IX Congreso de Ingeniería de Organización
Gijón, 8 y 9 de septiembre de 2005

Planificación operacional de la capacidad en Logística de procesos con restricciones de tiempo

José Manuel García Sánchez, Jesús Racero Moreno, Gabriel Villa Caro

Escuela Superior de Ingenieros. Universidad de Sevilla. Camino de los Descubrimientos s/n, 41092 Sevilla. jmgs@esi.us.es

Palabras clave: Planificación operacional, Métodos exactos, Flujo en redes

1. Introducción

La logística de procesos sujeta a restricciones de tiempo aparece en sistemas donde las tareas tienen que ser completadas dentro de un intervalo de tiempo. En la mayoría de los trabajos sobre planificación y programación de tareas, el tiempo de finalización de las tareas no se encuentra restringido por ventanas temporales, permitiéndose cualquier instante del horizonte temporal como tiempo de finalización. Sin embargo, en algunos sistemas las tareas poseen restricciones de tiempo motivadas por aspectos físicos de los materiales involucrados en la fabricación, por estrictas normas de proceso o, simplemente, por la imposición de un intervalo de procesamiento. En estos casos la programación de las tareas depende principalmente de la capacidad del sistema, es decir, de la cantidad de recursos disponibles.

Los sistemas logísticos con restricciones de tiempo aparecen en la literatura con diferentes variaciones, dependiendo de los intervalos de tiempo para el procesamiento de las tareas, del objetivo definido, del numero de clases de recursos y de las características que motivan las diferentes clases.

Una primera clasificación de estos sistemas se produce según la amplitud del intervalo para el procesamiento de las tareas. Según ello se distingue entre:

- Problemas donde el tiempo de proceso de las tareas coincide con la amplitud del intervalo. Este grupo recibe el nombre de Programación de Trabajos Fijos (FSP).

- Problemas donde el tiempo de proceso de las tareas es menor que la amplitud del intervalo. Este tipo se denomina Programación de Trabajos Variables (VSP).

Con respecto a los objetivos, la optimización se orienta en dos direcciones:

- Planificación táctica de la capacidad: Se optimiza el uso de los recursos necesarios para atender todas las tareas.

- Planificación operacional de la capacidad: A partir de un conjunto de recursos, la planificación operacional es la asignación de recursos a tareas con objeto de maximizar el peso total de las tareas completadas.

De acuerdo al número de clases de recursos, los primeros trabajos sobre el problema consideraron una única clase de recurso, de forma que cada tarea podía ser procesada por

cualquier recurso. Sin embargo, el caso más interesante es considerar varias clases de recursos (Kroon et al, 1995). En este caso dos variantes se establecen:

- License Class Scheduling: Los recursos están disponibles durante todo el horizonte de planificación. La compatibilidad entre tareas y recursos está motivada por aspectos técnicos.
- Shift Class Scheduling: Cada recuro está solo disponible en un intervalo o conjunto de intervalos de tiempo. Un recurso puede procesar únicamente las tareas que se producen dentro de su intervalo de tiempo.

En este trabajo nos concentramos sobre la Planificación operacional con varias clases de recursos e intervalo de proceso coincidente con el tiempo de proceso del trabajo(problema FSP), analizando la complejidad del problema y proponiendo un *branch&bound* como alternativa a los métodos exactos ya existentes propuestos por Arkin y Silverberg(1987) y Brucker y Nordmann(1994).

2. Revisión bibliográfica

Con una única clase de máquinas y trabajos, FSP ha sido considerado por diversos autores entre los que destacan Arkin y Silverberg(1987), Bouzina y Emmons(1996), Garcia et al (2004) y Kroon et al(1985), los cuales muestran que el problema se resuelve en tiempo polinomial mediante un algoritmo de flujo a coste mínimo.

El caso con varias clases de máquinas ha sido considerado entre otros por Arkin y Silverberg(1987), Carter y Tovey (1991), mostrando su pertenencia a la clase NP de problemas. En Kolen y Kroon (1991,1993) se realiza un análisis de la complejidad del problema, distinguiéndose entre el tipo de compatibilidad entre máquinas y trabajos.

