3rd International Conference on Industrial Engineering and Industrial Management
XIII Congreso de Ingeniería de Organización
Barcelona-Terrassa, September 2nd-4th 2009

Un procedimiento GRASP para la programación de pedidos en máquinas paralelas con tiempos de preparación*

Manuel Mateo[1], Imma Ribas[1], Ramon Companys[1]

[1] Dpto. de Organización de Empresas. Escuela Técnica Superior de Ingeniería Industrial de Barcelona. Universitat Politècnica de Catalunya. Av. Diagonal, 647, 08028. Barcelona. manel.mateo@upc.edu, imma.ribas@upc.edu, ramon.companys@upc.edu

Palabras clave: programación, máquinas paralelas, tiempo de preparación, GRASP, multicriterio.

1. Introducción

Los pedidos que se reciben en muchos entornos industriales incluyen cantidades de diferentes artículos con una fecha de entrega común. Llamaremos a la pareja (artículo, cantidad) línea de pedido, y su traducción en producción lote. En este trabajo, se estudia el problema de programación de los productos solicitados en máquinas diferentes en paralelo. La medida de eficiencia considerada tiene en cuenta más de un criterio, dado que un planificador generalmente tiene necesidades a veces contrapuestas, en particular el nivel de stock y el servicio ofrecido a los clientes. El primer criterio se puede medir con una media de tiempo de permanencia de los lotes de cada pedido en el sistema y el segundo, con la media de retraso por pedido. Por lo tanto, se ha creído oportuno usar como función objetivo la combinación ponderada de ambos valores.

Sea l lotes a producir en una de las m máquinas paralelas diferentes (j = 1,..., m). Se considera que el lote se fabrica en una única máquina, sin interrupciones. La fabricación de cada artículo i en la máquina j requiere un tiempo de proceso $u_{i,j}$ y un tiempo de preparación $s_{i,j}$, según la familia a qué pertenezca un producto. En consecuencia, un lote l, del artículo i, consume, si es realizado en la máquina j, un tiempo operatorio $p_{l,j}=u_{i,j}*q_{i,k}$, siendo $q_{i,k}$ la cantidad de unidades del artículo i solicitada en el pedido k. Sea **K** la cartera de pedidos existente. Para un pedido k se define:

- Una fecha de entrega d_k.
- Un conjunto de líneas **L**(k) con el producto i si se pide en el pedido k y la cantidad $q_{i,k}$ solicitada.

Es trivial transformar pedidos en lotes definiendo el tiempo de proceso de los lotes y los tiempos de preparación $\mathbf{S}_j$, para cada máquina. También se supone conocido el estado inicial de cada máquina función del último artículo fabricado (φ_j). Se considera que cualquier lote de un pedido se puede empezar a producir en el instante 0 y que las máquinas están disponibles en ese mismo instante.

* Este trabajo se deriva de la participación de sus autores en el proyecto de investigación financiado por el Ministerio Ciencia y Educación con referencia DPI2007-61371, titulado "Programación de operaciones multicriterio con máquinas paralelas, en varias etapas, sin interrupciones ni almacenajes".

Dado un programa de fabricación de los lotes se puede determinar, para cada pedido *k*, $C_k=\max_{(l\in \mathbf{L}(k))} c_l$ y su eventual retraso $T_k = \max\{0; C_k - d_k\}$.

La función objetivo bicriterio considerada (1) trata de minimizar la suma ponderada del retraso medio $\overline{T}$ y del tiempo de permanencia medio $\overline{C}$ de los diferentes pedidos:

$$[MIN]=\alpha\left(\frac{1}{n}\sum_k T_k\right)+(1-\alpha)\left(\frac{1}{n}\sum_k C_k\right) \quad (1)$$

2. Formalización del problema

Sea un conjunto **M** de *m* máquinas paralelas diferentes o no relacionadas (j = 1, ..., m). Sea un conjunto **N** de *n* familias (i = 1, ..., n). Cada una de las máquinas pueden tratar todos o un subconjunto de familias de productos. Así pues, $x_{i,j} = 1$ indica que la máquina *i* puede producir lotes de la familia *j* y $x_{i,j}=0$, en caso contrario. Si la máquina *j* puede producir lotes de la familia *i*, se define un tiempo de preparación $s_{i,j}$ para iniciar la producción de la familia *i* si el anterior lote fabricado era de una familia diferente, y un tiempo de proceso unitario $u_{i,j}$ del producto de la familia *i* en una máquina *j*.

