VIII Congreso de Ingeniería de Organización
Leganés, 9 y 10 de septiembre de 2004

Una heurística para la asignación de máquinas a trabajos fijos

José Manuel García Sánchez, Marcos Calle Suárez, Gabriel Villa Caro

Dpto Organización Industrial y Gestión de Empresas. Escuela Superior de Ingenieros. Camino de los Descubrimientos, s/n 48091 Sevilla. jmgs@esi.us.es, mcalle@esi.us.es, gvilla@esi.us.es

Resumen

En este trabajo se estudia el problema de la asignación de recursos a actividades fijas, conocido como Fixed Job Scheduling Problem (FSP). FSP se caracteriza como el problema de planificar, sobre un conjunto de máquinas en paralelo, un conjunto de trabajos no interrumpibles, caracterizados cada uno de ellos por un instante fijo de comienzo y finalización. El objetivo considerado en el problema es maximizar el numero de trabajos procesados con un número fijo de máquinas, asumiendo que existen diferentes clases de máquinas y trabajos. Para la resolución del problema se propone una aproximación heurística formada por un procedimiento constructivo y fase de mejora local. La calidad del mismo se compara con otros métodos heurísticos existentes en la bibliografía, así como con un método exacto. Los resultados computacionales ponen de manifiesto que el método encuentra soluciones de mejor calidad que esos otros métodos con tiempos de computación aceptables.

Palabras clave: Heurística, Trabajos fijos.

1. Introducción

La teoría de la programación de trabajos en intervalos fijos sobre máquinas en paralelo abarca problemas de dos tipos. Por un lado, problemas donde el intervalo para el proceso de cada trabajo coincide con el tiempo de duración del trabajo, y por otro lado, problemas donde el intervalo de tiempo para el procesamiento de cada trabajo es mayor al tiempo de proceso del trabajo. El primer tipo de problemas es denominado en la literatura como *Fixed Job Scheduling Problem* (FSP) mientras que el segundo se conoce como *Variable Job Scheduling Problem* (VSP).

En este trabajo abordamos el estudio del problema FSP, el cual presenta las siguientes restricciones:

1) Cada máquina puede realizar un solo trabajo al mismo tiempo.

2) Cada trabajo puede ser completado sobre un subconjunto de máquinas. Se asume que existe un conjunto de clases de máquinas y otro de clases de trabajos, de forma que cada clase de máquina puede realizar trabajos de un número limitado de clases de trabajos.

El objetivo a considerar en el problema corresponde con maximizar el número total de los trabajos procesados con un número fijo de máquinas.

Para la resolución del problema se presentará una heurística constructiva que incorpora un procedimiento de mejora basado en grafos. El método heurístico será comparado con otras

heurísticas definidas en Gabrel (1995) y Kroon *et al* (1995) para el mismo problema y a partir de las soluciones óptimas obtenidas con el método propuesto en Arkin y Silverberg (1987). Para dicha labor se generará aleatoriamente una batería de problemas a partir de los parámetros definidos en Kroon *et al* (1995) para la generación de instancias del problema FSP.

2. El Problema FSP

La asignación de máquinas a trabajos fijos se define como el problema de programar una serie de trabajos sobre un conjunto de máquinas en paralelo. Cada trabajo posee un instante fijo de comienzo, un instante fijo de finalización, un peso o valor y una clase de trabajo. Respecto a las máquinas, puede considerarse un coste asociado al uso de cada máquina y, en ocasiones, un intervalo de tiempo para cada una de ellas en el que únicamente están disponibles. De forma genérica, este problema es denominado en la literatura como *Fixed Job Scheduling Problem* (FSP).

La clasificación del problema FSP se realiza, principalmente, en torno a los siguientes parámetros:

1) Objetivo del problema
2) Número de clases de máquinas y trabajos

En el problema se consideran principalmente dos objetivos, minimizar los costes totales asociados al uso de las máquinas realizando todos los trabajos (minimizar el número de máquinas requeridas para procesar todos los trabajos, en el caso de no considerar costes) y, a partir de un número fijo de máquinas, maximizar el valor total de los trabajos completados. En nuestro caso planteamos el segundo de los objetivos, suponiendo el mismo valor para todos los trabajos, por lo que el objetivo se convierte en maximizar la suma de los trabajos procesados.