Para la resolución exacta del problema, Arkin y Silverberg(1987) y Brucker y Nordmann(1994) proponen métodos basados en la construcción de grafos que recogen todos los estados admisibles del problema. Mediante el cálculo de la ruta máxima en el grafo, se obtiene la solución óptima del problema. Sin embargo, estos métodos pueden únicamente ser útiles cuando el número de máquinas es pequeño.

Respecto a enfoques heurísticos, en Gabrel (1995) se modela el problema, sin considerar pesos para los trabajos, como un problema de máximo conjunto de nodos independientes y se proponen tres enfoques heurísticos para su resolución. Kroon et al (1995) describen un método basado en grafos que aprovecha el hecho de que FSP puede ser modelado como un problema de flujo a coste mínimo si todas las máquinas son de la misma clase. Los autores proponen también una cota superior del óptimo. Faigle y Nawijn(1995) presentan una heurística voraz que puede ser satisfactoriamente utilizada en una versión *online* del problema.

3. Formulación del problema

FSP es el problema de encontrar la programación óptima de una serie de trabajos no interrumpibles, denotados por $J = \{J_1 \ldots J_j \ldots J_n\}$, que tienen que ser procesados por un conjunto de máquinas en paralelo. Cada trabajo J_j posee un instante fijo de comienzo y finalización s_j y f_j respectivamente, y un peso o valor w_j. Se definen un conjunto de p clases de trabajos, $A=\{a_1 \ldots a_k \ldots a_p\}$. Cada trabajo J_j pertenece a una cierta clase de trabajo a_k del conjunto A. Existen también q diferentes clases de máquinas denotadas por el conjunto

$C=\{c_1 \ldots c_i \ldots c_q\}$. Sea m el número total de máquinas, cada una perteneciendo a una cierta clase c_i, y m_i el número total de máquinas de la clase c_i.

Cada trabajo perteneciente a la clase a_k puede únicamente ser llevado a cabo por máquinas que pertenezcan a un subconjunto de clases de máquinas. Para ello, se define una matriz L que representa la compatibilidad entre clases de máquinas y trabajos. L se define como sigue:

$$L_{pxq} = L(k, i) = \begin{cases} 1 & \text{si un trabajo de clase } a_k \text{ puede ser procesado por una maquina de la clase } c_i \\ 0 & \text{en otro caso} \end{cases}$$

Para considerar FSP como un problema de clase NP-duro, L tiene que satisfacer los siguientes características (Kolen y Kroon, 1991):

- Ser una matriz uno/cero irreducible
- Contener al menos dos columnas (dos clases de máquinas)

Para modelar el problema definimos las siguientes variables de decisión:

$$x_{ji} = \begin{cases} 1 & \text{si el trabajo } J_j \text{ se procesa por una maquina de clase } c_i \\ 0 & \text{en otro caso} \end{cases}$$

Siguiendo la notación definida, podemos formular el problema de maximizar el peso total de los trabajos procesados como sigue:

$$Max \sum_{j=1}^{n} w_j \sum_{i / J_j \in a_k \& L(k,i)=1} x_{ji}$$

s.a.

$$\sum_{i / J_j \in a_k \& L(k,i)=1} x_{ji} \leq 1 \quad \forall j / J_j \in J$$

$$\sum_{r / J_r \in a_k \& L(k,i)=1 \& s_r \leq t \& f_r > t} x_{ri} \leq m_i \quad \forall j / J_j \in J, t = s_j, \forall c_i \in C$$

$$x_{ji} \in \{0,1\}$$

El primer grupo de restricciones asegura que cada trabajo es asignado, a lo sumo, una vez a una máquina de una clase compatible. El segundo grupo garantiza que en el instante de inicio de cada trabajo, el número de trabajos asociados a máquinas de una cierta clase compatible no excede del número total de máquinas de esa clase.

4. Algoritmo basado en *Branch and bound*

En esta sección describimos el método exacto basado en un algoritmo de exploración dirigida. *Branch and Bound* es un método de exploración dirigida ampliamente utilizado para la resolución de problemas combinatorios.

4.1. Cotas y exploración

Como cota superior empleamos la relajación del problema. Para la resolución del problema se utilizaron las librerías de optimización XA.