Sea un conjunto **C** de *c* pedidos (k=1,…,c) que tiene la empresa. Para un pedido *k* se define:

- Una fecha de entrega d_k.
- El total de líneas de un pedido a_k (p=1,…, a_k).
- Para cada línea de pedido, el artículo pertenece a una familia *i* y la cantidad a fabricar q_p unidades. El lote de producto de la familia *i* requiere una operación cuyo tiempo de proceso en la máquina *j* es $p_{i,j}=u_{i,j}*q_i$. Se requiere un tiempo de preparación s_i si la familia *i* se fabrica después de lotes pertenecientes a productos de otras familias.

Además, se establecen las siguientes hipótesis:

- En el instante 0 las máquinas están disponibles y se conocen los pedidos a producir.
- Se conoce la familia del último producto fabricado en la máquina *j* (φ_j).

3. Estado del arte

El problema de programación de máquinas paralelas ha sido ampliamente estudiado (Chen & Sin, 1990; Pinedo, 1995). Incluso en los problemas más sencillos y siempre que la función de eficiencia no sea ΣC_j, en cuyo caso el óptimo puede hallarse en un tiempo polinómico, se demuestra que son problemas NP-hard (Karp, 1972).

Los problemas de programación de máquinas paralelas se han clasificado tradicionalmente en tres categorías diferentes (Graham *et al*., 1979). Sea $p_{i,j}$ el tiempo de proceso de la pieza o producto j en la máquina i: P, para máquinas idénticas: $p_{i,j}= p_{i,j'}$ para todas las máquinas $j\neq j'$; Q, para máquinas uniformes: $p_{i,j}=t_i/v_j$ para todas las máquinas j y un producto i; R, para máquinas diferentes que no guarden ninguna relación entre ellas. El problema tratado en este trabajo es de la categoría R| |. En términos de complejidad, los problemas de máquinas diferentes presentan mayor dificultad de resolución dentro de la configuración de máquinas paralelas. En Ribas (2007) se propone una ampliación de dicha notación en la que este caso se califica de máquinas en paralelo Arbitrarias (A).

Brucker (1998) presenta una serie de casos especiales de asignación de tareas y secuenciación de máquinas paralelas: piezas con restricciones de precedencia; piezas con fechas de entrega; tiempos de preparación; y tiempos de transporte. Algunos estudios recientes se han ocupado de la asignación y secuenciación de tareas a gran escala para sistemas de fabricación

complejos, como el caso real tratado en Yu et al (2002). Por ejemplo, Ovacik y Uzsoy (1996) desarrollan una clase de métodos heurísticos conocidos como métodos de descomposición. Precisamente, para este tipo de problemas, se utilizan a menudo simples heurísticas, como la de Dunstall y Wirth (2005) para máquinas paralelas idénticas, o metaheurísticas, como la búsqueda tabú en Logendran et al (2007), en este caso para máquinas paralelas diferentes. Los procedimientos exactos son más bien escasos, basados en esquemas branch-and-bound y para problemas de dimensiones reducidas, como en Rocha et al (2008).

En algunos de estos trabajos, los tres últimos citados, se tiene en cuenta la necesidad de tiempos de preparación dependientes o no de la máquina. En Allahverdi et al. (2008) se expone una actualización de una versión anterior (Allahverdi et al, 1999) de problemas con tiempos de preparación, convenientemente clasificados. Según ya planteaba Allahverdi et al. (1999), los problemas relacionados con las configuraciones se pueden clasificar teniendo en cuenta: los conjuntos de familias o no, y la preparación dependiente o independiente de la secuencia. En nuestro caso los productos a fabricar se agrupan en familias y los tiempos de preparación dependen de la secuencia, o sea que entre dos lotes para productos de una misma familia no es necesario tiempo de preparación y, en cambio, en caso contrario sí.

La programación de piezas en máquinas paralelas con tiempos de preparación dependientes de la secuencia ha sido tratada desde hace tiempo, aunque a menudo para un único criterio y para máquinas paralelas idénticas como en Lee y Pinedo (1997).

