del grupo j tras una nueva operación:

$$ff_j = \min \{ rp_k + p_k \mid k \in Q_j \text{ y } Q_j \text{ no es vacío} \}$$

siendo p_k la duración en j de la operación k.

(si Q_j es vacío ff_j también carece de significado y es cómodo asignarle el valor infinito)

Para cada *grupo* de máquinas idénticas con cola de operaciones en espera se puede calcular *fp* y *ff*.

Puede utilizarse tanto *fp* como *ff* para la elección de la máquina en la que se va a programar una operación, como ya hemos indicado anteriormente. En ambos casos la máquina elegida es aquella con el menor valor. La utilización de *fp* se justifica si deseamos construir programas compactos sin tiempos muertos, en la medida de lo posible, de las máquinas, y por tanto, una vez elegida la máquina, la operación elegida será una que precisamente pueda comenzar en el instante *fp* (por lo menos debe existir una). Sin embargo nuestro objetivo, en la programación, no es tanto la utilización de las máquinas como el cumplimiento de las fechas d_i. Por tanto en ocasiones puede resultar conveniente mantener una máquina ociosa, aunque exista una operación que potencialmente puede

realizarse en ella, para esperar la llegada de otra operación en un lanzamiento más prioritario. Por dicha causa adoptamos *ff* como índice para la elección de la máquina, eligiendo la operación de forma que no queden huecos de tiempo muerto de máquina en los que pudiera realizarse otra operación programable.

La posibilidad de solapamientos (y de tiempo de transporte de un grupo a otro) no repercute en los cálculos descritos anteriormente sino en la actualización de los valores una vez adoptada una decisión, pudiendo ser la disponibilidad de un lanzamiento para una operación anterior (posterior) a la finalización de la operación precedente del mismo lanzamiento.

La descripción del algoritmo es la siguiente:

Repetir hasta agotar las operaciones (o hasta superar un horizonte dado).

A1 - Elegir el grupo j con menor ff_j y programar en él una operación para la máquina que está disponible antes.

Para ello construir previamente la lista abreviada QE_j de las operaciones que no dejan huecos:

$$QE_j = \{ k \in Q_j \text{ y } r_k < ff_j \}$$

A2 - Si en QE_j hay un solo candidato, programarlo.

Si hay más de uno, calcular:

$$IND = \max \{ TE_i / a_i \mid k \in QE_j \text{ es una operación de } i \}$$

TE_i = tiempo de elaboración pendiente del lanzamiento i

y determinar:

$$QE1_j = \{ k \in QE_j \text{ y } TE/a_i \geq \alpha.IND \}$$

Si en $QE1_j$ hay un solo candidato, programarlo.

Si hay más de uno, calcular:

$$TMM = \min \{ TM_k \mid k \in QE1_j \}$$

$$TM_k = r_k - fp_j \text{ (tiempo muerto)}$$

y determinar

$$QE2_j = \{ k \in QE1_j \text{ y } ß.TM_k \leq TMM \}$$

Si en $QE2_j$ hay un solo candidato, programarlo.

Si hay más de uno, calcular

$$CGM = \min \{ CMS_k \mid k \in QE2_j \}$$

(CMS_k = estado de carga relativo del grupo en que se desarrolla la operación siguiente de la k)

y determinar

$$QE3_j = \{ k \in QE2_j \text{ y } \gamma.CMS_k \leq CGM \}$$

Si en $QE3_j$ hay un solo candidato, programarlo.

Si hay más de un candidato, programar aquél cuya operación en j es la más corta.

Las experiencias realizadas han dado un buen comportamiento de la heurística adoptando los valores:

$$\alpha = 0{,}7 \; ; \; 0{,}8 \leq ß \leq 1 \; ; \; 0{,}8 \leq \gamma \leq 1$$

Cabe realizar las siguientes observaciones:

S1 - Un candidato k induce, si $rp_k - fp_j > 0$ un tiempo muerto TM_k en la máquina (y el grupo) que otro candidato no produce. Aunque aceptamos la posibilidad de este tiempo muerto, de alguna manera debe *penalizarse* a los candidatos con tiempo muerto inducido.

S2 - Dado que en primera instancia se pretende que el algoritmo sea directo, sin *backtracking*, deberá añadirse más adelante, en una nueva versión del sistema, una exploración del "estado de carga" (relativo), CMS_k del grupo en que tiene lugar la operación siguiente de cada candidato (alternativamente las 2, 3, ... operaciones siguientes, aunque ello es difícil) para evitar dar prioridad a un candidato, que posteriormente quedará retenido frente a un grupo cuello de botella con mucha cola, frente a otro que tiene el resto del camino libre.

En un momento dado el estado de carga de un grupo está definido por el trabajo pendiente en curso más el de la cola (normalizado por la capacidad de producción, debe poder

medirse en horas). Sin embargo, dicho estado varía continuamente, y un estado previsional dará tantas o más complicaciones que el *backtracking*.

El procedimiento queda como sigue:

Inicialización. Se sitúan los lanzamientos donde les corresponde: en cola, en elaboración (o ambos si hay solapamientos).

Aplicación del algoritmo.

Presentación de resultados.

Intervención del operador y evaluación interactiva.

Archivo.

Previamente hay que proceder a:

- eliminar lanzamientos acabados,

- modificar la situación de lanzamientos en curso,

- añadir (emitir) lanzamientos de acuerdo con la cartera de pedidos.

6.1.9 Concepto de OPT

Las dificultades que presenta en ciertos casos complejos la transformación del plan de necesidades en un programa de operaciones ha llevado al desarrollo de técnicas especiales, normalmente apoyadas en un paquete informático, para obtener lo que un autor denomina SSS (*smart scheduling system*) o más comúnmente EST (*enhanced schedule technologies*). La más conocida de las EST es la que responde a las siglas OPT, y se produce la paradoja de que en los textos se trate como un procedimiento de ámbito extenso lo que correponde propiamente al nombre registrado de un paquete informático. Seguramente a ello contribuyen el histrionismo y buen sentido comercial del Dr. Eliyahu Goldratt (calificado como uno de los nuevos "gurus del management") autor de los libros *La meta* y *La carrera*.

Bajo la denominación de sistema OPT (*Optimized Production Technology*) se comercializa un paquete informático de programación y control de la producción que proclama como principal característica el tener en cuenta explícitamente las consideraciones de capacidad para obtener el programa detallado de las operaciones.

El sistema MRP realiza la explosión de necesidades empleando tres simplificaciones fundamentales:

- capacidad infinita,
- lotes de fabricación constantes y predeterminados,
- plazos de fabricación constantes y predeterminados,

la consideración de capacidad finita lleva o bien a modificar el plan maestro, o bien a desplazar las órdenes en períodos completos, preservando el tamaño de los lotes. Las consecuencias de las simplificaciones anteriores son que los programas obtenidos no son realistas, y su modificación para adaptarlos a la compleja realidad productiva entra en conflicto con la lógica del sistema MRP. La "filosofía" del OPT se basa en los nueve principios o reglas siguientes:

1) Debe equilibrarse el flujo de producción en lugar de la capacidad de las secciones, lo que implica que el programa de producción debe tener en cuenta las limitaciones de capacidad en su preparación. Lo opuesto es la política de determinar dicho programa (el flujo de producción) y, "a posteriori", modificarlo para equilibrar la carga en exceso que se requiere. Además intenta transmitir el concepto de que el enfoque inicial debe ser el de obtener un programa de producción que dé lugar a un flujo equilibrado sin grandes stocks intermedios en las secciones.

2) El nivel de utilización de un recurso no saturado no depende de su propia capacidad sino de alguna otra limitación del sistema productivo. Normalmente los procesos productivos tienen varias etapas por las que fluyen los materiales en diversos estadios de elaboración; la saturación de capacidad en una sección repercute sobre las secciones anteriores y posteriores. Sobre las posteriores ya que la salida de una etapa es la entrada de la siguiente, y las entradas de las etapas posteriores a una saturada quedan limitadas por la capacidad de producción de ésta. Sobre las anteriores, porque si dichas etapas no tienen en cuenta la saturación que existe posteriormente se generará un stock que aumentará continuamente. La sección saturada es el "cuello de botella" de todo el proceso.

3) La utilización y la activación de un recurso son conceptos distintos, existe diferencia entre la utilización útil o inútil de un recurso. Si una sección tiene capacidad superior a la necesaria para la ejecución del programa, y la utiliza, lo único que logrará es crear stocks inútiles; sus recursos están siendo utilizados, pero sólo están activados cuando producen las unidades requeridas por el programa.

4) Las ineficiencias en los recursos saturados repercuten sobre todo el sistema, puesto que constituyen el cuello de botella para el cumplimiento del programa de producción. Los cuellos de botella son los que definen, con sus limitaciones locales, las limitaciones globales sobre el programa de producción.

5) El ahorro de recursos que no están saturados no repercute sobre el cumplimiento del programa de producción. Los incrementos de productividad en dichos recursos (consumo de recurso/unidad producida) sólo repercute en el incremento de exceso de capacidad existente, pero no en la producción del sistema global. En estos recursos sus limitaciones no son limitaciones globales del sistema (son locales pero no globales).

6) Los recursos saturados determinan la tasa de producción y el nivel de stocks del sistema productivo. Si el flujo de producción no está equilibrado se creará stock en la sección cuello de botella pues no será capaz de procesar todas las entradas a la misma, y en las posteriores se producirá carestía. Por tanto, el programa de producción y el nivel de stocks deben regularse teniendo en cuenta los recursos saturados. El programa de producción no debe plantear exigencias superiores a la capacidad de producción de los cuellos de botella. Los stocks deben regularse para preservar el sistema contra posibles ineficiencias de los cuellos de botella. Cualquier circunstancia que disminuya momentáneamente la capacidad de producción del cuello de botella repercute en la producción global del sistema por lo que no puede permitirse, por ejemplo, que el cuello de botella deje de producir por falta de alimentación. Podría pensarse que este stock de seguridad en las secciones cuello de botella también es necesario en las secciones posteriores a las mismas; aunque fuese deseable no es posible, pues se trataría de material ya elaborado en el cuello de botella, y mantenerlo como stock sin darle salida reduciría la producción global del sistema.

7) El lote de fabricación no tiene por qué coincidir con el lote de transferencia entre secciones; a menudo conviene que sea distinto. El lanzamiento de una serie de producción requiere la puesta a punto de las máquinas y de los utillajes que intervienen, es decir, un consumo de recursos ligado al lanzamiento, antes de empezar a producir unidades e independiente del tamaño de la serie. Debido a ello las puestas a punto se perciben en producción como un mal inevitable que debe disminuirse lo más posible y constituyen una ineficiencia intrínseca que obliga a la producción por lotes. De hecho, todas las teorías sobre el dimensionamiento económico del lote se basan en equilibrar el coste de lanzamiento asociado esencialmente a la preparación distribuyéndolo sobre los elementos de una serie de producción de tamaño adecuado; a mayor coste mayor tamaño. Este razonamiento es lógico e inatacable, pero se aplica únicamente al lote de fabricación. Habitualmente se emplea un lote de transferencia del mismo tamaño que el de producción, es decir, se espera a que se hayan producido todas las unidades del lote antes de trasladarlas a la sección siguiente. El resultado de esta forma de actuar es que el tamaño de las series es excesivamente grande en el conjunto de las instalaciones, lo que da lugar a un plazo total, suma de plazos individuales, excesivamente grande. Un procedimiento razonable debería tener en cuenta la capacidad de las secciones que intervienen en la fabricación. En las secciones saturadas debe minimizarse el número de puestas a punto para poder fabricar más unidades utilizando lotes de fabricación grandes. Sin embargo, el lote de transferencia debe reducirse para posibilitar que las secciones con exceso de

capacidad que reciben dicho material empleen lotes de fabricación más pequeños. Estas secciones no saturadas pueden emplear eficientemente su holgura de capacidad aumentando las frecuencias de las puestas a punto, con lo que disminuyen los stocks intermedios así como el plazo acumulado de fabricación. La partición de los lotes de fabricación en varios lotes de transferencia debe ser la práctica habitual recomendada y no la excepción.