El cálculo de la cota inferior está basado en un procedimiento constructivo que en cada iteración incorpora un par (J_j, c_i) a una solución parcialmente construida, hasta que no sea posible la introducción de ningún otro trabajo. El par (J_j, c_i) representa la asignación del trabajo J_j a una máquina de la clase c_i. En el procedimiento se hace uso del cálculo de flujo a coste mínimo en un conjunto de grafos que contemplan las restricciones existentes en el problema, así como la solución del problema obtenida del cálculo de la cota superior. El diseño de cada grafo se describe a continuación.
La exploración se realiza mediante un recorrido en anchura. Cada nodo de un nivel representa la asignación de cada variable x_{ji} a 1.

4.2. Diseño de los grafos del problema

Para la obtención de la cota inferior es necesaria la construcción de un conjunto de q grafos G_i, i=1...q (uno por cada clase de máquina), donde representaremos todos los trabajos que pueden ser procesados por máquinas de la clase c_i.
Para la construcción de cada grafo G_i se definen un conjunto de nodos $N_i = \{s_j, f_j / J_j \hat{I} a_k \,\&\, L(k, i) = 1\}$ donde cada nodo r representa un instante de comienzo o finalización de los trabajos que pueden ser procesados por máquinas de c_i. Es decir, $\{r / r = 1...R_j\} = \{s_i, f_i, L(a_j,c_i)=1\}$. El conjunto de nodos aparece ordenado cronológicamente. Cada trabajo $j_i \in N_i$ tiene asociado un arco desde el nodo que representa el instante de comienzo s_i hasta el nodo que representa su instante de finalización f_i. El coste de dicho arco es $-w_j$ y posee una unidad de capacidad. Además de estos arcos, existe un arco de coste cero y capacidad ilimitada desde cada nodo r a r+1, r = 1...R_j-1.
Por otra parte, consideramos aporte y demanda de flujo en el primer y último nodo, respectivamente, de m_i unidades (donde m_i es el número de máquinas de la clase c_i).
La solución optima del problema de flujo a coste mínimo sobre G_i provee de un conjunto de trabajos con peso total máximo sobre las máquinas de la clase c_i. Sin embargo, ello no garantiza optimalidad para el problema general, puesto que otra decision del problema es decidir con que clase de máquina procesar cada trabajo.
La Figura 2 recoge una ilustración de la construcción de un grafo a partir del problema descrito en la Tabla 1 y Figura 1.

Tabla 1. Datos del problema

Dimensiones: n=4; m=2; p=3; q=2

n	s_j	f_j	a_j	w_j	m	c_j
J_1	1	5	1	1	M_1	1
J_2	2	10	2	1	M_2	2
J_3	3	6	3	1		
J_4	7	10	1	1		

$$L = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

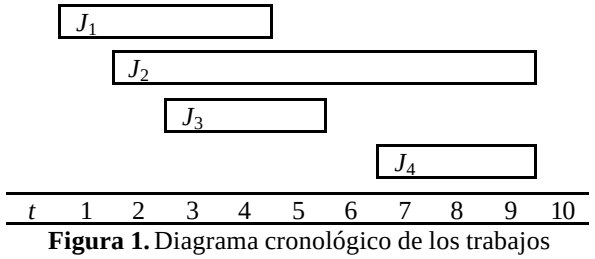
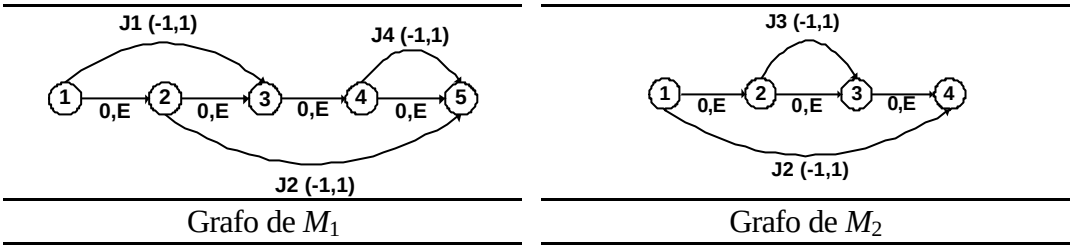