Respecto al número de clases de máquinas y trabajos, se distingue entre una única clase de máquina y trabajo y varias clases. El primero de ellos es el caso más sencillo del problema FSP. En este escenario cada trabajo puede ser procesado por cualquier máquina. En el segundo caso, cada trabajo puede únicamente ser procesado por un subconjunto de máquinas (Kroon *et al*, 1995). En este segundo caso se conjungan dos variantes respecto al intervalo de disponibilidad de las máquinas. La primera, denominada *Shitf Class Design* (SCD) (Kolen y Kroon, 1993), establece uno intervalo(*shifts*) en el que se encuentra disponible cada máquina. Cada máquina puede procesar cualquier trabajo que se procese dentro del intervalo en el que está disponible. En la segunda variante, llamada *License Class Design* (LCD) (Kolen y Kroon, 1992), las máquinas están disponibles durante todo el horizonte temporal. La compatibilidad entre clases de trabajos y máquinas se rige por motivos técnicos.

Otros parámetros que distinguen también algunos de los problemas son el valor del pedido y la propiedad de interrumpir el procesamiento del trabajo en una máquina (*preemption*). En el primer caso se distingue, por un lado, la asignación de un valor que mide el beneficio por completar el trabajo y que puede ser diferente para cada pedido, y por otro, que todos los trabajos posean el mismo valor, o lo que es lo mismo, el valor de cada trabajo es la unidad.

En el caso de *preemption*, el formato más general del problema asume la prohibición de interrumpir el trabajo por la propia filosofía de los mismos. Sin embargo, en algún escenario

del problema se ha considerado admisible que el trabajo puede ser procesado por más de una máquina.

Problemas prácticos que implican la resolución de instancias del problema FSP aparecen en múltiples áreas pertenecientes a la optimización de recursos. Las más importantes que han sido estudiadas han sido: Procesos de mantenimiento de aviones en aeropuertos, asignación de puertas a vuelos en aeropuertos, control del tráfico aéreo, planificación de salas de operaciones en hospitales, asignación de conductores en líneas de autobuses y planificación de la captura de imágenes desde satélites.

3. Notación

El conjunto de trabajos a programar se representa por $N=\{J_1,..,J_i, ...J_n\}$ donde cada trabajo J_i se describe con un instante de comienzo s_i, un instante de finalización f_i y una clase de trabajo a_i. $M=\{M_1,.., M_j,...,M_m\}$ representa las máquinas del sistema, donde cada máquina M_j se distingue por pertenecer a una clase de máquina c_j.

Para el problema se definen A diferentes clases de trabajos y C clases de máquinas. Se definen también una matriz de compatibilidad L entre clases de máquinas y trabajos cuyos valores pueden ser:

$$L_{A\times C} = L(fila, col) = \begin{cases} 1 & \text{Trabajos } J_i \text{ con } a_i = fila, \text{ son compatibles con maquinas } M_j \text{ si } c_j = col \\ 0 & \text{en otro caso} \end{cases}$$

4. Método heurístico para la resolución del problema

El método propuesto para la resolución del problema FSP consta en primer lugar de un procedimiento constructivo que genera una solución admisible del problema y sobre la que se realiza, en segundo lugar, un procedimiento de mejora local.

La fase constructiva del método incorpora, en cada iteración del proceso, un trabajo a la solución que está siendo construida, hasta que no sea posible la introducción de ningún otro trabajo. La fase de mejora extrae cada uno de los trabajos de la solución e intenta incorporar otros que no pertenecen a la misma, con objeto de mejorar el número total de trabajos en la solución. Tanto la primera fase, como la segunda, hacen uso del cálculo de flujo a coste mínimo en un conjunto de grafos que contemplan las restricciones existentes en el problema. El diseño de cada grafo y cada una de las fases del método se describen a continuación.

4.1. Diseño de los grafos del problema

Para el cálculo de soluciones admisibles se construye un conjunto de m grafos $G_j(N_j, A_j)$, j=1...m, en el que se recogen los trabajos que puede procesar cada máquina M_j.

