Programación multicriterio de piezas en un taller de máquinas en paralelo con especialización

Manuel Mateo
UPC, Barcelona, España, manel.mateo@upc.edu

Aballo José
UPC, Barcelona, España, aballo.jose@upc.edu

Mayra D'Armas
UNEXPO, Puerto Ordaz, Venezuela, mdarmas@unexpo.edu.ve

ABSTRACT

In this paper a variant of the scheduling problem of parallel machines is solved in systems where *n* jobs are divided in different categories exist and *m* machines classified in the same categories. In particular, the study is focussed when machines and jobs are separated into three categories (high, medium and low). Each machine can produce jobs of its own or a lower level. Two simultaneous objectives are considered: minimize the makespan (c_{max}) and the total penalty (w) for the cost of making a job in a machine not in the high level. The problem was solved by applying a reallocation heuristic using time criteron and a GRASP. As a result, a number of feasible sequences for each instance are obtained, with corresponding values c_{max} and w_{tot}. In order to verify the efficiency of the procedure, a computational experience with instances with up to 10 machines and 200 jobs is carried out, where the difference in percentage between c_{max} of the first and the last solutions and the average number of obtained solutions are analyzed.

Keywords: parallel machines, multi-criteria programming, heuristic, GRASP

RESUMEN

En este trabajo se resuelve una variante del problema del taller mecánico con máquinas en paralelo, en sistemas donde existan *n* piezas de categorías diferentes y *m* máquinas clasificadas en las mismas categorías. En particular, se estudia el escenario donde máquinas y piezas pueden encontrarse en tres categorías (alta, media y baja). Cada máquina podrá producir las piezas de su nivel y sus inferiores. Se considerarán como objetivos simultáneos la minimización del instante de entrega de la última pieza (c_{max}) y la penalización (w_{tot}) correspondiente al costo de fabricar una pieza en una máquina de categoría inferior a la máxima. El problema se resolvió mediante la aplicación de una heurística de reasignación por criterios de tiempo y un procedimiento GRASP. Como resultado se obtienen varias secuencias factibles para cada ejemplar con sus correspondientes valores de c_{max} y w_{tot}. Con la finalidad de verificar la eficiencia del procedimiento de resolución, se realizó una experiencia computacional sobre instancias de hasta 10 máquinas y 200 piezas, donde se analiza la diferencia porcentual de c_{max} de la primera a la última solución y el número medio de soluciones obtenidas.

Palabras claves: máquinas en paralelo, programación multi-criterio, heurística, GRASP

1. INTRODUCCIÓN

El problema de programación de máquinas paralelas es muy usual en las empresas. Una sola operación debe hacerse en un conjunto de piezas y sólo es necesario realizarlo en una de las máquinas disponibles. Los tiempos de procesamiento de las operaciones se conocen normalmente (Pinedo, 2002). Algunos problemas de asignación y secuenciación para dos o más máquinas paralelas se han descrito como muy complejos (Blazewicz et al, 2001).

En este trabajo nos ocupamos de un problema particular en la programación de máquinas en paralelo: cuando ciertas piezas no puede realizarse en algunas máquinas, lo que se conoce en la literatura como elegibilidad (Leung y Li, 2008).

Dado un conjunto de *n* piezas (j=1,...,n) a programar en *m* máquinas paralelas (i=1,...,m) , estas máquinas se distribuyen generalmente entre *p* grupos o niveles (k=1,...,p). En particular, el algoritmo propuesto se plantea para p=3, que significa que las máquinas y las piezas se dividen en 3 niveles (k=1 se considera para máquinas de nivel alto; k=2 se considera de nivel medio y k=3 para de nivel bajo). Todas las máquinas deben clasificarse en uno de los tres niveles. Lo mismo sucede para las piezas: cada una es asignada a uno de los niveles. Una máquina en el nivel *k* puede fabricar piezas de su propio nivel *k* y también de los niveles inferiores. El tiempo de proceso de una pieza es el mismo para cualquier máquina. Esto se conoce en la literatura como proceso anidado (Leung y Li, 2008).

Dado una pieza *j*, se conoce el tiempo de proceso p_j en la máquina paralela, el instante de disponibilidad o tiempo de pre-proceso r_j (también llamado tiempo de cabeza), que puede ser consecuencia de las anteriores operaciones recibidas por la pieza, y el tiempo de entrega o post-proceso q_j (también tiempo de cola), como resultado de las tareas subsiguientes y el transporte al final del sistema de producción de las piezas.