Respecto a la optimización muticriterio, ambos criterios se agregan en una función objetivo compuesta $F(f(\Pi), g(\Pi))$, donde Π es el programa evaluado. La mayoría de trabajos publicados abordan problemas de las categorías P y Q de máquinas paralelas (por ejemplo, McCormick y Pinedo, 1995). No obstante, en algunas ocasiones se plantea un problema de programación similar al tratado aquí, como en Hall y Potts (2004), pero con una única máquina y reprogramación con la llegada de nuevos pedidos.

4. Algoritmo GRASP propuesto

4.1. Estructura general del algoritmo

De acuerdo con la complejidad de resolución para el tipo de problemas tratado, se ha optado por procedimientos heurísticos. Entre los diversos procedimientos heurísticos, existen las llamadas metaheurísticas que presentan esquemas estándares de resolución; aquí se define un GRASP (Greedy Randomized Adaptive Search Procedure).

El GRASP propuesto se compone de dos fases consecutivas: en la primera fase, se determina una solución inicial; y en la segunda, la fase de mejora, se realiza la búsqueda local en el vecindario de la solución inicial.

Para empezar, en la primera fase del GRASP, se propone 3 reglas para la asignación de lotes a máquinas:

- Regla 1: basada en los tiempos de preparación (dando preferencia al menor tiempo de preparación)
- Regla 2: basada en los tiempos de proceso the processing times (dando preferencia al menor tiempo de proceso, o sea regla SPT –Shortest Processing Time-)
- Regla 3: basada en las fechas de entrega (donde se dé preferencia a la fecha de entrega más próxima, o sea regla EDD –Earliest Due Date-).

La determinación de la solución en la primera iteración, llamada *solucion0*, se hace siguiendo estrictamente la ordenación de lotes considerando cada regla. En el resto de iteraciones, cada

solución generada, llamada *solucion*, se determina introduciendo la aleatoriedad en la asignación y no eligiendo necesariamente según el binomio lote-máquina previsible a priori. A continuación, en cualquiera de ambos casos, se desarrolla la búsqueda local (fase II del algoritmo).

Para la segunda fase del GRASP, se combina un procedimiento de intercambio entre lotes de una o distintas máquinas y un procedimiento de inserción de un lote en una máquina diferente, que a veces requiere una reordenación posterior. En un caso, el número de lotes en cada máquina se mantiene, y en el otro, varía. El número de iteraciones se fijó en 100.000.

Ejemplo

A continuación, se presenta los datos de un ejemplar (en el caso de pedidos con una sola línea) que se utilizará para mostrar la aplicación de las reglas de la fase I. Las tres máquinas están preparadas inicialmente para fabricar las familias 4, 7 y 2, respectivamente.

Tabla 1. Tiempos unitarios de fabricación por máquina y familia.

$u_{i,j}$	1	2	3	4	5	6	7	8
1	1.8	0.7	0.8	0.8	0.5	1.1	-	0.2
2	1.4	1.0	0.1	0.5	1.5	0.6	1.6	0.9
3	0.9	0.5	0.9	1.9	0.5	0.6	1.4	1.3

Tabla 2. Tiempos de preparación por máquina y familia.

$s_{i,j}$	1	2	3	4	5	6	7	8
1	39	38	21	38	47	41	-	50
2	23	29	34	43	45	34	48	48
3	48	30	49	28	49	27	46	42

Tabla 3. Datos de los lotes (líneas de pedido, cantidad, familia y fecha de entrega).

k,i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
d_i	140	150	160	270	280	210	280	220	220	250	320	180	240	340	220
f_i	4	6	8	7	4	2	3	5	5	5	1	7	2	4	3
q_i	60	30	80	70	60	30	60	10	50	40	90	100	30	70	30

4.2. Regla 1: solución basada en los tiempos de preparación

La regla 1 se aplica de la siguiente manera. Si se debe fabricar alguna línea de pedido de la familia para la cual está preparada alguna de las máquinas, se asigna a ésta. Para el resto de líneas de pedidos, se ordena los tiempos de preparación de los diferentes lotes a fabricar por cada familia aún no asignada y se asigna el lote a la máquina que suponga el mínimo tiempo de preparación. A esta asignación la llamaremos asignación base. A partir de aquí, para hallar *solucion0* en la primera iteración, los lotes se ordenan según la regla EDD dentro de cada familia y se evalúa la función objetivo. Después de esto, se pasa a la fase de mejora local.