8) El lote de fabricación no debe ser el mismo en todas las secciones, tampoco debe ser constante en el horizonte de producción. Parte de la argumentación se ha dado en el párrafo anterior. Al considerar la realización de un programa de produción se observa que a lo largo del tiempo se suelen producir situaciones de saturación transitoria en algunas secciones; ello se debe a que el plan maestro detallado requiere un número variable de productos acabados en cada uno de los intervalos, lo que comporta que las necesidades de carga que se solicitan a las secciones no sean constantes a lo largo del horizonte. Por otra parte las incidencias que vayan produciéndose pueden amplificar esta variabilidad. Para tener en cuenta el comportamiento dinámico del sistema, en el que los cuellos de botella son variables, es adecuado responder con lotes de fabricación variables; en situaciones de saturación, lotes mayores, eliminando puestas a punto; con exceso de capacidad, lotes reducidos empleando la capacidad sobrante en la preparación de las series.

9) Las prioridades deben fijarse teniendo en cuenta simultáneamente todas las limitaciones de recursos; el plazo de fabricación no es constante sino dependiente del plan maestro detallado de producción. Ante retrasos en el desarrollo del programa de producción se reacciona mediante el establecimiento de listas de urgencias para poder completar los pedidos que tengan prioridad, estas listas suelen establecerse con graduaciones de urgencia: prioridad 1, prioridad 2, etc. La fijación de reglas de prioridad es compleja y suele basarse en criterios tales como el retraso respecto a la fecha comprometida, o bien en el margen (diferencia entre el tiempo que queda hasta la fecha comprometida y los plazos acumulados de fabricación que todavía restan). La fijación de prioridades absolutas de esta índole, sin tener en cuenta su repercusión sobre el resto de fabricaciones, puede dar lugar al caos. El adelantamiento de órdenes debido a las prioridades suele conducir a interrumpir series de fabricación para iniciar otras nuevas en lotes más reducidos, aumentando el número de puestas a punto. Si esto ocurre en secciones que son cuello de botella, se disminuye su capacidad y por tanto la de todo el sistema, con lo que aumentan los retrasos y se deteriora el cumplimento de todo el programa de producción. Esto no significa que puedan evitarse las listas de urgencias, sino que las prioridades deben establecerse teniendo en cuenta sus implicaciones para todo el sistema.

Estas nueve reglas que el OPT describe como la introducción de una nueva cultura en la gestión de la producción no pueden considerarse como aportaciones originales. Son verbalización sistemática de las consecuencias del análisis de un sistema complejo, con múltiples etapas y limitaciones de capacidad. Su mérito, si existe, está en el énfasis de su

crítica sobre las consecuencias del exceso de simplificación en la derivación de sistemas de programación y control a partir del cálculo de necesidades tipo MRP, enfocando el análisis hacia la obtención de soluciones que tengan en cuenta las limitaciones globales del sistema (los cuellos de botella) y el carácter dinámico del comportamiento del mismo. Conviene destacar dos puntos más de interés:

10) Los cuellos de botella no existen, sino que se identifican. En algunas instalaciones son siempre las mismas secciones las que se saturan, con lo que dan lugar a situaciones de cuellos de botella permanentes. Esto sucede cuando la capacidad de dichas secciones está claramente infradimensionada respecto a las necesidades de la fabricación. En este caso podríamos decir que en dicha planta existen cuellos de botella. Pero en la mayoría de los casos los cuellos de botella no son una característica de las secciones sino una combinación de la capacidad de éstas con el plan de producción que se desea realizar; existe un grupo de secciones que potencialmente son cuello de botella: la explosión del plan maestro concreto que se pretende realizar identifica aquéllas que actúan efectivamente como tales para dicho plan, pero tal vez no para otro. Nada permite en general afirmar con toda seguridad que ese plan de producción es el mejor, en el sentido de satisfacer la demanda (en firme y prevista) de la forma más económica, si bien si se han tomado las debidas precauciones al elaborarlo (niveles jerárquicos, factibilidad, evaluación) se puede suponer que es razonablemente bueno. De otra forma habría que decidir el plan maestro teniendo en cuenta *todas sus consecuencias*, y por tanto los cuellos de botella efectivos que produce, etc. lo cual es en general irrealizable.

11) No sólo son críticas las secciones saturadas. Hemos insistido anteriormente sobre las limitaciones globales para la realización del programa de producción producidas por las secciones cuya capacidad se satura. La eficiencia de dichas secciones es crítica para todo el sistema. Pero todas las secciones posteriores, que reciben material ya procesado en los cuellos de botella, aunque dispongan de exceso de capacidad con respecto a sus entradas también son críticas. Cualquier retraso en dichas secciones repercute en el programa de producción, al igual que los rechazos de calidad debidos a su proceso propio; por tanto, su actividad es crítica.

6.1.9.1 El paquete informático OPT

La empresa Creative Output Inc. comercializa un paquete informático de programación y control de la producción basado en las reglas anteriores. La figura 6.1.9.1 recoge un esquema del mismo. Consta de cuatro módulos:

(1) *built net*
(2) *serve*
(3) *split*

(4) *brain*

El paquete carece de un módulo de planificación que determine el plan maestro detallado, por lo que el requisito fundamental para utilizar el paquete es que dicho plan esté fijado previamente. También requiere un módulo de lista de materiales y un módulo de situación de stocks (los creadores del OPT consideran los paquetes MRP como una base de datos de gran utilidad, a la vez que ponen en duda su eficacia como procedimientos de planificación, programación y control). A partir de estos elementos, de los ciclos y rutas de trabajo y de las características de los recursos y centros de trabajo, se construye un grafo o red que modeliza el sistema productivo (módulo 1).

A continuación se procede a la identificación de los centros de trabajo saturados (cuellos de botella) que son aquellos en los que el nivel de utilización de los recursos está próximo o supera el 100 % de la capacidad disponible (módulo 2); esto permite descomponer el grafo inicial en dos subgrafos, el primero de los cuales agrupa las operaciones críticas (secciones saturadas y todas las que las siguen en el proceso de fabricación), y el segundo engloba las restantes secciones en las que no existen problemas de capacidad (módulo 3). Para identificar los cuellos de botella que limitan la ejecución del programa se calcula la fracción de carga (número de horas que se requieren de un recurso dividido por el número de horas disponibles, en el horizonte a que se extiende el programa) que supone la realización de dicho programa si no actuaran las limitaciones de capacidad. Para ello se parte del plan maestro detallado y se realiza la explosión de necesidades tal como se realiza en el MRP, empleando los plazos (*lead time*) constantes incluidos en los ficheros maestros. Como resultado de este análisis se obtiene una relación de secciones con fracciones ordenadas de mayor a menor, un análisis ABC de las mismas permite detectar inmediatamente cuáles son las más saturadas, las que condicionan efectivamente la realización del programa. Un análisis profundo de las secciones de la clase A permite una depuración de posibles errores para ajustar con precisión el grado de saturación de las mismas; a continuación se detallan dos aspectos:

- relaciones de precedencia entre secciones saturadas,
- desagregación de cargas en los cuellos de botella,

Si entre dos secciones saturadas existe una relación de precedencia es muy posible que solo una de ellas actúe como cuello de botella, por ejemplo si la sección anterior tiene un coeficiente mayor que la que le sigue, de la que es precedente único. En cuanto se confeccione un programa factible, si la primera sección puede con la carga, también podrá la segunda. En segundo lugar se analizan las causas de las saturaciones, detallando qué referencias del programa de producción dan lugar a las cargas. Ordenadas de mayor a menor se obtiene la desagregación de las cargas; este análisis ABC de las referencias responsables de las cargas señala las más importantes, aquéllas en las que las actuaciones de incremento de productividad más repercuten sobre el sistema. En la fabricación los

tiempos de elaboración se desdoblan en:

- tiempos de puesta a punto para la serie,
- tiempo necesario para producir una unidad.

Pero sobre los plazos de fabricación también actúan otros factores que traen consigo mayores duraciones:

- tiempo de espera, mientras se reciben en la sección otros componentes que intervienen en el proceso,

- tiempo en cola, debido al tamaño de la serie de fabricación; hasta que no se fabriquen los anteriores de su serie el componente está en cola para ser elaborado.

La regulación del proceso de fabricación repercute sobre estos tiempos. Las disminuciones de los dos primeros aumentan la capacidad de la sección.

Una vez identificados los cuellos de botella se modifica el programa de producción teniendo en cuenta sus limitaciones de capacidad, y para obtener el máximo aprovechamiento de la capacidad de las secciones saturadas se programan en ellas lotes de fabricación grandes y lotes de transferencia pequeños. El procedimiento exacto empleado por este módulo 4 no ha sido hecho público por parte de Creative Output Inc. Tras realizar la programación de los cuellos de botella, se realiza una programación *hacia adelante* en las secciones siguientes que elaboran material que ya ha pasado por el cuello de botella; como se ha dicho ya estas secciones también son críticas y su flujo de material viene condicionado por el que emite el cuello de botella. El programa de producción del cuello de botella actúa como un elemento que "empuja" toda la producción; el sistema OPT da lugar a un método *push* desde los cuellos de botella hacia adelante.

La selección de los niveles de stock de seguridad en las secciones también se lleva a cabo teniendo en cuenta los condicionamientos impuestos por los cuellos de botella; debido a su aspecto crítico debe disponerse de stock de seguridad a la entrada de los mismos, el nivel mínimo indispensable que garantice su funcionamiento continuado, con pleno aprovechamiento de su capacidad, aun cuando se produzcan contingencias que interrumpan la actividad de las secciones que alimentan a los cuellos de botella. Además, si en el grafo de producción confluyen varias ramas en una sección, procedentes unas de secciones saturadas y otras de secciones no saturadas, también debe incorporarse stock de seguridad en la rama no saturada de la sección de confluencia, pues no puede retrasarse la fabricación de los componentes que proceden del cuello de botella debido a que faltan otros componentes. En definitiva, debe emplearse stock de seguridad en las secciones críticas, siempre que esto no suponga un retraso del programa de producción.