En esta situación y debido a razones económicas, la empresa prefiere utilizar las máquinas de alto nivel antes de que las otras, y también se prefieren las máquinas de nivel medio antes que las máquinas de bajo nivel. Por tanto, se define una penalización *w*, también conocida como peso, cuyo valor es 1 si una pieza pasa a programarse en una máquina de nivel medio en lugar de una de nivel alto; el valor es 1 si una pieza se programa en una máquina de nivel bajo en lugar de una de nivel medio; y es 2 si una pieza se traslada de una máquina de nivel alto a una de bajo.

Así pues, se tienen en cuenta dos objetivos al determinar una programación factible: el tiempo mínimo de realización o makespan, c_{max} , y la penalización mínima total w_{tot}. Una solución se representará mediante un vector σ, donde cada uno de sus elementos σ_j se compone de la máquina asignada y el instante de inicio de proceso de esa pieza *j*. Un programa será factible si se cumplen las siguientes condiciones:

- Cada máquina procesa como máximo una pieza a la vez.
- Una pieza sólo se procesa en una sola máquina.
- No se permite interrupciones de las operaciones.
- El tiempo de inicio no es menor que el tiempo de disponibilidad: $t_j \geq r_j$.
- Una pieza asignada a un nivel *k* se procesa en una máquina de su mismo nivel o superior.

Teniendo en cuenta los tiempos de pre-proceso y post-proceso de las piezas, el algoritmo propuesto tiene como referencia para cada nivel al algoritmo descrito en Gharbi y Haouari (2002), que considera los dos conjuntos de tiempos, pero no la elegibilidad, que requiere trabajar con subconjuntos de piezas en cada nivel.

La Sección 2 presenta el estado del arte en máquinas paralelas con la elegibilidad y tiempos de pre-proceso y post-proceso. La Sección 3 describe el problema y la notación, mientras que la Sección 4 presenta la Heurística de enfoque bicriterio propuesta. La Sección 5 describe el procedimiento GRASP también para optimización bicriterio y la Sección 6, la experiencia computacional. Finalmente, se plantean algunas conclusiones.

2. EL PROBLEMA DE LA MÁQUINA EN PARALELO, LOS TIEMPOS ASOCIADOS Y LA ELEGIBILIDAD

El problema se puede notarse como $P_m|r_j;q_j;M_j|(c_{max}, w_{tot})$, donde la pieza j sólo puede ser producida en un subconjuntoMj de las m máquinas. $P_m|r_j;q_j|c_{max}$ o $P|r_j;q_j|c_{max}$ es una extensión del clásico problema de máquinas paralelas idénticas ($P||c_{máx}$), que es básico en programación. Nuestro problema, teniendo en cuenta sólo el primer objetivo, puede verse también como una generalización del clásico problema de una máquina $1|r_j;q_j|c_{max}$, que se sabe que es fuertemente NP-hard. Por lo tanto, el $P_m|r_j;_j;M_j|(c_{max},w_{tot})$ es NP-hard en sentido fuerte.

Los problemas de máquinas paralelas han recibido la atención de la comunidad investigadora desde hace mucho tiempo. Una revisión de esta literatura se presenta, por ejemplo, en Pinedo (2002). Si se añaden tiempos de pre-proceso y post-proceso, el problema se convierte en $P|r_j;q_j|c_{max}$, pero a pesar de su interés teórico y práctico, sólo

ha recibido una atención reducida. De hecho, la literatura referida a este problema no es muy amplia. Según nuestro conocimiento, el único algoritmo exacto en la literatura para este problema se desarrolló por Carlier (1987). Carlier (1987) y Gharbi y Haouari (2002) analizan reglas de secuenciación simples. Lancia (2000) presenta un método enumerativo para el problema con dos máquinas independientes, notado como $R2|r_j;q_j|c_{max}$. También se ha investigado otros problemas similares. Por una parte, para máquinas paralelas uniformes Dessouky (1998) desarrolla un algoritmo branch-and-bound para $Q|r_j;q_j;p_j=1|c_{max}$.

Por otro lado, puesto que $P||c_{max}$ es NP-hard en sentido fuerte, el problema teniendo en cuenta la elegibilidad es también fuertemente NP-hard. De acuerdo con Leung y Li (2008) existen dos casos importantes de elegibilidad tratados en la literatura: las restricciones para piezas anidadas y las restricciones globales de proceso. El primero tiene la propiedad de que para cada par M_j y M_k, ya sea M_j y M_k son disjuntos, M_j se incluye en M_k, o M_k está incluido en M_j. El segundo es un caso especial de tratamiento anidado en que por cada par M_j y M_k, ya sea M_j está incluido en M_k o M_k está incluido en M_j.