Ejemplo. Aplicando la regla bajo un enfoque determinista, la asignación para la primera solución es:

- Máquina 1: lotes 1, 5, 14 (por ser los tres de la familia 4), 7 y 15 (familia 3).
- Máquina 2: lotes 4,12 (familia 7), 11 (familia 1), 8, 9, 10 (familia 5).
- Máquina 3: lotes 6,13 (familia 2), 2 (familia 6), 3 (familia 8).

Aleatoriedad. Para hallar *solución* del resto de iteraciones, se parte de la asignación base y se genera una lista que empieza por los lotes asignados a la primera máquina y acaban por los de la última. Siguiendo la lista, se selecciona lote y lote y equiprobablemente se elige una de las máquinas que puedan realizar la producción de aquella línea de pedido. Se secuencia tal cual se van asignando lotes y se evalúa la función objetivo. Para cada máquina se ordena la lista de lotes asignados de menor a mayor tiempo de preparación. Se vuelve a evaluar la función objetivo y en caso de mejor valor, se retiene esta última secuencia como solución inicial. A continuación, se pasa a la fase de mejora local.

4.3. Regla 2: solución basada en los tiempos de proceso

Para cada lote se busca la máquina que implique la operación más corta, y se asigna a esa máquina. Los posibles empates se resuelven asignando el lote a la máquina menos cargada. A esta asignación la llamaremos asignación base. A partir de aquí, para hallar *solucion0* en la primera iteración, los lotes se ordenan de menor a mayor tiempo de proceso y se evalúa la función objetivo. Después de esto, se pasa a la fase de mejora local.

Ejemplo. La asignación aplicando la regla bajo un enfoque determinista da como resultado:

- Máquina 1: lotes 8, 9,10 (de la familia 5), y 3 (familia 8).
- Máquina 2: lotes 7,15 (familia 3), y 1, 5, 14 (familia 4).
- Máquina 3: lotes 11 (familia 1), 6 y 13 (familia 2), 2 (familia 6) y 4, 12 (familia 7).

Aleatoriedad. A partir de la asignación base, se genera una lista como en la anterior regla. Lote a lote, y de forma equiprobable, se les asigna a una máquina que pueda producir esa línea de pedido. Se secuencia tal cual se van asignando lotes y se evalúa la función objetivo. Para cada máquina, se ordena los lotes asignados de menor a mayor tiempo de proceso. Se vuelve a evaluar la función objetivo y si mejora el valor anterior, se retiene esta última secuencia como solución inicial. Después de esto, se pasa a la fase de mejora local.

4.4. Regla 3: solución basada en las fechas de entrega

En este caso, se ordenan los lotes por fecha de entrega, de menor a mayor. Si en algún momento se pueden evitar preparaciones, se asigna un lote a la máquina correspondiente; también se tiene en cuenta donde durará menos la fabricación y se busca encontrar un equilibrio entre las asignaciones de las distintas máquinas. A esta asignación la llamaremos asignación base. Para esta *solucion0* de la primera iteración, se evalúa la función objetivo. Seguidamente, se pasa a la fase de mejora local.

Ejemplo. Aplicando la regla bajo un enfoque determinista, la asignación es la siguiente:

- Máquina 1: lotes 1, 8, 15, 4, 7.
- Máquina 2: lotes 3, 12, 9, 10, 11.
- Máquina 3: lotes 2, 6, 13, 5, 14.

Aleatoriedad. A partir de la asignación base, se elige lote a lote al azar; si el lote corresponde a la familia para la cual está preparada una máquina, se asigna inmediatamente, y si no, se asigna al azar entre la máquina que minimice su tiempo de preparación o de proceso. En este último caso, se elige la máquina que equilibre el número de lotes asignados por máquina. Se secuencia tal cual se van asignando lotes y se evalúa la función objetivo. Después de esto, se pasa a la fase de mejora local.