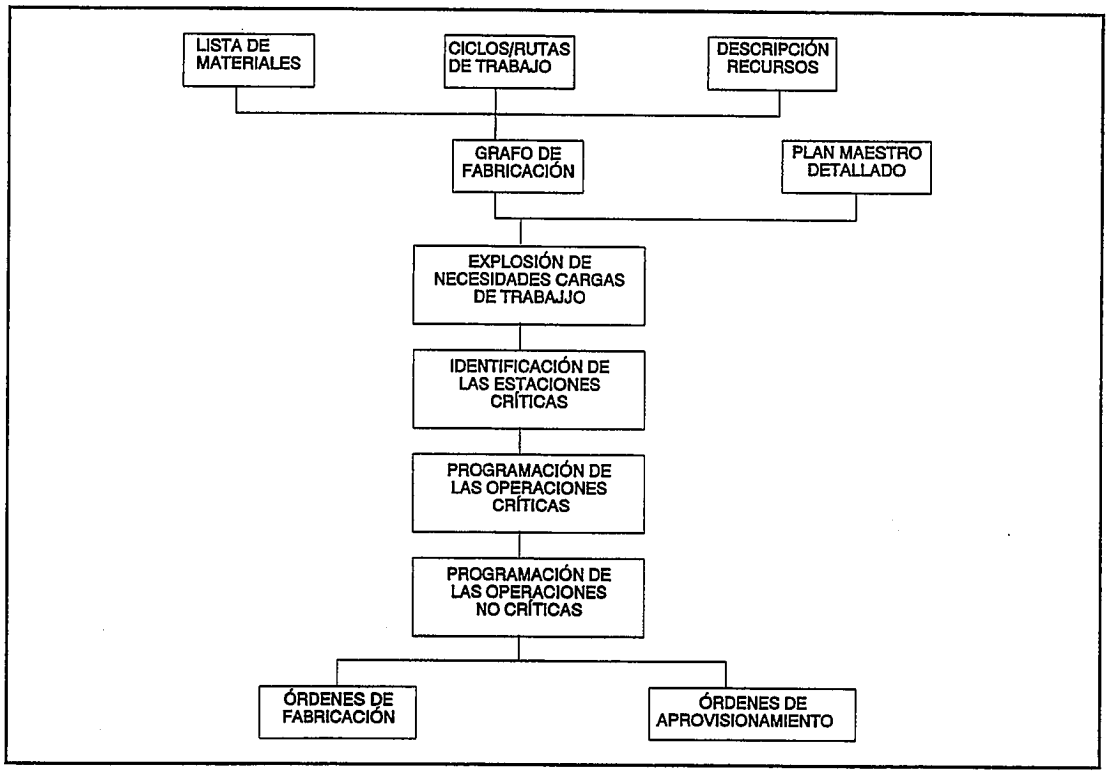


Fig. 6.1.9.1 Esquema del paquete informático OPT

La programación de las secciones no críticas depende de la de las críticas, las secciones que no condicionan el cumplimiento del programa de producción deben regularse para satisfacer las necesidades de material de las secciones críticas que alimentan, su tasa de produción viene determinada por la tasa de producción de las secciones críticas. Por tanto, deben eliminarse los stocks de productos intermedios en estas secciones ya que tienen exceso de capacidad y no son críticas para el cumplimiento del programa. La disminución de los stocks de obra en curso se consigue empleando lotes de fabricación reducidos, aprovechando la holgura de capacidad para aumentar el número de lotes.

En cuanto a los stocks de seguridad, no existe justificación para su empleo. La protección respecto a posibles incidencias se efectúa a la entrada de las secciones críticas. Todos estos aspectos dan lugar a que la programación de la producción en las secciones no críticas se desarrolle "hacia atrás", a partir de las secciones críticas. Es un sistema *pull*, en el que los cuellos de botella y secciones críticas tiran de la producción de las otras secciones.

6.2 Bibliografía

[01] BAKER, K. R.; *Introduction to Sequencing and Scheduling*; Wiley, 1974.

[02] BELLMAN, R.; ESOGBUE, A. O.; NABESHIMA, I.; *Mathematical Aspects of Scheduling and Applications*; Pergamon Press, 1982.

[03] CARLIER, J.; CHRETIENNE, Ph.; *Problèmes d'Ordonnancement*; Masson, 1988.

[04] CONWAY, R. W.; MAXWELL, W. L. MILLER, L. W.; *Theory of Scheduling*; Addison Wesley, 1967.

[05] DEMPSTER, M. A. H.; LENSTRA. J. K.; RINNOOY KAN, A. H. G. (eds.); *Deterministic and Stochastic Scheduling*; D. Reidel Pu. Co., 1981.

[06] FRENCH, S.; *Sequencing and Scheduling*; Wiley, 1982.

[07] FOGARTY, D. W.; HOFFMANN, Th. R.; *Production and Inventory Management*; South-Western Publishing Co., 1983.

[08] MARTELLO, S.; LAPORTE, G.; MINOUX, M.; RIBEIRO, C. (eds.); *Surveys in Combinatorial Optimization*; North-Holland, 1987.

[09] MUTH, J. F.; THOMPSON, G. L.; *Industrial Scheduling*; Prentice Hall, 1963.

Comentarios

La mayoría de trabajos sobre secuenciación se encuentran en revistas especializadas. [04] constituye un tratado básico, a pesar de los años transcurridos desde su publicación. Por su parte [03] contiene un tratamiento más reciente, con algunos aspectos originales. [09] es una colección de artículos, una de las primeras publicaciones sobre el tema, y que contiene elementos que todavía sirven de referencia. El problema "Los cuatro amigos" procede de [06].

Anexo

Reproducimos a continuación un conjunto de subrutinas aplicables a los problemas $n/m/P/F_{max}$. Permiten determinar las cotas longitudinales y transversales y las secuencias dadas por los procedimientos trapecios y Palmer, calculando el valor F_{max}. También permiten mejorar la solución obtenida mediante el procedimiento RAES.

```
'.....número de piezas n y número de máquinas m
'.....duraciones en la matriz p(j,i)
'.....cotas en cotl, cott y cotanp
'.....soluciones en el vector k(i)
'.....índices de Palmer en los vectores s1, s2 y s3
'.....instantes de liberación de las máquinas después
'.....de cada operación de f(j,i)
'.....Fmax en f(m,n)
calculcot:          '...cota longitudinal y transversal
cotl = 0
FOR j = 1 TO m
    i1 = 0
    IF j = 1 THEN
        cot1 = 0: cot2 = 0
    ELSE
        cot1 = 9999: cot2 = 9999
        FOR i = 1 TO n
            s = 0
            FOR jj = 1 TO j - 1
                s = s + p(jj, i)
            NEXT jj
            IF s < cot2 THEN
                cot2 = s
                IF cot2 < cot1 THEN
                    SWAP cot1, cot2
                    i1 = i
                END IF
            END IF
        NEXT i
    END IF
    cot3 = 0
    FOR i = 1 TO n
        cot3 = cot3 + p(j, i)
    NEXT i
    i2 = 0
    IF j = m THEN
        cot4 = 0: cot5 = 0
    ELSE
        cot4 = 9999: cot5 = 9999
        FOR i = 1 TO n
            s = 0
            FOR jj = j + 1 TO m
                s = s + p(jj, i)
            NEXT jj
            IF s < cot5 THEN
                cot5 = s
                IF cot5 < cot4 THEN
                    SWAP cot4, cot5
                    i2 = i
                END IF
```

```
            END IF
        NEXT i
    END IF
    IF i1 = i2 THEN
        x = cot1 + cot5
        IF x > cot2 + cot4 THEN x = cot2 + cot4
    ELSE
        x = cot1 + cot4
    END IF
    cotj = cot3 + x
    IF cotl < cotj THEN cotl = cotj
NEXT j

cott = 0
FOR i = 1 TO n                          'válido sólo para
    s3(i) = p(1, i)                     'el caso P
    IF s3(i) > p(m, i) THEN s3(i) = p(m, i)
NEXT i
FOR i = 1 TO n
    coti = 0
    FOR j = 1 TO m
        coti = coti + p(j, i)
    NEXT j
    FOR ii = 1 TO n
        IF ii <> i THEN coti = coti + s3(ii)
    NEXT ii
    IF cott < coti THEN cott = coti
NEXT i
RETURN

calcúlese:      '...índices de Palmer
FOR i = 1 TO n
    s1(i) = 0: s2(i) = 0
    FOR j = 1 TO m
        s1(i) = s1(i) + (m - j) * p(j, i)
        s2(i) = s2(i) + (j - 1) * p(j, i)
    NEXT j
    s3(i) = s1(i) - s2(i)
NEXT i
RETURN
trapecios:      '...ordenación Trapecios
FOR i = 1 TO n
    k(i) = i
NEXT i
alfa = 1
WHILE alfa = 1
    alfa = 0
    FOR i = 1 TO n - 1
        i1 = k(i): i2 = k(i + 1)
        x = s1(i1)
        IF x > s2(i2) THEN x = s2(i2)
        y = s1(i2)
```

```
        IF y > s2(i1) THEN y = s2(i1)
        IF x > y THEN
            GOSUB permuta
            ELSEIF x = y THEN
                IF s3(i1) > s3(i2) THEN
                GOSUB permuta
                ELSEIF s3(i1) = s3(i2) THEN
                    IF p(1, i1) > p(1, i2) THEN
                        GOSUB permuta
                    END IF
                END IF
            END IF
    NEXT i
WEND
RETURN

permuta:        '...auxiliar para trapecios y Palmer
SWAP k(i), k(i + 1)
alfa = 1
RETURN


calcular:       '...determinación de f(j,i)
FOR j = 1 TO m
    FOR i = 1 TO n
        i0 = k(i)
        fp = f(j, i - 1)
        IF fp < f(j - 1, i) THEN fp = f(j - 1, i)
        f(j, i) = fp + p(j, i0)
    NEXT i
NEXT j
RETURN

palmer:         '...ordenación Palmer
FOR i = 1 TO n
    k(i) = i
NEXT i
alfa = 1
WHILE alfa = 1
    alfa = 0
    FOR i = 1 TO n - 1
        i1 = k(i): i2 = k(i + 1)
        IF s3(i1) > s3(i2) THEN
            GOSUB permuta
        ELSEIF s3(i1) = s3(i2) THEN
            IF s1(i1) = s1(i2) AND p(1, i1) > p(1, i2) THEN
                GOSUB permuta
            END IF
            IF s3(i1) * (s1(i1) - s1(i2)) < 0 THEN
                GOSUB permuta
            END IF
        END IF
```

```
    NEXT i
WEND
RETURN

raes:           '...mejora por intercambio
fmax = f(m, n)
alfa = 1
WHILE alfa = 1
    alfa = 0
    FOR i1 = 1 TO n - 1
        FOR i2 = i1 + 1 TO n
          SWAP k(i1), k(i2)
          GOSUB calculaf
          IF fmax > f(m, n) THEN
                fmax = f(m, n)
                alfa = 1
          ELSE
                SWAP k(i1), k(i2)
          END IF
        NEXT i2
     NEXT i1
WEND
RETURN

calsupcot:      '...cota de NABESHIMA-POTTS
cotanp = 0
FOR j1 = 1 TO m - 1
      FOR j2 = j1 + 1 TO m
          '.....cálculo tiempos intermáquinas
          FOR i = 1 TO n
                q(i) = 0
                k(i) = i
                FOR j = j1 + 1 TO j2 - 1
                      q(i) = q(i) + p(j, i)
                NEXT j
          NEXT i
          '.....cálculo tiempo de maq. 1 a j1-1
          cap = 9999
          FOR i = 1 TO n
                s = 0
                FOR j = 1 TO j1 - 1
                      s = s + p(j, i)
                NEXT j
                IF cap > s THEN cap = s
          NEXT i
          IF j1 = 1 THEN cap = 0
          '.....cálculo tiempo de maq. j2+1 a m
          cua = 9999
          FOR i = 1 TO n
                s = 0
                FOR j = j2 + 1 TO m
                      s = s + p(j, i)
```

```
                NEXT j
                IF cua > s THEN cua = s
            NEXT i
            IF j2 = m THEN cua = 0
            '.....ordenación piezas
            alfa = 1
            WHILE alfa = 1
                alfa = 0
                FOR i = 1 TO n - 1
                    i1 = k(i)
                    i2 = k(i + 1)
                    GOSUB ordena
                NEXT i
            WEND
            f1 = 0
            f2 = 0
            FOR i = 1 TO n
                ii = k(i)
                f1 = f1 + p(j1, ii)
                s = f1 + q(ii)
                IF s < f2 THEN s = f2
                f2 = s + p(j2, ii)
            NEXT i
            IF cotanp < cap + f2 + cua THEN cotanp = cap + f2 + cua
        NEXT j2
NEXT j1
RETURN

ordena:         '...auxiliar para calsupcot
x = p(j1, i1) + q(i1)
IF x > p(j2, i2) + q(i2) THEN x = p(j2, i2) + q(i2)
y = p(j1, i2) + q(i2)
IF y > p(j2, i1) + q(i1) THEN y = p(j2, i1) + q(i1)
IF x > y THEN
    SWAP k(i), k(i + 1)
    alfa = 1
END IF
IF x = y THEN
    IF p(j1, i1) - p(j2, i1) > p(j1, i2) - p(j2, i2) THEN
        SWAP k(i), k(i + 1)
        alfa = 1
    ELSEIF p(j1, i1) - p(j2, i1) = p(j1, i2) - p(j2, i2) THEN
        IF p(j1, i1) > p(j1, i2) THEN
            SWAP k(i), k(i + 1)
            alfa = 1
        END IF
    END IF
END IF
RETURN
```