Para la construcción del grafo correspondiente a la máquina M_j (grafo G_j) se definen un conjunto de nodos $\{r / r = 1...R_j\}$ donde cada nodo r representa un instante de comienzo o finalización de los trabajos que pueden ser procesados por la maquina M_j. Es decir, $\{r / r = 1...R_j\} = \{s_i, f_i, L(a_i,c_j)=1\}$. El conjunto de nodos aparece ordenado cronológicamente. Cada trabajo J_i compatible con M_j 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 –1 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.

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; A=3; C=2

N	s_i	f_i	a_i
J_1	1	5	1
J_2	2	10	2
J_3	3	6	3
J_4	7	10	1

M	c_j
M_1	1
M_2	2

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

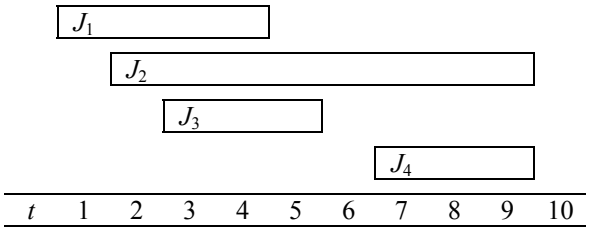

Figura 1. Diagrama cronológico de los trabajos

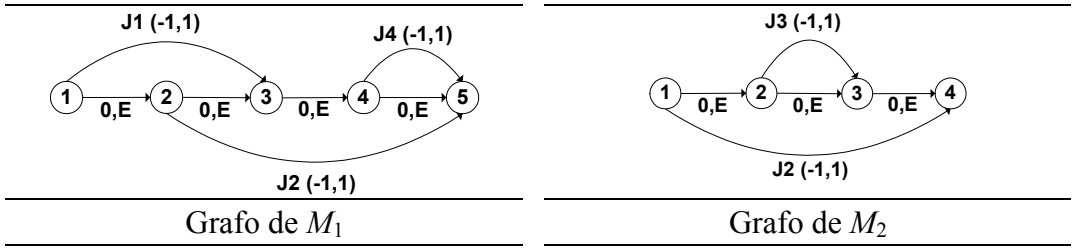

Figura 2. Grafos de M_1 y M_2 (E = un valor muy grande; $\xrightarrow{(coste,capacidad)}$)

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.

4.2. Fase constructiva

La fase constructiva de la heurística 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.

En cada iteración se distinguen dos conjuntos, S y S'. S está formado por aquellos trabajos introducidos en la solución junto con la máquina que lo procesa, esto es, S={(J_i,M_j)} con $L(a_i,c_j)$=1, mientras que S' contiene los trabajos que no seleccionados. Por tanto $S \cup S' = N$ en cualquier iteración del proceso. Inicialmente, $S=\varnothing$ y $S'=N$.

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

Dados $J_i \in S'$ y $L(a_i,c_j)=1$, B_{ij} se obtiene según el siguiente proceso:

1. Cálculo del flujo a coste mínimo en el grafo M_j, facilitando el procesamiento de J_i.

   Este se consigue modificando el grafo M_j del siguiente modo:

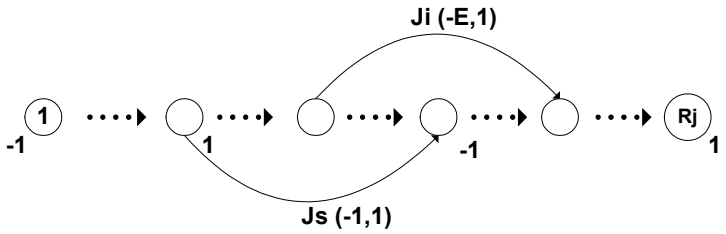

Figura 3. Modificación del grafo M_j

   Respecto a los trabajos J_s, tal que $(J_s,M_j) \in S$, 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 2 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.

2. Cálculo de los trabajos a procesar en el resto 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, procesados en cada máquina.

   De forma iterativa y comenzando desde la primera a la última máquina, exceptuando la máquina M_j, se procesa cada máquina 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_i a M_j con mayor beneficio B_{ij} 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_{ij} para todo $J_i \in S'$ y M_j.