4.5. Fase II, búsqueda local

En este caso, se selecciona al azar uno de los lotes secuenciados. Una herramienta, llamada revolver (Ribas, 2007), permite explorar el vecindario al azar. Esto se realiza determinando desde qué lote de la secuencia se producen los intercambios (en caso de mover la posición de dos de los lotes, manteniendo el número de lotes por máquina) o las inserciones (en este caso, puede producirse el desplazamiento de la fabricación de un lote internamente en una máquina o cambiar de máquina asignada). Con este cambio, situaciones habituales observadas en problemas de programación (como la tendencia a elegir siempre los mismos vecinos en caso de empate) tienden a desaparecer.

5. Experiencia computacional

Se ha contrastado la eficiencia de las tres reglas para obtener una solución inicial, usadas en la primera fase, mediante una experiencia computacional. En los ejemplares analizados, el número de máquinas *m* varía entre 3 y 9, el número de pedidos *k* entre 1 y 14, y cada pedido entre 1 y 4 líneas por pedido, lo que implica un mínimo de 15 líneas de pedido y un máximo de 25. El número máximo de familias es de 15.

Se ha trabajado sobre tres conjuntos de 100 ejemplares cada uno:

- Conjunto (8_3): productos pertenecientes a 8 familias diferentes (i=1,…,8) a programar en 3 máquinas paralelas independientes (j=1,2,3).
- Conjunto (12_6): caso de productos pertenecientes a 12 familias diferentes (i=1,…,12) a programar en 6 máquinas paralelas independientes (j=1,…,6).
- Conjunto (15_9): caso de productos pertenecientes a 15 familias diferentes (i=1,…,15) a programar en 9 máquinas paralelas independientes (j=1,…,9).

El diseño de los ejemplares se basó en los cuatro siguientes parámetros:

- Los tiempos de proceso por familia $u_{i,j}$ son valores que se distribuyen según una ley uniforme en [0,1;2], redondeados a un decimal.
- Los tiempos de preparación $s_{i,j}$ son valores enteros, determinados a partir de una distribución uniforme [20;50].
- Las fechas de entrega se distribuyen uniformemente entre [*dmin*, *dmax*], donde $d_{max} = \lambda_2 fv$; $d_{min} = \lambda_1 fv$; $\lambda 1=0,3$; $\lambda 2=0,9$ y *fv* es una estimación del makespan.
- Las cantidades a fabricar por línea de pedido están uniformemente distribuidas en el intervalo [10; 100], siempre en valores múltiplos de 10.

Siendo el total de lotes a fabricar entre 15 y 25, el número de productos diferentes solicitados por pedido puede variar entre 1 y 4.Tanto la familia para la cual cada máquina se encuentra disponible inicialmente como las familias de las líneas de pedido se determinan de manera aleatoria sobre la distribución uniforme [1; *n*].

Las pruebas fueron efectuadas en un Pentium V Quad con 2 Gb RAM y procesador 2,4 GHz para los tres conjuntos de instancias. Se ha realizado un mínimo de tres réplicas por cada conjunto de instancias, valor de α y regla.

6. Análisis de los resultados

La ejecución del algoritmo se ha realizado para dos supuestos:

1. En primer lugar, se estudia la situación en que cada pedido está formado por una única línea, con lo cual el número de lotes y pedidos coinciden.
2. Después, se pasa a estudiar la situación en que cada pedido puede estar formado por más de una línea.

En las siguientes tablas (Tabla 4 y Tabla 5) se compara el porcentaje de ejemplares de cada conjunto que alcanzan en media el mejor resultado, valor mínimo de la función objetivo, en cada uno de ambos supuestos.

Como se observa en la Tabla 4, la regla 3 basada en las fechas de entrega es la más eficiente, ya que conduce para cualquier colección y valor de α, un mayor número de veces a la mejor solución (casi siempre por encima del 75% de veces). Esta supremacía se manifiesta aún más para los ejemplares de mayor dimensión y valores de α=0,5 y α=0, en cuyo caso las otras reglas no llegan al 10% y R3 sobrepasa el 90%.