Figura 1. Diagrama cronológico de los trabajos



Grafo de M_1 | Grafo de M_2

Figura 2. Grafos de M_1 y M_2 (E = un valor muy grande; $\xrightarrow{(\text{coste}\text{ç}\text{apacidad})}$)

Sobre esos grafos, se resolverán problemas de flujo a coste mínimo, inyectando siempre una unidad de flujo desde el nodo inicial hasta el nodo final. La razón de una unidad se debe al hecho de que una máquina puede procesar sólo un trabajo en un mismo instante de tiempo. Los trabajos seleccionados serán aquellos por cuyo arco circule flujo.

4.3 Descripción del proceso constructivo

La fase constructiva es un procedimiento iterativo que incorpora un trabajo a la solución en cada paso, hasta que no sea admisible la incorporación de ningún trabajo más. Inicialmente, se establece una solución inicial según la solución obtenida para el problema relajado. Dicha solución establece un valor para las variables x_{ji} de forma que $0 \leq x_{ji} \leq 1$. Como solución inicial del proceso constructivo consideramos la asignación de pares (x_j,c_i) que correspondan con las variables $x_{ji}=1$. A partir de ahí, en cada iteración se distinguen dos conjuntos, S y S'. S está formado por aquellos pares ya asignados, esto es, $S=\{(J_j,c_i)\}$ con $L(a_j,c_i)=1$, mientras que S' contiene los trabajos no seleccionados. Por tanto $S \cup S'=N$ en cualquier iteración del proceso.

Dada una iteración k, se calcula una función de beneficio B_{ji} por procesar cada trabajo $J_i \in S'$ en una máquina de c_i compatible con J_j. El cálculo se realiza mediante el siguiente proceso:

Dados $J_j \in S'$ y $L(a_j,c_i)=1$, B_{ji} se obtiene según el siguiente proceso:

1. Cálculo del flujo a coste mínimo en el grafo G_i, facilitando el procesamiento de J_j. Este se consigue modificando el grafo M_j del siguiente modo:
   Respecto a los trabajos J_s, tal que $(J_s,c_i) \in S$, se demanda una unidad de flujo en su nodo origen, unidad que se inyecta en su nodo destino. La demanda de flujo aparece reflejado en el grafo de Figura 3 con signo positivo, mientras que inyectar flujo se representa con signo negativo. Con ello se obliga a que el cálculo del flujo a coste mínimo tenga obligatoriamente que contar con el procesamiento de estos pedidos.

   Respecto al trabajo J_i, se le supone un coste de $-E$, con E un numero grande. De ese modo, siempre pasará una unidad de flujo por el arco de J_i, si las condiciones del problema lo permiten. Si el arco del trabajo J_i no es seleccionado en el calculo de flujo a coste mínimo, se descarta el calculo del beneficio B_{ij}, puesto que no es admisible la selección del trabajo J_i en la máquina M_j.

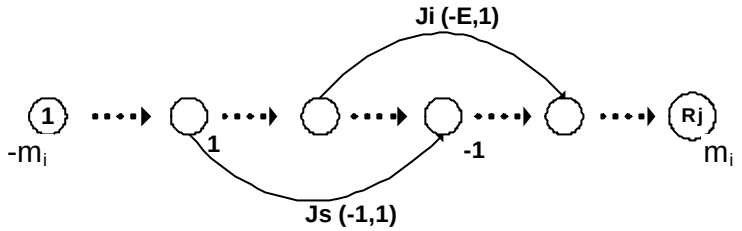


Figura 2. Modificación del grafo G_i

2. Cálculo de los trabajos a procesar en el resto de clases de máquinas.
Para el cálculo de los trabajos a realizar en el resto de máquinas, se asigna capacidad cero a los arcos de los trabajos seleccionados en el paso 1, en el resto de grafos del problema. Del mismo modo que en el apartado anterior, se asigna una unidad de flujo, como demanda e inyección de flujo, en los nodos de los trabajos pertenecientes a S asignados previamente a la clase del grafo a resolver.