6.3 Enunciados

Enunciado 6.3.1

En un Centro de Análisis, un equipo de médicos analistas ha citado a 5 enfermos con el siguiente plan de trabajo:

PACIENTE		ANAL.1		ANAL.2		ANAL.3		ANAL.4		ANAL.5	
Nº	hora llegada	Dr.	dur	Dr.	dur	Dr.	dur	Dr.	dur	Dr.	dur
1	8h 30m	A	20m	B	50m	C	30m	D	25m	A	10m
2	9h	E	20m	D	25m	C	30m	A	50m	E	10m
3	9h 30m	A	20m	E	50m	D	25m	C	30m	A	10m
4	10h	E	20m	D	25m	C	30m	B	50m	E	10m
5	10h 30m	A	20m	E	30m	B	50m	A	10m	-	--

donde ANAL. significa ANALISIS
Dr. significa doctor
dur significa duración

a) ¿Cómo deben programarse las actividades de los médicos para terminar el ciclo lo antes posible?

Los médicos estarán disponibles en el Centro a la hora que se precise a partir de las 8h 30m. Un médico sólo atiende a un paciente a la vez, y un paciente sólo está sometido a un análisis a la vez.

b) Sabiendo que el primer y el último análisis de cada enfermo puede ser realizado indistintamente por los doctores A y E (no así el 4º del paciente 2 que lo debe realizar el Dr. A y el 2º de los pacientes 3º y 5º que lo debe realizar el Dr. E, ¿podría mediante una reestructuración del plan de trabajo lograrse una reducción del tiempo? ¿En qué medida y cómo?

c) Manteniendo el plan de trabajo original podría cambiarse el orden de llegadas de los pacientes (manteniendo los 30 minutos de intervalo). ¿Qué orden de llegadas sería el mejor para reducir el plazo total?

d) Id. alterando el Plan de Trabajo según lo indicado en b).

Enunciado 6.3.2

a) Un equipo formado por tres médicos analistas, cada uno de ellos encargado de un tipo específico de examen y análisis debe visitar 7 pacientes el lunes de la próxima semana en su centro médico. La duración de cada visita es en función del binomio (doctor-paciente) y se evalúa aproximadamente como sigue:

	Doctor A	Doctor B	Doctor C
Paciente a	25 min.	40 min.	25 min.
Paciente b	50 min.	35 min.	20 min.
Paciente c	35 min.	20 min.	50 min.
Paciente d	20 min.	45 min.	30 min.
Paciente e	30 min.	25 min.	40 min.
Paciente f	45 min.	30 min.	45 min.
Paciente g	40 min.	50 min.	35 min.

Los tres doctores llegan al centro un poco antes de las 9 de la mañana y están disponibles a partir de dicha hora. Siempre visitan a los pacientes por orden: primero el doctor A, luego el B y finalmente el C. Los usos médicos quieren que se cite a los pacientes 15 minutos antes de que se prevea que los pueda visitar el doctor A. La enfermera encargada de citar telefónicamente a los pacientes desea hacerlo de forma que la mañana del lunes quede libre de pacientes lo antes posible.

¿A qué hora debe citar a cada paciente?

b) Resolver el mismo problema en el supuesto de que el orden en que los pacientes ven a los doctores está prefijado pero no es siempre el ABC, sino:

Paciente	a	b	c	d	e	f	g
1ª visita	A	A	A	B	B	C	C
2ª visita	B	B	C	A	C	A	B
3ª visita	C	C	B	C	A	B	A

c) Resolver el problema anterior en el supuesto de que el orden en que los doctores ven a los pacientes no está prefijado, y que la enfermera puede decidirlo antes de citarlos.

Enunciado 6.3.3

Los señores XAVIER, YAGO y ZOILO de CONSULTORES DE INDUSTRIAS MANUFACTU-

RERAS Y DE SERVICIOS (CIMS) deben realizar las Declaraciones Fiscales de 5 empresas: MUNIESA, NIQUI, OPSEA, PACSA y QUISA. El Sr. TOMAS, al programar los trabajos, considera las siguientes secuencias de operaciones y duraciones:

	Operación 1	Operación 2	Operación 3	Operación 4	Operación 5	Operación 6
MUNIESA	X: 2 h	Y: 3 h	Z: 5 h	Y: 4 h	X: 1 h	
NIQUI	X: 2 h	Z: 3 h	Y: 5 h	Z: 3 h	X: 1 h	
OPSEA	Y: 2 h	X: 3 h	Z: 5 h	X: 4 h	Y: 1 h	
PACSA	Y: 2 h	Z: 4 h	X: 5 h	Z: 2 h	Y: 1 h	
QUISA	Z: 2 h	X: 3 h	Y: 3 h	X: 4 h	Y: 4 h	Z: 1h

(h = hora)

¿Cómo debe programar los trabajos para terminar lo antes posible, sabiendo que todos los consultores estarán disponibles desde el día 28-06-1995 a las 8 h? Los consultores trabajan 8 horas/día, pero si es necesario pueden quedarse 2 horas más. ¿Alteraría la programación el hecho de saber que la Declaración de OPSEA es la más urgente?

Enunciado 6.3.4

Tres miembros de la empresa de estudios CONSULTORES DE INDUSTRIAS MANUFACTURERAS Y DE SERVICIOS (CIMS), Sres. XAVIER, YAGO y ZOILO deben analizar los Planes Estratégicos de cinco empresas (ALMUNIA, BANEX, COMPSA, DYPSA y ENYSA) y realizar determinados estudios, cálculos e informes. Una primera estimación de los trabajos a realizar ha establecido el orden de las actuaciones y su duración.

Empresa	A	B	C	D	E
1ª actuación	X 2 h	X 3 h	X 1 h	Y 3 h	Y 2 h
2ª actuación	Y 1 h	Z 2 h	Z 3 h	Z 2 h	X 4 h
3ª actuación	Z 4 h	Y 2 h	Y 3 h	X 2 h	Z 1 h
4ª actuación	X 1 h	Z 1 h	X 2 h	Z 2 h	Y 1 h
5ª actuación	------	X 1 h	------	Y 1 h	------

Puesto que todos los trabajos son urgentes, el Sr. TOMÁS, encargado de la planificación de los mismos, desea establecer un programa de actuación a fin de que el último de ellos que termine lo haga lo más pronto posible. Todos los trabajos están disponibles en el momento inicial (lunes 2 de febrero) para comenzar con ellos. Las actuaciones deben

realizarse secuencialmente, sin posibilidad de solapar dos actuaciones relativas al mismo trabajo, y en el orden indicado. Los tres consultores están dedicados con prioridad absoluta a las mismas. Si al llegar al término de la jornada laboral (de 8 horas efectivas diarias) una actuación se halla parcialmente realizada se prosigue a partir de la primera hora del día siguiente sin que la interrupción cause ningún efecto sobre la duración total. ¿Qué programa puede proponer el Sr. TOMÁS?

Enunciado 6.3.5

Un taller/estación de servicio debe atender a cinco vehículos industriales.

Según recepción, la secuencia de secciones por los que han de pasar y los tiempos estimados de trabajo en cada una de ellas son los que figuran a continuación:

	V_1		V_2		V_3		V_4		V_5	
Operac.	Secc.	Min.	Secc.	Min.	Sec.	Min.	Secc.	Min.	Secc.	Min.
1	Acce.	40	Prep.	30	Acce.	90	Prep.	70	Prep.	140
2	Prep.	80	Meca.	30	Elec.	60	Elec.	110	Acce.	120
3	Meca.	100	Elec.	90	Chap.	140	Meca.	110	Chap.	60
4	Elec.	60	Prep.	70	Meca.	100	Chap.	80	Meca.	50
5	Chap.	120								

Cada sección sólo puede trabajar con un vehículo a la vez.

A primera hora están disponibles todos los vehículos, pero las secciones Prep. y Meca. tienen 50 minutos comprometidos, y la Chap., 3 horas y 20 minutos.

El trabajo sobre el vehículo V_4 es de máxima prioridad pues hay un compromiso de tenerlo listo al cabo de 7 horas.

Confeccione un programa de taller que intente minimizar el tiempo total necesario para terminar con todos los vehículos. Represéntelo gráficamente.

Enunciado 6.3.6

Para montar un determinado subconjunto se requiere disponer simultáneamente de cuatro

piezas que se elaboran en una misma planta. Cada pieza, para su fabricación, debe ser objeto de las siguientes operaciones (duración expresada en horas):

PIEZA	001		002		003		004	
N° operación	máq.	dur.	máq.	dur.	máq.	dur.	máq.	dur.
1	A	5	A	7	B	8	A	3
2	B	6	C	9	C	10	C	10
3	C	7	D	6	E	12	B	9
4	D	8	E	5	D	6	D	8
5	-	-	-	-	-	-	E	8

a) Determinar un programa de realización de las operaciones que intente minimizar el plazo hasta la iniciación del montaje, suponiendo que las máquinas, de las que sólo hay una de cada tipo, están disponibles para dicha fabricación en los siguientes instantes:

Máquina	A	B	C	D	E
Disponible a la hora	6	0	8	14	22

b) Si a partir de este momento se destinase la maquinaria exclusivamente a la fabricación de piezas para el subconjunto del que se desea montar un importante lote, ¿cuál sería la tasa de producción en estado de régimen?

El montaje tiene una duración de 20 horas.

c) ¿Cómo enfocaría la resolución de los apartados a) y b) si le comunicasen que acaba de instalarse una segunda máquina del tipo C idéntica a la existente?