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

4.3 Fase de mejora

La fase de mejora explora un espacio de soluciones vecinas que corresponde con el número de trabajos pertenecientes a la solución. Por cada trabajo J_i perteneciente a la solución construida S, se define una solución vecina calculada del siguiente modo:

Dado el trabajo J_i tal que $(J_i, M_j) \in S$, se asigna capacidad cero al arco de J_i en el grafo M_j y se procede de forma semejante al cálculo del beneficio B_{ij} de la fase constructiva. Por tanto, se resuelve el problema de flujo a coste mínimo en el grafo M_j, se descartan los trabajos seleccionados en el resto de grafos y se continúa iterativamente hasta resolver todos los grafos.

El proceso de mejora continúa hasta que todas las soluciones vecinas no mejoren a la solución actual.

5. Resultados Computacionales

La primera etapa de los experimentos computacionales comprendió la construcción de una batería de problemas (sección 5.1). En la sección 5.2 se presentan los resultados obtenidos de la aplicación del procedimiento heurístico a dicha batería. En estos resultados se incluye una comparación con los procedimientos heurísticos propuestos en Kroon *et al* (1995) y Gabrel(1995), así como con la soluciones óptimas, para algunas de las instancias, obtenidas con el método exacto definido en Arkin y Silverberg(1987) para el problema FSP.

5.1 Generación aleatoria de problemas

Las instancias que se crearon para probar los diferentes métodos heurísticos fueron generadas aleatoriamente a partir de un ratio de utilización a priori ρ diseñado en *Kroon* et al (1995) para el problema FSP.

El ratio ρ del sistema es un indicador de la carga de trabajo por unidad de capacidad disponible, y se define como sigue:

$$\rho = \frac{\text{carga total de trabajo esperada (en unidades de tiempo)}}{\text{capacidad total (en unidades de tiempo)}} = \frac{n \times \frac{1}{2} D}{T \times m}$$

donde:
- D indica la duración máxima de un trabajo
- T representa la longitud del horizonte temporal
- n denota el número de trabajos
- m es el número de máquinas

Se utilizan tres valores del ratio de utilización:
- Ratio de utilización baja ($\rho = 0.8$)
- Ratio de utilización media($\rho = 1$)
- Ratio de utilización alta ($\rho = 1.2$)

Consideramos un horizonte temporal de 1000 unidades de tiempo e instancias con n = 50, 75, 100 y 200 trabajos. El número de máquinas fue de m = 4, 6 y 8. Se consideraron A=3 clases de trabajos y C = 2 (con m = 4), C = 3 (m = 6), y C = 4 (m = 8) clases de máquinas.

El número de máquinas pertenecientes a cada clase se distribuye uniformemente sobre el número de máquinas del problema. Cada trabajo se asigna de forma aleatoria a una clase de trabajo y la matriz L se determina aleatoriamente, teniendo en cuenta que cada clase de máquina debe ser compatible con dos clases de trabajos, y siempre debe existir alguna clase de máquina compatible con cada clase de trabajo.

El parámetro D se obtuvo a partir de cada valor especificado por el ratio ρ de utilización. El tiempo de proceso t_i de cada trabajo J_i se generó aleatoriamente de la distribución uniforme (0,D), y el instante de comienzo s_i se generó aleatoriamente de igual forma en el intervalo (0, T-t_i). Para cada combinación (ρ, n, C) se generaron 10 instancias, lo que produce un total de 360 instancias. Todos los valores que aparecen en los resultados van referidos al promedio sobre las 10 instancias generadas para cada combinación.

5.2. Resumen de Resultados

Para cada método heurístico se muestra el promedio sobre el número de trabajos realizados y el tiempo de ejecución, dado en segundos de CPU. Respecto a Gabrel (1995), se muestra el mejor de los resultados obtenidos con los tres procedimientos heurísticos que se presentan en ese trabajo para el problema. La heurística constructiva se representa como *HC*.