Tabla 4. Porcentaje de ejemplares en que se alcanza la mejor solución, por colección y α (1 línea por pedido).

fam_maq	α = 1			α = 0,5			α = 0		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
8_3	59	56	82	35	54	74	34	44	74
12_6	43	49	77	15	21	87	9	18	84
15_9	24	25	75	5	4	93	3	4	95

En cambio, si se analiza el segundo supuesto (Tabla 5), la regla 3 ya no se ha demostrado mejor en todos los casos. En una situación (α=1 para la colección 12_6), la regla 1 consigue globalmente mejores resultados, aunque la regla 3 se sitúa en segundo lugar, no lejos de la primera. Se observa como en este caso (α=1 para la colección 12_6) las diferencias entre reglas han disminuido y el número de mejores soluciones según reglas está prácticamente igualado (85, 80 y 82 respectivamente).

Tabla 5. Porcentaje de ejemplares en que se alcanza la mejor solución, por colección y α (más de 1 línea por pedido).

fam_maq	α = 1			α = 0,5			α = 0		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
8_3	82	78	85	41	40	59	37	40	52
12_6	85	80	82	49	34	58	36	26	59
15_9	55	51	76	32	28	71	26	23	76

Además, de considerar estos valores en porcentaje de veces que se alcanza la mejor solución, también puede evaluar la desviación sobre éste. Para ello se calcula:

$$x = \frac{Alg - Best}{Best} * 100 \quad (2)$$

siendo *Alg* el valor de la función objetivo para la solución dada por un cierto algoritmo y *Best* el valor de la mejor solución hallada.

En la Tabla 6 y Tabla 7 (para ambos supuestos) se compara el valor medio de *x* de cada colección basado en el mejor resultado, valor mínimo de la función objetivo.

Tabla 6. Valor medio de x por colección y α (1 línea por pedido).

fam_maq	α = 1			α = 0,5			α = 0		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
8_3	62,24	70,22	12,07	11,38	12,88	0,53	11,06	12,50	0,41
12_6	76,90	61,90	28,24	21,09	23,18	0,36	20,54	21,99	0,27
15_9	96,37	92,78	14,26	26,88	30,30	0,21	26,40	29,48	0,12

Tabla 7. Valor medio de x por colección y α (más de 1 línea por pedido).

fam_maq	α = 1			α = 0,5			α = 0		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
8_3	17,13	6,34	5,69	6,20	6,55	1,34	5,97	6,22	1,08
12_6	24,81	10,98	8,27	3,62	11,06	2,60	9,43	10,34	2,06
15_9	38,35	29,54	9,78	11,15	11,32	2,09	10,86	11,03	0,98

Observando las Tablas 6 y 7, la regla 3 es la que siempre presenta mejores resultados, tanto en un supuesto como en el otro. Esto indica que, aunque con esta regla no se obtenga la mejor solución, el valor de la función objetivo alcanzado no se aleja del de la mejor solución. Además, para más de una línea por pedido (Tabla 7) el valor *x* de la desviación disminuye cuanto menor es α. Estas diferencias también son apreciables si se muestra los gráficos de las mejores soluciones partiendo de cada una de las reglas (Figura 1).

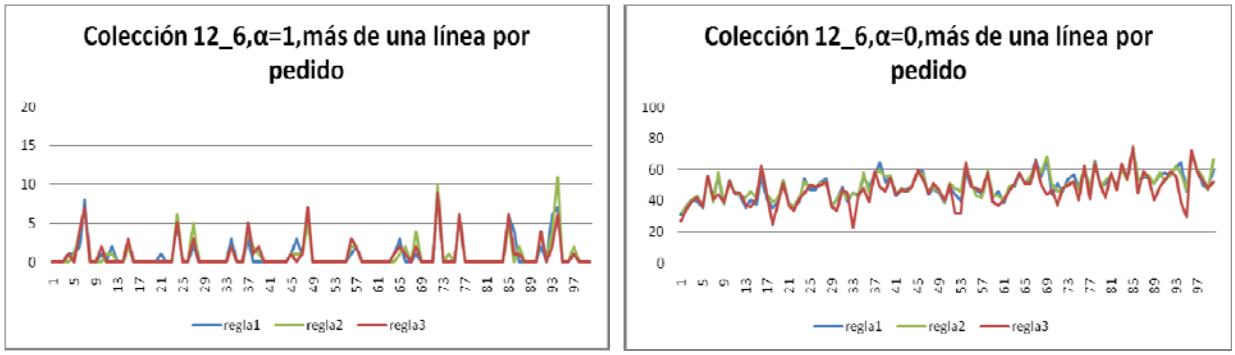

Figura 1. Mejores soluciones según cada regla para la colección 12_6 con α=1 y α=0, en caso de más de una línea por pedido.