Enunciado 6.3.7

Determinar el programa que minimiza F_{max} en el siguiente problema de flow-shop, limitándose a las permutaciones:

		PIEZAS					
		1	2	3	4	5	6
Máquinas	A	3	2	1	6	8	4
	B	7	2	3	6	10	3
	C	5	8	10	4	5	6
	D	4	6	7	5	6	8

¿Qué ocurre si no nos limitamos a las permutaciones?

Enunciado 6.3.8

En el taller del Sr. Pedret, se deben efectuar cuatro trabajos, todos ellos tienen que pasar sucesivamente por las cinco máquinas que hay en el taller. La duración (en minutos) del trabajo por hacer en cada una de las máquinas es la que se expresa en la tabla adjunta:

		MAQUINA				
		1	2	3	4	5
TRABAJO	1	20	10	60	70	100
	2	60	50	40	20	90
	3	60	10	50	50	70
	4	30	80	10	70	80

y el Sr. Pedret se pregunta en qué orden debería hacer los trabajos a fin de terminar el último lo antes posible.

Para no complicarse la vida, el Sr. Pedret quiere que en cada máquina se hagan los trabajos en el mismo orden. ¿Qué orden recomendaría, y cuánto tiempo se tardaría en tener hecho todo el trabajo?

El Sr. Pedret sabe que en el taller de su amigo Tomeu tienen una máquina automática capaz de hacer las tres operaciones iniciales en una sola, la cual duraría lo siguiente:

$$p_{k,i} = \max \{ 0'8 \cdot \sum_{j=1}^{3} p_{j,i} \; ; \; 1{,}5 \cdot \max_{j=1,2,3} p_{j,i} \}$$

Además, naturalmente haría falta hacer también las operaciones 4 y 5 en las máquinas de las que dispone. Como que lo que quiere es servir el trabajo lo antes posible, se pregunta si acudiendo a su amigo podría terminarlo antes. ¿Qué opina de ello?

Finalmente, como último recurso, considera la posibilidad de que los trabajos se puedan hacer en cada máquina en un orden diferente; ¿cuál sería la ordenación que para cada máquina aconsejaría en este caso?, ¿cuánto tardaría en tener todos los trabajos terminados?

Enunciado 6.3.9

Un cliente nos ha encargado un trabajo que consiste en fabricar siete componentes (A, B, C, D, E, F y G), y montarlos después. Para hacerlos, deben intervenir sucesivamente las cuatro secciones de que consta el taller. El tiempo de proceso (en horas) consta en la siguiente Tabla:

Trabajo	Sección 1	Sección 2	Sección 3	Sección 4
A	7	3	5	4
B	2	6	9	1
C	8	7	7	3
D	2	4	4	6
E	3	9	6	5
F	1	4	2	3
G	2	0	8	1

El montaje puede empezar sólo una vez terminados los siete componentes, y se tarda 19 horas en lograrlo.

Hemos pactado con un cliente un precio total de 1.200.000 um si terminamos el pedido en un intervalo de tiempo comprendido entre 70 y 80 horas. Si lo terminamos en menos de 70 horas, podremos, además, facturar un incremento de 50000 um por cada hora menos.

Se pide:

a) Dada la estructura del problema, de todas las soluciones posibles sólo hace falta

explorar una clase especial que tienen una cierta propiedad. ¿Cuál es esta propiedad? ¿Quiere decir esto que no puede haber soluciones óptimas fuera de esta clase? Si puede haber, ¿por qué no las hemos de tener en cuenta?

b) Calcular una cota inferior (lo más ajustada posible) del tiempo mínimo que se puede tardar en hacer este trabajo.

c) Limitándonos sólo a las soluciones en las que el orden de proceso de los componentes es el mismo en las cuatro secciones, dar dos soluciones al problema. ¿Qué importe podríamos facturar con cada una?

d) Sabemos que si sólo hubiera dos máquinas, hay cierto algoritmo que nos daría la solución exacta. Pues bien, si aplicamos este algorismo a las secciones 1 y 2 , y después lo aplicamos a las secciones 3 y 4, y hacemos que cada sección fabrique las piezas de acuerdo con el orden que nos resulte, la solución obtenida ¿sería óptima? ¿por qué? (No hay que hacer cálculos; sólo razonar lógicamente).

e) Intentar mejorar las soluciones obtenidas en el apartado c) aplicando un método de *dispatching*.

Enunciado 6.3.10

Uno de los problemas que se presenta periódicamente a ALEPH es la programación del Taller de Reparaciones y Mantenimiento. En este momento para una determinada orden de reparación es necesario elaborar cuatro piezas que luego deben montarse conjuntamente. Por ello interesa terminar las cuatro lo antes posible. Los datos técnicos son los siguientes (duraciones en horas):

PIEZA	001		002		003		004	
Nº operación	máq.	dur.	máq.	dur.	máq.	dur.	máq.	dur.
1	A	8	B	8	A	7	A	9
2	B	10	C	10	C	12	B	8
3	D	7	D	6	D	9	C	5
4	E	6	E	8	E	5	D	3
5	-	-	-	-	-	-	E	6

Las cinco máquinas que intervienen en el proceso estarán disponibles para llevarlo a cabo a partir de los siguientes instantes contados (en horas) a partir de las 6 horas del miércoles 2 de septiembre de 1996. (Este taller trabaja cada día a razón de dos turnos de 8 horas y no realiza horas extra bajo ningún concepto).

Máquina	A	B	C	D	E
Instante disponibilidad	6	0	8	14	22

a) ¿Qué tipo de problema se nos presenta?

b) ¿Qué procedimientos se pueden utilizar para resolverlo?

c) Determinar un programa procurando que las cuatro piezas estén dispuestas para ser montadas lo antes posible. ¿Qué día y a qué hora ocurrirá esto?

d) Si todas las máquinas estuviesen disponibles simultáneamente (por ejemplo el día 2-09-1996 a las 6 h), ¿cambiarían la estructura del problema y los procedimientos de resolución posibles?

e) Si se tuviese que montar un segundo conjunto idéntico al primero, y dispusiéramos de las máquinas cuando las dejan libres las piezas del primero, ¿nos costaría el doble de tiempo tener las piezas del segundo conjunto que tener las del primero? ¿Cuándo podríamos disponer de este segundo conjunto?

Enunciado 6.3.11

El director de Planificación de la Producción de los Talleres de Reparación, Acondicionamiento y Montaje de Productos Artesanos (TRAMPA, S.L.) tiene que servir tres encargos urgentes, que deben salir juntos en una misma furgoneta de reparto. Estos encargos se hacen a partir del montaje de siete piezas que hace falta elaborar previamente. En un taller hay ocho máquinas, y la secuencia de operaciones y el tiempo necesario (en minutos) para la realización de cada una de ellas sigue el esquema indicado en la tabla adjunta.

Piensa encargar este trabajo a dos operarios, Xavier y Zenón, los cuales forman un equipo experto y bien avenido. Las máquinas son automáticas, de manera que un solo operario puede llevar más de una a la vez; concretamente, Xavier se encargará de las máquinas de la 1 a la 5 simultáneamente, y al mismo tiempo Zenón se ocupará de las máquina 6, 7 y 8.

Los trabajadores empiezan su jornada laboral a las ocho de la mañana, y es costumbre que la interrumpan para desayunar sobre las once o como muy tarde las doce.

Como se trata de tres trabajos con rutas casi idénticas, el Director piensa que es recomendable que Xavier empiece a hacer el montaje cuando acabe con la tarea de las máquinas 1 hasta la 5, y Zenón vaya a hacer la expedición cuando acabe la de las máquinas 6 hasta la 8. Para terminar todavía más pronto, y a fin de poder desayunar

juntos, si Xavier termina el montaje cuando todavía hay algún encargo por expedir, puede dedicarse a ello, con la condición de que no pueden trabajar los dos al mismo tiempo en el mismo encargo, pero sí en dos diferentes.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	Mont.	Exp.
Encargo 1										
a	8 →	6 →	6 →	7 →	3					
b						12 →	5 →	9	→ 25	→ 95
c						4 →	6 →	11		
Encargo 2										
d	3 →	7 →	5 →	4 →	6					
e						5 →	0 →	8	→ 47	→ 76
Encargo 3										
f						7 →	4 →	9		
g						5 →	7 →	13	→ 18	→ 64
h						4 →	5 →	8		

¿Cómo debería organizar el trabajo? Podrían, Xavier y Zenón, de acuerdo con esto, terminar los tres encargos antes de ir a desayunar? Si es así, ¿a qué hora irían? Por el contrario, ¿a qué hora podría salir la furgoneta, teniendo en cuenta que para desayunar están veinte minutos?

Enunciado 6.3.12

Resolver los siguientes supuestos de problemas de Taller Mecánico. En cada caso se debe responder a las siguientes preguntas:

1. ¿Qué algoritmo o procedimiento se aplica? ¿Es exacto o heurístico?

2. Aplicación de los datos del enunciado al algoritmo indicado: Valor de la variable a optimizar con la solución obtenida.

3. Si la solución no es forzosamente óptima, dar (si es posible) una cota que permita tener una idea de la bondad de la solución encontrada.

A) Modelo: $7/2/G/F_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	17
	2	M_2	24
B	1	M_1	11
C	1	M_2	14
D	1	M_1	4
	2	M_2	18
E	1	M_2	11
	2	M_1	34
F	1	M_1	27
	2	M_2	9
G	1	M_2	16
	2	M_1	8

B) Modelo: $2/4/G/F_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	7
	2	M_2	4
	3	M_3	1
	4	M_4	4
	5	M_1	4
	6	M_2	8
B	1	M_2	11
	2	M_4	13
	3	M_1	7
	4	M_3	9

C) Modelo: $4/4/P/F_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	6
	2	M_2	7
	3	M_3	1
	4	M_4	3
B	1	M_1	5
	2	M_2	5
	3	M_3	11
	4	M_4	4
C	1	M_1	2
	2	M_2	4
	3	M_3	2
	4	M_4	9
D	1	M_1	10
	2	M_2	2
	3	M_3	8
	4	M_4	9

D) Modelo: $5/2/F/F_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	26
	2	M_2	31
B	1	M_1	11
	2	M_2	44
C	1	M_1	4
	2	M_2	58
D	1	M_1	41
	2	M_2	34
E	1	M_1	47
	2	M_2	19

E) Modelo: 4/3/F/Fmax

Pieza	Operación	Máquina	Duración
A	1	M_1	3
	2	M_2	2
	3	M_3	5
B	1	M_1	8
	2	M_2	4
	3	M_3	3
C	1	M_1	3
	2	M_2	2
	3	M_3	9
D	1	M_1	4
	2	M_2	4
	3	M_3	0

F) Modelo: $2/4/F/F_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	14
	2	M_2	9
	3	M_3	8
	4	M_4	11
B	1	M_1	4
	2	M_2	11
	3	M_3	9
	4	M_4	3

G) Modelo: $7/1/F/T_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	17
B	1	M_1	42
C	1	M_1	31
D	1	M_1	28
E	1	M_1	21
F	1	M_1	24
G	1	M_1	37

Las piezas A, B y C están comprometidas para antes del instante 90.

Las piezas D y E están comprometidas para antes del instante 105.

Las piezas F y G están comprometidas para antes del instante 167.