   De forma iterativa y comenzando desde la primera a la última clase, exceptuando la clase c_i, se procesa cada clase mediante el cálculo de un problema de flujo a coste mínimo, y se va asignando capacidad cero, en el resto de máquinas por procesar, a los trabajos que vayan siendo seleccionados.

La asignación del trabajo J_j a c_i con mayor beneficio B_{ji} es seleccionada para pertenecer a S. En caso de existir empates, se selecciona aleatoriamente una asignación de entre las mejores. Una vez actualizado S, se repite el proceso iterativo hasta que se descarte el cálculo de B_{ji} para todo $J_j \in S'$ y c_i.

El valor de la solución construida se obtiene mediante la suma de pesos de los trabajos seleccionados.

5. Resultados computacionales

La primera fase del estudio computacional consistió en la construcción de una batería de problemas. A partir de ahí, compararemos la eficiencia del método exacto con los resultados obtenidos usando el método exacto propuesto por Arkin y Silverberg(1987). Obviamos la comparación con el método propuesto por Brucker y Nordmann(1994) puesto que muestra peor comportamiento que el método de Arkin y Silverberg.

5.1. Generación de problemas

Los problemas generados para evaluar el algoritmo fueron obtenidos aleatoriamente a partir de una modificación en el ratio a priori de utilización r diseñado por Kroon et al(1995) para la generación de problemas FSP. El ratio r es un indicador de la carga de trabajo esperada por unidad de capacidad en el espacio de trabajo. Nuestro ratio es definido como sigue:

$$r = \frac{\text{carga de trabajo esperada (uds. tiempo)}}{\text{espacio total de trabajo (uds. tiempo)}} = \frac{n \times \frac{1}{15} D}{T}$$

donde D indica la duración máxima de un trabajo, T representa la longitud del horizonte de planificación y n denota el número de trabajos. En este nuevo ratio, la carga de trabajo es independiente del número de máquinas. Por ello, el promedio de trabajos procesados depende del número de máquinas utilizadas. Tres valores para el ratio de utilización han sido usados: baja utilización (ρ=1.5), utilización media (ρ =3) y alta utilización (ρ =6). La tabla 2 recoge el número de trabajos procesados. Ello da una idea del grado de solapamiento de los trabajos en función del ratio r.

Tabla 2. Número de trabajos procesados

n	m	r = 1.5	r = 3	r = 6
50	4	38,7	26,8	16,8
100	4	73,5	53,6	38,4
100	8	82,8	60,6	39,7
200	4	146,6	105,6	75
200	8	168,8	117,9	83,5
200	16	179,1	124,5	84
400	4	246,5	204,4	143,2
400	8	324,8	233,5	160,5
400	16	365,6	255,6	175,6

Consideramos un horizonte de planificación de T=1000 unidades de tiempo e instancias con n=50, n=100, n=200 y n=400 trabajos. El parámetro D es calculado de acuerdo a los valores asignados en la fórmula del ratio.

El número de máquinas consideradas fue de m=4, m=8 y m=16. Respecto al número de clases de máquinas y trabajos, cuatro diferentes matrices L irreducibles fueron usadas en los experimentos, con dimensiones L_{2x3}, L_{3x4} y L_{4x4}.

La Tabla 3 presenta las combinaciones (n, m, L) consideradas para cada valor de r en la generación de problemas.

Tabla 3. Combinaciones (n, m, L) consideradas

	m=4	m=8	m=16	
	L_{2x3}	L_{3x4}	L_{3x4}	L_{4x4}
n=50	1	0	0	0
n=100	1	1	0	0
n=200	1	1	1	1
n=400	1	1	1	1

El número de máquinas pertenecientes a cada clase de máquinas se determina uniformemente respecto a m. Cada trabajo es aleatoriamente asignado a una clase de trabajo.

Para cada trabajo J_j el tiempo de proceso t_j se obtiene aleatoriamente en el intervalo (0,D), y el instante de inicio s_j se genera aleatoriamente en el intervalo (0, T-t_j). El peso w_j se obtuvo de forma aleatoria en el intervalo (0,100). Para cada combinación del ratio r respecto a las combinaciones de la Tabla 3 se generan 10 instancias, lo cual produce un total de 330 problemas.