Es de destacar que en caso de una línea por pedido, el valor *x* está por debajo de 1 para valores de α=0,5 y α=0. En cambio, en caso de más de una línea por pedido, para los mismos valores de α, *x* es un poco superior y se encuentra entre 0,98 y 2,60.

En definitiva, la regla 3, basada en la ordenación EDD de los lotes para su programación, proporciona mejores resultados haya una línea de pedido o varias e independientemente del peso de cada uno de los dos criterios.

7. Conclusiones

En el presente artículo se ha presentado un procedimiento tipo GRASP para la programación de máquinas paralelas diferentes, considerando tiempos de preparación y con un objetivo bicriterio que pondera el retraso medio y el tiempo medio de permanencia. La regla utilizada que en la determinación de la solución ha dado mejores resultados con diferencia ha sido la que se basa en una ordenación EDD.

En futuras investigaciones, se prevé el estudio de las respuestas del algoritmo a otras funciones bicriterio, en el caso de múltiples lotes por pedido. Así pues, además de los resultados basados en los valores medios -ΣT, ΣC-, puede evaluarse los resultados basados en los valores máximos -Tmax, Cmax-. Los procedimientos también pueden ser más eficientes, en especial el tratamiento de los retrasos, con el desarrollo de cotas ajustadas.

Referencias

Allahverdi, A.; Gupta, J.N.D.; Aldowaisan, T. (1999).A review of scheduling research involving setup considerations. Omega, International Journal of Management Science, Vol. 27, pp. 219-239.

Allahverdi, A.; Ng, C.T.; Cheng, T.C.E.; Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. European Journal of Operational Research, Vol. 187, pp. 985-1032.

Brucker, P. (1998). Scheduling algorithms, 2nd edition, Springer, Heidelberg.

Chen, T.C.E.; Sin, C.C.S. (1990). A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research, Vol. 47, pp. 271-292.

Dunstall, S.; Wirth, A. (2005). Heuristic methods for the identical parallel machine flowtime problem with setup times, Computers and Operational Research, Vol. 32, pp. 2479–2491.

Graham, R.L.; E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan (1979): Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 4, 287-326.

Hall, N.G.; Potts, C.N. (2004). Rescheduling for new orders. Operations Research, Vol. 52, No. 3, pp. 440-453.

Karp, R.M. (1972). Reducibility Among Combinatorial Problems in R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum. pp. 85–103.

Lee, Y.H.; Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence dependent setup times, European Journal of Operational Research, Vol. 100, pp. 464–474.

Logendran, R.; McDonell, B.; Smucker, B. (2007). Scheduling unrelated machines with sequence-dependent setups. Computers and Operations Research, Vol. 34, No. 11, pp. 3420-3438.

McCormick, S:T.; Pinedo, M. (1995). Scheduling n independent on m uniform machines with both flow time and makespan objectives: A parametric analysis. ORSA J. Computing. Vol. 7, pp. 63-77.

Ovacik, I.M.; Uzsoy, R. (1996). Decomposition Methods for Scheduling Semiconductor Testing Facilities, *International Journal of Flexible Manufacturing Systems Vol. 8*, pp. 357-388.

Pinedo, M. (1995). Scheduling: Theory, Algorithms and Systems. Prentice-Hall

Ribas, I. (2007). Programación Multicriterio de un Sistema Productivo con Flujo Regular sin Esperas y Estaciones en Paralelo. Aplicación a una Fábrica de Helados. Tesis Doctoral. UPV.

Rocha, P.L.; Ravetti M.G.; Mateus, G.R.; Pardalos, P.M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. Computers and Operations Research, Vol. 35, Nº 4, pp. 1250-1264.

Yu, L.; Shih, H.M.; Pfund, M.; Carlyle, W.M.; Fowler, J.W. (2002). Scheduling of Unrelated Parallel Machines : An Application to PWB Manufacturing, IIE Transactions on Scheduling and Logistics, Vol. 34, Nº 11, pp. 921-931.