H) Modelo: $6/3/F/C_{max}$

Pieza	Operación	Máquina	Duración
A	1	M_1	3
	2	M_2	8
	3	M_3	9
B	1	M_1	6
	2	M_2	4
	3	M_3	1
C	1	M_1	7
	2	M_2	4
	3	M_3	2
D	1	M_1	10
	2	M_2	6
	3	M_3	7
E	1	M_2	2
	2	M_1	3
	3	M_3	2
F	1	M_1	2
	2	M_2	3
	3	M_3	2

Las máquinas están disponibles a partir de los instantes 0, 12 y 36. Las piezas llegan en los instantes 0 y 24 respectivamente.

Enunciado 6.3.13

Los datos de cierto problema son los siguientes:

10 máquinas, 10 piezas, 10 operaciones por pieza (una en cada máquina). Todas las máquinas están libres en el instante 0, todas las piezas pueden iniciar su elaboración en el instante 0.

a) Aplicar un método sistemático para encontrar una ordenación razonablemente buena (criterio:ocupación total del taller).

b) Dibujar un diagrama de GANTT del resultado anterior.

c) Deducir de dicho diagrama algunas permutaciones susceptibles de mejorar el valor obtenido.

d) Recalcular la ocupación del taller.

e) Reiterar c) y d) hasta agotar las posibilidades.

REFLEXIÓN: ¿Es interesante un procedimiento iterativo como el diseñado?

OP. = Operación
M. = Máquina
D. = Duración
P. = Pieza

	OP. 1		OP. 2		OP. 3		OP. 4		OP. 5		OP. 6		OP. 7		OP. 8		OP. 9		OP. 10	
P.	M.	D.	M.	D.	M.	D.	M.	D.	M.	D.	M.	D.	M.	D.	M.	D.	M.	D.	M.	D.
1	1	29	2	78	3	9	4	36	5	49	6	11	7	62	8	56	9	44	10	21
2	1	43	3	90	5	75	10	11	4	69	2	28	7	46	6	46	8	72	9	30
3	2	91	1	85	4	39	3	74	9	90	6	10	8	12	7	89	10	45	5	33
4	2	81	3	95	1	71	5	99	7	9	9	52	8	85	4	98	10	22	6	43
5	3	14	1	6	2	22	6	61	4	26	5	69	9	21	8	49	10	72	7	53
6	3	84	2	2	6	52	4	95	9	48	10	72	1	47	7	65	5	6	8	25
7	2	46	1	37	4	61	3	13	7	32	6	21	10	32	9	89	8	30	5	55
8	3	31	1	86	2	46	6	74	5	32	7	88	9	19	10	48	8	36	4	79
9	1	76	2	69	4	76	6	51	3	85	10	11	7	40	8	89	5	26	9	74
10	2	85	1	13	3	61	7	7	9	64	10	76	6	47	4	52	5	90	8	45

Enunciado 6.3.14

La empresa IRAQ (Industrial de Reactivos y Aditivos Químicos) fabrica cinco productos diferentes. Para obtenerlos hay que tratar las materias primas a lo largo de un proceso que comprende cuatro etapas, las cuales, por razones técnicas se han de llevar a cabo en cuatro reactores diferentes denominados convencionalmente R_1, R_2, R_3 y R_4. Cada reacción ocupará cada uno de los reactores durante un tiempo variable, –que expresado en horas– se indica en la *Tabla 1*. Durante este tiempo, el reactor está ocupado exclusivamente en la reacción, y puede producir cualquier cantidad hasta el máximo (expresado en kg de producto terminado) que se indica en la *Tabla 3*. Si en una operación no se carga completamente el reactor, éste hace el proceso igualmente, pero evidentemente con unos costes unitarios superiores. El tiempo de cargar y descargar el reactor se incluye en el tiempo indicado en la *Tabla 1*.

Después de fabricar cualquier producto y antes de pasar al siguiente, es necesario hacer una limpieza del reactor. Si en dos reacciones consecutivas de un reactor se hace el mismo producto, no hace falta limpiarlo, y por esta razón, con la finalidad de ahorrar tiempo de limpieza, siempre se hacen seguidas todas las operaciones de un mismo producto que se haya de tratar en un mismo reactor. Este tiempo de limpieza es el que se indica en la *Tabla 2*.

La planificación de la producción indica que los próximos días se deben fabricar las cantidades de cada producto indicadas en la *Tabla 3*, y el objetivo es terminar con toda la producción en el menor tiempo posible.

a) Encontrar dos soluciones heurísticas al problema si se añade la restricción adicional de que todos los reactores tratarán todos los productos en el mismo orden, y cuantificar su distancia máxima al óptimo.

b) Si no se considera la restricción adicional indicada en el apartado anterior, decir en qué lugares del proceso puede ser interesante considerar la posibilidad de cambiar el orden de los productos.

c) Para encontrar una solución en las condiciones más generales, seguir este proceso:

c.1: Encontrar la solución óptima considerando sólo R_1 y R_2.

c.2: En cada punto de los que se han mencionado en b), utilizar un método de dispatching para determinar el nuevo orden a seguir, teniendo en cuenta la situación creada por el orden indicado en los reactores anteriores.

¿Es óptima la solución hallada? ¿Por qué?

Tabla 1. *Duración del proceso de cada artículo en cada uno de los reactores*

Artículo	R_1	R_2	R_3	R_4
Arbax	3'00	1'75	5'50	0'75
Butilón	2'00	4'75	0'50	3'75
Criosamil	2'00	1'75	1'25	1'75
Difosforeno	3'00	7'75	5'50	2'75
Extramizol	1'00	8'75	4'50	6'75

Tabla 2. *Tiempo de limpieza posterior a cada artículo en una operación de cada uno de los reactores*

	R_1	R_2	R_3	R_4
Horas	1'00	0'25	0'50	0'25

Tabla 3. *Kilogramos máximos a producir de cada artículo en una operación de cada uno de los reactores*

Artículo	R_1	R_2	R_3	R_4
Arbax	300	1000	700	800
Butilón	100	200	270	450
Criosamil	500	1500	950	1600
Difosforeno	250	800	600	400
Extramizol	700	1950	1200	2250

Tabla 4. *Cantidades a producir (por artículo)*

Artículo	Kilogramos
Arbax	550
Butilón	100
Criosamil	1400
Difosforeno	250
Extramizol	650

Enunciado 6.3.15

Dados los siguientes tiempos de proceso de cuatro trabajos en tres máquinas, hallar la ordenación *óptima* para terminar lo antes posible con todos ellos:

trabajo	máquina 1	máquina 2	máquina 3
1	1	4	2
2	3	3	4
3	2	5	1
4	4	2	3

Enunciado 6.3.16

Preparar un programa y realizar un experimento por simulación relativo a la eficacia relativa de dos o tres heurísticas dinámicas (del tipo trapecios) en los problemas de *flow-shop*. El programa debe ser capaz de partir de unos valores de *n* (nº de piezas) y *m* (nº de máquinas) dados y generar un problema del taller mecánico *flow-shop* de tamaño (*n,m*) y tipo de tiempos especificado (por ejemplo con o sin duraciones nulas, distribución uniforme, unimodal, bimodal, etc.). A continuación para cada problema se realizará un cálculo de la cota inferior (valor que servirá para la normalización de los problemas). Se aplicarán las diversas heurísticas dinámicas y se evaluarán sus ocupaciones. A continuación se guardarán en memoria diversos datos:

- suma de las cotas
- suma de las ocupaciones según cada heurística
- suma de las mejores ocupaciones
- suma de las peoores ocupaciones
- nº de veces que una heurística ha dado un valor igual a la cota
- nº de veces que una heurística ha dado la mejor ocupación
- nº de veces que una heurística ha dado la peor ocupación
- indicadores binarios de las veces que una heurística ha dado un valor mejor que otra.

A continuación se podrá juzgar la eficacia relativa de las heurísticas para cada par de valores (*n,m*).

Puesto que para que los resultados sean significativos es preciso que el número de problemas generado sea alto (500 o 1000 por lo menos) será necesario que el programa esté compilado.

Enunciado 6.3.17

Los productos de la empresa BERENICE se fabrican tratando en unos moldes cierto material, a presión y temperatura establecidas. El material puede ser de diferentes colores; en consecuencia cada producto se define a partir del tipo de molde ("modelo") y el tipo de material ("color"). El sistema productivo se compone de una cadena, a cuyo inicio un robot coloca el molde y el material; a lo largo de dicha cadena se efectúan las operaciones,

y al final se retira el producto, regresando el molde a su almacén donde queda disponible para una nueva utilización. Para cada modelo pueden existir varios moldes idénticos ("réplicas").

Los datos productivos son:

número de modelos M = 20

número de colores C = 12

número de productos MxC = 240

número de réplicas por modelo NR(i) = 2 (inicialmente)

tiempo de recorrido de la cadena T = 30 minutos

ciclo de producción F = 1 producto cada 5 minutos

número de turnos en día laborable NT = 2

número de horas efectivas por ciclo HT = 7 horas

número de días laborables por semana S = 5

Los pedidos a producción tienen un número (k), una fecha (número de semana, $f(k)$), un índice de prioridad ($p(k)$), y para algunos productos i y colores j, la cantidad solicitada $q(i,j,k)$. La suma de cantidades solicitadas en todos los pedidos de una semana tiene un valor medio inferior a 420, pero oscila de una semana a otra. Inicialmente existe una cartera de pedidos a cumplimentar, de unas 800 unidades.

Se trata de desarrollar un programa que simule la programación y comportamiento de la cadena. El programa debería permitir:

1) Listar o presentar los pedidos de la cartera,

2) Introducir nuevos pedidos,

3) Cambiar los parámetros de la cadena,

4) Calcular una "nota" para cada producto-pedido (i,j,k), que resultará de la agregación de tres notas:

4.1. nota de pedido, teniendo en cuenta la antiguedad y el índice de prioridad,

4.2. nota de volumen de pedido, primando los pedidos grandes y pequeños,

4.3. nota de volumen de producto, función creciente del valor $q(i,j,k)$

5) Programar la realización de los productos en la cadena. Para ello se elegirá el producto que corresponda al producto-pedido con mayor nota, siempre que tenga réplicas disponibles. Cuando se lance un producto, se irá fabricando, dentro de la semana, toda la cantidad del mismo que esté pendiente, aunque corresponda a pedidos distintos,

6) Como consecuencia de lo anterior, se obtendrá para cada día el programa secuenciado para el robot de la cadena, y el número de pedidos cumplimentados.

Enunciado 6.3.18

Los productos de la empresa BERENICE se fabrican tratando en unos moldes cierto material, a presión y temperatura establecidas. El material puede ser de diferentes colores; en consecuencia cada producto se define a partir del tipo de molde ("modelo") y el tipo de material ("color"). El sistema productivo se compone de una cadena, a cuyo inicio un robot coloca el molde y el material; a lo largo de dicha cadena se efectúan las operaciones, y al final se retira el producto, regresando el molde a su almacén donde queda disponible para una nueva utilización. Para cada modelo pueden existir varios moldes idénticos ("réplicas").

Cada color resulta de la yuxtaposición de tres capas de material ("obleas"), D, E, C. Cada color tiene un terceto de obleas distinto de los demás pero puede tener alguna de las obleas común con otro(s). Los datos son:

número de colores C = 12,

número de obleas D, ND = 12

número de obleas E, NE = 6

número de obleas C, NC = 4

número máximo de productos a fabricar por día : 84

número máximo de colores a utilizar por día : 4

Los pedidos a producción tienen un número (k), una fecha (número de semana, $f(k)$), un índice de prioridad ($p(k)$), y, para algunos productos i y colores j, la cantidad solicitada $q(i,j,k)$. La suma de cantidades solicitadas en todos los pedidos de una semana tiene un valor medio inferior a 420, pero oscila de una semana a otra. Inicialmente existe una cartera de pedidos a cumplimentar, de unas 800 unidades.