Tabla 2. Resumen de resultados respecto a ρ y n

		Kroon		*Gabrel*		*HC*	
ρ	n	*N_Trbs*	*Tiempo*	*N_Trbs*	*Tiempo*	*N_Trbs*	*Tiempo*
0.8	50	37,6	5,3	37,9	**0,2**	**39,0**	0,7
	75	56,9	5,7	57,2	**1,7**	**58,5**	2,2
	100	76,1	6,0	76,8	**2,7**	**77,6**	3,5
	200	152,2	**7,1**	154,1	11,7	**155,2**	15,2
1	50	34,6	6,5	34,7	1,0	**35,7**	**0,8**
	75	52,0	7,0	52,3	**1,7**	**53,5**	2,2
	100	68,8	7,0	68,9	**3,5**	**70,5**	3,8
	200	141,5	**7,0**	141,3	13,3	**143,6**	14,5
1.2	50	32,2	8,1	32,2	**0,7**	**33,1**	0,8
	75	47,8	8,5	47,5	**1,7**	**49,2**	1,8
	100	64,2	7,0	64,0	**3,2**	**65,8**	3,7
	200	129,0	**9,0**	128,4	16,3	**131,2**	15,5

En la Tabla 2 se muestra el promedio sobre las diez instancias generadas para cada tipo de problema, además de las seis combinaciones de A y C. *N_Trbs* representa el nº medio de trabajos realizados y *Tiempo* el número medio de segundos de CPU.

En la Tabla 3 se muestran los mismos datos que en la Tabla 2, pero realizando promedio con respecto al número de clases trabajos A y máquinas C.

En la Tabla 4 se recoge el error medio respecto a los soluciones óptimas de las instancias con 4 máquinas, y el número medio de óptimos obtenidos con cada método, agrupando respecto al ratio de utilización. Con más de 4 máquinas aparecía desbordamiento de memoria en la mayoría de los casos, además de tiempos de computación superiores a 24 horas, por lo que no

se pudieron completar las ejecuciones. El error se estimó como la diferencia entre el valor óptimo y el valor obtenido por el método correspondiente.

Tabla 3. Resumen de resultados respecto a ρ, *A y C*

		Kroon		*Gabrel*		*HC*	
ρ	*C*	*N_Trbs*	*Tiempo*	*N_Trbs*	*Tiempo*	*N_Trbs*	*Tiempo*
0.8	2	79,3	6,1	79,7	**2,3**	**80,4**	3,5
	3	82,9	8,7	83,9	**4,5**	**85,2**	6,0
	4	87,1	11,6	87,4	**9,5**	**90,2**	11,0
1	2	72,3	6,7	72,5	**2,3**	**73,3**	3,5
	3	75,7	10,2	76,0	**5,5**	**77,7**	6,3
	4	79,7	13,6	79,3	12,0	**82,4**	**10,0**
1.2	2	66,6	7,9	66,2	**2,0**	**67,6**	2,8
	3	69,6	11,7	69,6	**5,3**	**71,4**	6,0
	4	73,8	15,4	72,8	14,5	**76,3**	**10,8**

Tabla 4. Error y nº de óptimos

		Kroon		*Gabrel*		*HC*	
C	*n*	*Error*	*NOpt*	*Error*	*Nopt*	*Error*	*NOpt*
2	50	0,8	4,7	0,6	4	**0,1**	**9**
	75	1,4	2,7	1,2	2,3	**0,4**	**6,7**
	100	1,6	1	1,6	1,3	**0,5**	**5**
	200	3,2	0,3	3,3	**1**	**1,7**	**1**

De las Tablas 2, 3 y 4 se puede concluir que nuestra aproximación heurística presenta mejores resultados que los métodos heurísticos ya existentes para el problema FSP, en el 100% de los casos, con un tiempo de computación ligeramente superior.

6. Conclusiones

En este trabajo, hemos presentado un procedimiento heurístico de resolución para el problema conocido como *Fixed Job Scheduling Problem* (FSP), considerando varias clases de máquinas y trabajos, y el mismo peso para todos los trabajos. El procedimiento heurístico es de carácter constructivo y posee un procedimiento final de mejora local. La calidad del mismo se ha comparado con otros métodos heurísticos existentes en la bibliografía, así como con un método exacto. Los resultados computacionales ponen de manifiesto que nuestro método encuentra soluciones de mejor calidad que esos otros métodos en un tiempo de computación aceptable.

Referencias

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

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

Kolen, A.W.J., and Kroon, L.G. (1992). License class design: complexity and algorithms. *European Journal of Operacional Research*, 63, pp. 432-444.

Kolen, A.W.J., and Kroon, L.G. (1993). On the computacional complexity of maximum shift class scheduling. *European Journal of Operacional Research*, 64, pp. 138-151.

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