5.2. Resumen de resultados

Los resultados que muestra la Tabla 4 ponen de manifiesto la gran eficiencia del método exacto propuesto en este trabajo. Los métodos exactos clásicos basados en grafos resultaban

inoperantes para un número de máquinas superior a 4. Sin embargo, nuestro método basado en la relajación del problema, presenta unos resultados que, a falta de un más amplio estudio computacional que explore otras alternativas en la construcción de problemas, permiten descartar el uso de métodos aproximados para el problema FSP, sea cual sea el número de máquinas y el número de trabajos. El estudio experimental fue llevado a cabo en un Intel Pentium IV 1.6Ghz.

La rapidez del método se debe a que en ningún caso se llega a explorar más de un nivel del árbol de exploración. Los tiempos que aparecen en la Tabla 4 vienen dados en CPU-segundos. En todos los casos se ha tomado promedio sobre el conjunto de problemas de cada tipo. Las celdas en blanco ("-") del método de Arkin y Silverberg se deben a que se descarto la resolución de ese grupo de problemas porque el tiempo excedía de 24 horas o bien se producían errores de sobre carga de memoria.

Tabla 4. Tiempos de ejecución

r	*n*	*m*	*Arkin y Silverberg*	*Branch&Bound*
1.5	50	4	13.7	1.6
	100	4	186.8	2
		8	-	2.4
	200	4	2945.2	2
		8	-	3.2
		16	-	3.6
	400	4	9654.3	3.2
		8	-	4.8
		16	-	5.2
3	50	4	325.5	0.8
	100	4	3169.8	0.9
		8	-	1.2
	200	4	-	1.6
		8	-	2
		16	-	2.8
	400	4	-	2
		8	-	3.6
		16	-	8.4
6	50	4	7750.0	0.8
	100	4	51676.0	1.2
		8	-	1.2
	200	4	-	1.2
		8	-	2.4
		16	-	3.2
	400	4	-	2.4
		8	-	6.4
		16	-	8

6. Conclusiones

En este trabajo hemos estudiado un caso particular dentro de la planificación operacional de la capacidad en sistemas con restricciones de tiempo. En concreto, el correspondiente al problema *Fixed Job Scheduling Problem*(FSP) con varias clases de trabajos y máquinas. Para el problema se ha diseñado un método exacto basado en exploración dirigida y en la resolución del problema FSP con una única clase de máquina. Los resultados computacionales han puesto de manifiesto la viabilidad del método para cualquier tamaño de

problema, e incluso permiten descartar el uso de algoritmos heurísticos para resolver este tipo de problemas.

Referencias

Arkin, E.M., Silverberg, E.L. (1987). Scheduling jobs with fixed starting and finishing times. *Discrete Applied Mathematics*, Vol. 18, pp. 1-8.

Bouzina, K.I., Emmons, H.(1996). Interval scheduling on identical machines. *Journal of Global Optimization*, Vol. 9, pp. 379-393.

Brucker, P., Nordmann, L. (1994). The k-Track Assignment Problem. *Computing*, Vol. 52, pp. 97-122.

Carter, M.W., Tovey, C.A.(1991). When is the classroom assignment problem hard?. *Operation Research*, Vol. 40, pp. 28-39.

Faigle, U., Nawijn, W.M.(1995). Note on scheduling intervals on-line. *Discrete Applied Mathematics*, Vol. 58, pp. 13-17.

Gabrel, V.(1995). Scheduling job within time windows on identical parallel machines: New model and algorithms. *European Journal of Operational Research*, Vol. 83, pp. 320-329.

Kolen, A.W.J., Kroon, L.G.(1991). On the computational complexity of (maximum) class scheduling. *European Journal of Operational Research*, Vol. 54, pp. 23-38.

Kolen, A.W.J., Kroon, L.G. (1993). On the computational complexity of (Maximum) Shift Class Scheduling. *European Journal of Operational Research*, Vol. 64, pp. 138-151.

Kroon, L.G., Salomon, M., Van Wassenhove L. N. (1995). Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research* , Vol. 82, pp. 190-205.