PROBLEMA 1. Elegir los cuatro colores para cada día de una semana de 5 días, de forma que realizando el máximo de producción posible, se dé cierta prioridad a la antigüedad e índice de los pedidos. Indicar el número de obleas a consumir de cada tipo.

PROBLEMA 2. Dadas unas cantidades disponibles de obleas para un día determinado,

elegir los cuatro colores a fabricar, de forma que realizando el máximo de producción posible, se dé cierta prioridad a la antigüedad e índice de los pedidos.

El programa dispondrá de una base de colores, una base de obleas, la adscripción de las obleas a los colores y una base de pedidos. Los métodos a utilizar pueden ser exactos y/o heurísticos.

Enunciado 6.3.19

Para la resolución del problema de secuenciación n/m/F/F_{max} con todas las máquinas y piezas disponibles en el instante 0, Widmer y Hertz proponen un nuevo método. Sea $p_{j,i}$ la duración de la operación de la pieza *i* en la máquina *j*. Definen una "distancia" entre dos piezas *i* y *h* de la siguiente forma (aunque han trabajado con otras):

$$\delta(i,h) = p_{1,i} + \sum_{j=1}^{m} (m-j) \cdot \mid p_{j,i} - p_{j-1,h} \mid + p_{m,h}$$

y desarrollan el siguiente algoritmo:

Paso uno: buscar las piezas *i* y *h* tales que:

$$\delta(i,h) = \underset{x,y}{MIN} \{ \delta(x,y) \}$$

y establecer el camino i ————> h

Paso dos: WHILE queden piezas fuera del camino:

i) elegir una pieza al azar,

ii) situar la pieza en la posición que alargue lo menos posible el camino.

END WHILE

Se trata de construir la versión dinámica del algoritmo de Widmer y Hertz. Para ello debemos resolver la situación que se presenta cuando no todas las piezas están disponibles en el instante 0, sino que *i* lo está a partir de r_i. Probablemente siendo:

$$\beta(i,h) = MAX \{ rp_h - rp_i ; 0 \}$$

donde, llamando f_1 al instante en que la primera máquina está libre y fp_1:

$$fp_1 = MAX \{ f_1 \ ; \ MIN \{r_j\} \}$$

$$rp_i = MAX\{\ fp_1\ ;\ r_i\ \}$$

una generalización de la distancia podrá obtenerse restando β(i,h) en la expresión anterior, por ejemplo:

$$\delta(i,h) = p_{1,i} + m\cdot[\beta(h,i)-\beta(i,h)] + \sum_{j=1}^{m}(m-j)\cdot|p_{j,i}-p_{j-1,h}| + p_{m,h}$$

El algoritmo consistiría en lo siguiente:

1) Inicialmente $f_j = 0$ (o igual al instante de disponibilidad de las máquinas). Hacemos $j = 1$.

2) Estamos en la máquina j, r_i son los instantes de llegada a dicha máquina de las piezas, de acuerdo a la secuenciación en la máquina j-1. Colocamos en N todas las piezas,

3) Determinamos el conjunto E de las piezas programables:

$$ff_j = MIN\{\ rp_i + p_{j,i}\ \} \text{ para todo i } \epsilon \text{ N}$$

con rp_i y fp_j de acuerdo a lo anterior.

$$E = \{i \mid i \in N \quad y \quad rp_i < ff_j\}$$

4) Calculamos las distancias entre los elementos de E y hallamos el camino hamiltoniano más corto. Programamos la primera pieza i, con lo que actualizaremos f_j (y r_i). Eliminamos i de N (lo pasamos a P). Si N no es vacío volvemos a 3.

5) Incrementamos j en 1. Si no es la última máquina volvemos a 2.

6) La última máquina la programamos en el orden de llegada de las piezas (r_i creciente). FIN

Naturalmente lo anterior es mejorable.

Tratar 1000 problemas con dicho algoritmo.

Enunciado 6.3.20

Definir y establecer un algoritmo de cálculo para las cotas transversales de los problemas n/m/F/F_{max} cuando alguna(s) operacion(es) tiene(n) duración nula.

Enunciado 6.2.21

Para la resolución del problema de secuenciación n/m/F/F_{max} con todas las máquinas y piezas disponibles en el instante 0, Widmer y Hertz proponen un nuevo método. Sea $p_{j,i}$ la duración de la operación de la pieza *i* en la máquina *j*. Definen un "distancia" entre dos piezas *i* y *h* de la siguiente forma (aunque aseguran que han trabajado con otras):

$$\delta(i,h) = p_{1,i} + \sum_{j=1}^{m} (m-j) \cdot \mid p_{j,i} - p_{j-1,h} \mid + p_{m,h}$$

y aplican el siguiente algoritmo:

Paso uno: buscar las piezas *i* y *h* tales que:

$$\delta(i,h) = \underset{x,j}{MIN} \{ \delta(x,j) \}$$

y establecer el camino i ————> h

Paso dos: WHILE queden piezas fuera del camino:

i) elegir una pieza al azar,

ii) situar la pieza en la posición que alargue lo menos posible el camino.

END WHILE

Posibles variantes pueden consistir en buscar la secuencia de piezas que corresponda al camino hamiltoniano mínimo, bien mediante el algoritmo exacto de Little, bien mediante heurísticas *greedy* (utilizando los "ahorros").

Se trata de programar el algoritmo anterior, generar aleatoriamente colecciones de 1000 problemas, (cada una con valores distintos de *m* y *n*), calcular las cotas longitudinales y si es posible las transversales para cada problema, y analizar los resultados obtenidos con el algoritmo citado y otros métodos (PALMER, TRAPECIOS, etc.). Si tiene imaginación puede definir otro tipo de "distancia". Deberá redactar un informe en el que se describan (sucintamente) los métodos utilizados y los resultados numéricos obtenidos (calidad relativa media de las soluciones y tiempos de proceso). Suministrará con los programas, en lenguaje fuente, la colección de problemas utilizados en soporte magnético con formato adecuado.

Enunciado 6.3.22

Nos interesamos por el tratamiento del problema del taller mecánico (*flow-shop*) mediante simulación. El método a seguir es muy sencillo. En una fase cualquiera del algoritmo

estableceremos la secuencia en una máquina cualquiera. Sean:

f: instante en que la máquina está disponible,

N: conjunto de piezas no programadas todavía en la máquina,

r_i: instante de llegada de la pieza *i* a la máquina,

p_i: duración de la operación de la pieza *i* en la máquina,

como de costumbre:

$$rp_i = MAX\{f\ ,\ r_i\} \text{ a todo } i \in N$$

$$fp = \underset{i}{MIN}\{\ rp_i\ \}$$

$$ff = MIN\{\ rp_i + p_i\ \} \text{ a todo } i \in N$$

y el conjunto elegible será,

$$E = \{i \mid i \in N \text{ "and" } r_i < ff\}$$

La elección del elemento de E se realizará aleatoriamente (pudiendo atribuirse pesos diferentes a los mismos, de acuerdo a algún criterio: duración, tiempo muerto que provocan, índices de PALMER o TRAPECIOS, etc.). Dos máquinas son especiales. La primera, sobre todo si todas las piezas están disponibles en el instante 0, y la última, en la que no hay razón para cambiar el orden respecto a la penúltima, y por tanto no procede realizar ningún proceso especial. Atención especial merece el caso en que una pieza "salta" una máquina (hay que tenerlo en cuenta en el programa).

Se trata de preparar un programa informático capaz de realizar el proceso de simulación con ciertas variantes (por lo menos elección en la primera máquina del procedimiento, impuesto, sorteo, PALMER o TRAPECIOS; y en las demás sorteo no ponderado o bien ponderado en un par de formas). El programa leerá de un diskette el enunciado de un problema, determinará las cotas y la solución por PALMER y TRAPECIOS (simples), y si no ha obtenido una solución igual a la cota podrá pasar a la simulación. Se indicará el número de tentativas. Realizará los cálculos, guardando siempre la mejor solución obtenida. Abortará un cálculo cuando sea detectable que no puede obtenerse nada mejor de lo que ya se dispone. Terminará el proceso antes de llegar al límite si es detectable que ha alcanzado el óptimo. Usando adecuadamente un orden impuesto en la primera máquina puede tratarse el caso en el que las disponibilidades de las piezas no sean todas en el instante 0. Deberá suministrarse un programa que permita preparar el diskette de datos (siempre en lenguaje fuente) y un informe sobre los resultados.

Enunciado 6.3.23

Nos interesamos por el tratamiento del problema del taller mecánico (flujo general) mediante *dispatching* con simulación. El método a seguir es muy sencillo. En una fase cualquiera del algoritmo tendremos las operaciones clasificadas en P, E y N, estando las del subconjunto E clasificadas por máquinas E(j).

En una máquina *j* sean para las operaciones de E(j):

f_j : instante en que la máquina *j* está disponible,
p_i : duración de la operación de la pieza *i*,
r_i : instante de llegada de la pieza *i* a la máquina,

como de costumbre:

$$rp_i = MAX\{f_j , r_i\} \text{ a todo } i \in N$$

$$fp_j = \underset{i}{MIN}\{ rp_i \}$$

$$ff_j = MIN\{ rp_i + p_i \} \text{ a todo } i \in N$$

Se elige la máquina *j* con menor ff_j y el conjunto elegible será (programas activos):

$$E'(j) = \{i \mid i \in E(j) \text{ "and" } r_i < ff_j\}$$

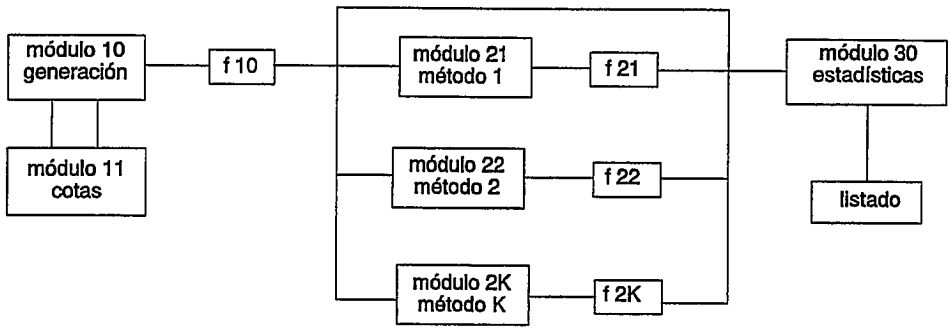
La elección del elemento de E'(j) se realizará aleatoriamente (pudiendo atribuirse pesos diferentes a los mismos, de acuerdo a algún criterio: duración, duración pendiente, tiempo muerto que provocan, número de operaciones restantes, etc.).

Se trata de preparar un programa informático capaz de realizar el proceso de simulación con ciertas variantes (por lo menos elección de un sorteo no ponderado o bien ponderado en un par de formas). El programa leerá de un diskette el enunciado de un problema, hallará (si es posible) una cota y una solución heurística simple, y si ésta no tiene un valor igual a la cota podrá pasar a la simulación. Se indicará el número de tentativas. Realizará los cálculos, guardando siempre la mejor solución obtenida. Abortará un cálculo cuando sea detectable que no puede obtenerse nada mejor de lo que ya se dispone. Terminará el proceso antes de llegar al límite si es detectable que ha alcanzado el óptimo. Deberá suministrarse un programa que permita preparar el diskette de datos (siempre en lenguaje fuente) y un informe sobre los resultados.

Enunciado 6.3.24 Evaluación de diferentes algoritmos de flow-shop

El trabajo pretende efectuar un estudio de la eficacia de diferentes algoritmos para la

resolución de problemas de flow-shop n/m/F/F_{max}. El esquema del trabajo completo es el siguiente:



Por la estructura de muchos de los métodos el problema tratado será esencialmente n/m/P/F_{max}.

Módulo 10: generación de problemas

Este módulo generará *N* problemas del tipo (*m,n*) donde *m* será el número de máquinas y *n* el número de piezas. Para cada problema determinará en forma aleatoria los valores *p(j,i)* de las duraciones de cada operación, que serán números enteros comprendidos entre 1 y p_{max} (inicialmente no se aceptan duraciones nulas). *N*, *m*, *n* y p_{max} serán parametrizables. Se utilizarán dos tipos de distribución aleatoria:

- la primera: uniforme

- la segunda: una trapezoidal con probabilidad creciente de 0'5 a $(p_{max}/3)+0'5$, constante de $(p_{max}/3)+0'5$ a $(2*p_{max}/3)+0'5$ y decreciente de $(2*p_{max}/3)+0'5$ a $p_{max}+0'5$.

El módulo describirá un fichero denominado FLP*α*mn.DAT, donde *α* será igual a U si la ley es la uniforme y T si es trapezoidal. *m* y *n* se escribirán en una posición (se adoptará la convención 10 = A, 11 = B, etc.), es decir si *m* = 4 y *n* = 8 con distribución uniforme, el fichero será FLPU48.DAT. Los cuatro primeros campos del fichero serán *m*, *n*, *α* y *N* y a continuación, para cada problema se escribirá su número de orden, la cota (hallada en el módulo 11) una letra indicando si la cota es longitudinal o transversal (L/T/X) y las *m*n* duraciones en la forma:

p(1,1) p(1,2) p(1,3) .... p(1,n) p(2,1) p(2,2) ..... p(m,n)

Módulo 11: cálculo de cotas

Será una subrutina que llamará el módulo 10 y determinará las cotas longitudinales y transversales, conservando la más exigente. El valor de la cota estará acompañado de una

L, si se ha obtenido como longitudinal, T como transversal, X ambas.

Módulos 2k en general

Dichos módulos pretenden aplicar unos métodos dados, en algún caso con diversas variantes, y escribir en un fichero (*f2k*) las soluciones halladas y el tiempo empleado. Como normas de caracter general se indican las siguientes:

- los métodos se describen en la literatura en líneas generales, pero para construir programas hay que descender a detalles, por ejemplo los relativos a los desempates producidos con los criterios básicos; dichos detalles deben quedar completamente definidos en la documentación que acompañe a los programas,

- en el tiempo de cada método se incluirá la evaluación de la solución,

- hay que tener cuidado al contabilizar los tiempos en los casos con diversas variantes, puesto que pueden existir partes comunes,

Cada variante estará identificada mediante un código de cuatro posiciones.

Ficheros f2k : soluciones de los métodos

El nombre del fichero será FLP*α*mnk.SOL, sus primeros campos serán *m*, *n*, *α*, *N*, *k*, *nv*(*k*) (donde *nv*(*k*) es el número de variantes) seguido de los códigos de las variantes. A continuación se indicará para cada problema:

n° de problema/solución variante 1/..../solución variante *nv*(*k*)

finalmente el tiempo de resolución de los *N* problemas según cada variante en segundos de "elapsed time"

Módulo 21: PALMER (2 variantes: PAL0, TDPC)

Primero aplicar PALMER, y a continuación trapecios dinámicos sobre la solución de PALMER (regla C).

Módulo 22: TRAPECIOS (2 variantes: TRA0, TDTC)

Primero aplicar TRAPECIOS, y a continuación trapecios dinámicos sobre la solución obtenida (regla C).

Módulo 23: CDS (1 variante: CDS0)

Método de Campbell, Dudek & Smith, se aplica el algoritmo de Johnson a *m*-1 problemas de dos máquinas ficticias (con $r = 1, 2, ..., m-1$):

problema r con duraciones $\sum_{j=1}^{r} p(j,i)$, $\sum_{j=r+1}^{m} p(j,i)$

Las secuencias obtenidas se aplican al problema real y se guarda la mejor solución.

Módulo 24: Dannebring (1 variante: DAN1)

Se aplica JOHNSON al problema de dos máquinas ficticias con duraciones:

$$\sum_{j=1}^{m} (m-j+1)\cdot p(j,i) \qquad \sum_{j=2}^{m} j\cdot p(j,i)$$

se calcula F_{max} y a continuación se intenta mejorar mediante la técnica RAES: se generan (n-1) soluciones permutando los pares de piezas adyacentes, se valoran y se toma la mejor; si es distinta de la inicial se aplica de nuevo el procedimiento y así sucesivamente.

Módulo 25: NEH (1 variante: NEH0)

Debido a Nawaz, Enscore & Ham. Se calcula:

$$S(i) = \sum_{j=1}^{m} p(j,i)$$

y se ordenan las piezas por orden decreciente de S(i). Se toman las dos primeras piezas y se evaluan las dos secuencias posible adoptando (definitivamente) la mejor. Se toma la siguiente pieza y se evalúan las tres posibles secuencias, adoptando la mejor; y así sucesivamente hasta haber situado las n piezas.

Módulo 26: PPE (1 variante: PPE0)

Se construyen m-1 problemas ficticios de dos máquinas con duraciones:

$$\sum_{j=1}^{r} (m-j+1)\cdot p(j,i) \qquad \sum_{j=1}^{r} (m-j+1)\cdot p(m-j+1,i)$$

$$\text{con } r = 1, 2, \ldots, m\text{-}1$$

se encuentra la secuencia óptima (mediante JOHNSON) para cada problema, y dichas secuencais se aplican al problema original, conservando la que da mejor F_{max}.

Módulo 30: estadísticas

A partir de los ficheros *f* 10 y los de las soluciones de los métodos (f2k) permitirá analizar

el comportamiento de éstos. Puesto que cada método puede tener variantes, y alguno de los ficheros no haber sido cumplimentado inicialmente el módulo pedirá los valores de *m*, *n*, *N* y *K*, y para cada método inicialmente previsto (de 1 a *K*) el número de soluciones (una por variante). Un 0 en el número de soluciones indicará que el fichero correspondiente no existe. El cálculo a realizar consiste en:

- determinar la suma de cotas, la suma de las duraciones obtenidas por cada método, las sumas de las duraciones mejor y peor para cada problema. Los valores se darán en forma absoluta (dividiendo por el número de problemas *N*) y relativa (haciendo 1 la suma de cotas).

- comparación de los métodos dos a dos (haciendo salvedad de los casos en que *todos* los métodos dan el mismo resultado. Se obtendrá una tabla en la que para cada par de variantes se indicará las veces que la primera ha dado mejor resultado que la segunda y viceversa.

- comparación con la mejor solución: se indicará el número de problemas para los que todos los métodos han dado la misma solución (y en su caso la coincidencia de dicho valor con la cota), y para cada variante el número del resto de problemas en que la misma ha dado el mejor valor y el peor valor (y en su caso el número de veces que la mejor solución coincide con la cota)

- relación de problemas en los que la mejor cota ha sido la longitudinal, la transversal o ambas; posible correlación con el mejor (y el peor) método,

- comparación de tiempos; correlación con la calidad de las soluciones, (las conclusiones pueden resultar engañosas al no utilizar el mismo ordenador para todos los métodos, ni ser la misma la habilidad de todos los programadores),

Se obtendrá un listado con las conclusiones.

Dimensiones

variable	valor normal	máximo
n	5 a 10	20
m	3 a 8	10
N	1000	10000
K	8	10
nv(k)	1 a 3	5
total variantes	10	20

Codificación

Los programas se realizarán en BASIC o QUICK-BASIC. Se ejecutarán complilados. Se entregarán en fuente y ejecutable.

Enunciado 6.3.25 Análisis de reglas para el problema 1-máquina

6.3.25.1 Propósito

Analizar las reglas de secuenciación en un sistema 1-máquina con tiempos de preparación dependientes de la pieza a realizar y de la pieza precedente, y con fecha prevista de entrega.

6.3.25.2 Formato

- el sistema productivo será capaz de realizar diez tipos de pieza (A, B, .., J),

- para cada pareja de tipos de pieza (*i,j*) existirá una tabla (en principio no simétrica), conocida por el usuario, que indicará el tiempo de preparación de la máquina TP(*i,j*), con TP(*i,i*) = 0

- para cada tipo de pieza existirá una tasa de fabricación, conocida por el usuario, RF(*i*),

- al inicio de una semana (80 horas) existirá una cartera de pedidos, con la siguiente estructura:

N° pedido/tipo pieza/cantidad/fecha comprometida

así mismo será conocida la última elaboración realizada,

- en la presentación se incluirá el tiempo de elaboración así como otros datos considerados convenientes para la decisión de secuenciación,

- el usuario elegirá el orden de realización de los pedidos,

- el sistema calculará la fecha en que los pedidos estarán realizados,

- el sistema calculará un índice de calidad de servicio y lo mostrará.

6.3.25.3 Realización

- el programa se realizará en alguna variante de BASIC (por ejemplo QUICK-BASIC) y se entregará la fuente, el compilado y el manual del usuario,

- se emplearán las posibilidades de la pantalla en color, aunque el programa será utilizable en bicolor,

- inicialmente se mostrará al usuario los datos y la cartera de pedidos, solicitándose la secuencia de realización,

- la cartera de pedidos (aleatoria, pero repetitiva a voluntad) deberá ser suficiente, por lo menos, para saturar la máquina durante dos semanas,

- podrá prolongarse la situación varias semanas, con adición (aleatoria) cada vez de los pedidos nuevos, substracción de los realizados e indicación de la situación de la máquina (si está realizando un pedido se supone que no quedará libre hasta haberlo culminado, pero el resto de secuencia es revisable),

- se desea una estructura que favorezca los aspectos lúdicos y competitivos.

Enunciado 6.3.26

La sección S de la empresa E, que cuenta con una máquina M_1 y otra M_2, fabrica un producto P que se obtiene por unión (en un tiempo despreciable) de dos componentes C_1 y C_2.

Para producir C_1 se requiere una operación que se puede hacer sólo con M1 y que dura 20 minutos. Para producir C_2 son necesarias tres operaciones (01, 02 y 03, precisamente en este orden). 01 se puede hacer sólo con M_2, en 18 minutos; 02, con M_1 y 15 minutos; 03, con M_2 y 10 minutos. Para pasar de una operación a otra hay un tiempo de preparación que es de 3 horas para M_1 y de 2 horas para M_2.

Se trata de establecer un procedimiento para determinar la máxima producción horaria y la forma de obtenerla (*a*: con horizonte ilimitado; *b*: con horizonte limitado) y aplicarlo a los datos indicados anteriormente; asimismo se trata de generalizar el procedimiento para valores cualesquiera del tiempo de operación y de preparación.