Nuestro problema se puede clasificar en el segundo grupo de situaciones. En este caso las restricciones globales de procesamiento, cada M_j se asocia con una sola máquina indexada a_j de tal manera que indica la primera o la última máquina útil para la pieza j dependiendo de la secuencia de las máquinas. En nuestro caso, a_j implicará que una pieza j puede procesarse entre las máquinas 1 y a_j, de acuerdo con sus niveles. Creemos que sólo Centeno y Armacost (1997) y una vez más Centeno y Armacost (2004) han trabajado en ambas características al mismo tiempo. Sin embargo, ellos sólo buscan la minimización del *makespan* en máquinas paralelas con tiempos de pre-proceso y las restricciones de elegibilidad de la máquina. No tienen en cuenta los tiempos de post-proceso.

3. EL PROBLEMA MULTICRITERIO

Nuestro problema se basa en la programación de la empresa dedicada a las pinturas industriales. Cuenta con un conjunto de máquinas, que son reactores. Los directivos de la empresa prefieren el uso de los recursos más modernos, en nuestro caso, las máquinas de alto nivel. Sin embargo, si todos los trabajos (piezas en nuestra notación) se deben hacer en este subgrupo de máquinas el *makespan* alcanza un valor muy alto. Y el resto de las máquinas están totalmente desocupadas. Por esta razón, algunas piezas en máquinas de nivel alto se trasladan al resto de máquinas. Así, los movimientos sucesivos conducen a un incremento del valor de penalización combinado con un descenso del *makespan*. Se propone un algoritmo bi-objetivo para afrontar este problema y proponer al gerente una aproximación del conjunto de programas eficientes.

Un estudio sobre investigación multicriterio se puede encontrar en Ehrgott y Gandibleux (2002). Según Loukil et al (2007), en lo referente a problemas de optimización mutiobjetivo podemos distinguir cinco principales enfoques en la literatura:

- Enfoque jerárquico: los objetivos se clasifican en un orden de prioridad y se optimizan siguiendo este orden.
- Enfoque utilidad: utiliza una función de utilidad o función ponderación, a menudo combinación lineal ponderada de los objetivos, para unir los objetivos considerados en uno solo.
- Programación por metas: se tienen en cuenta todos los objetivos como limitaciones que expresan niveles de satisfacción (o metas) y el objetivo es encontrar una solución cuyos valores sean lo más cercanos a la meta pre-definida para cada objetivo.
- Enfoque simultáneo (o de Pareto): el objetivo es generar, o aproximar en caso de un método heurístico, el conjunto completo de soluciones eficientes.
- Enfoque interactivo: en cada paso del procedimiento, el decisor expresa sus preferencias respecto una (o varias) de las soluciones propuestas. Por lo tanto, el método convergerá progresivamente en la satisfacción de las concesiones a los objetivos considerados.

Nuestro procedimiento se puede clasificar en el cuarto enfoque. Recordamos que para un problema de optimización multi-objetivo:

$$\min_{X \in S} \; z_k(X) \quad k = 1, \ldots, K$$

Una solución $X^* \in S$ es eficiente o Pareto óptima (o no dominada) si no hay otra solución $X \in S$ tal que $z_k(X) \leq z_k(X^*)$ $\forall k$ con al menos una desigualdad estricta.

A pesar de su aplicabilidad, no se ha prestado mucha atención a los problemas con criterios múltiples de programación, sobre todo en problemas de máquinas múltiples. Esto se debe a la complejidad de estos problemas combinatorios. Analizando la literatura, se observa que a menudo los métodos propuestos son o muy complejos de implementar o sólo capaces de resolver instancias de pequeño tamaño y con dos objetivos.

4. PROCEDIMIENTO HEURÍSTICO PROPUESTO

Una vez definidas todas las características del problema se procede a describir la heurística que se ha desarrollado para obtener las programaciones que formarán la solución del problema. Para conocer qué piezas hay en cada nivel se utilizarán los conjuntos J_H, J_M y J_L que corresponderán, por sus siglas en inglés, a las piezas programadas en el nivel alto, medio y bajo respectivamente. La heurística consta de dos fases:

Fase 1: cálculo de la solución inicial. Siempre se encuentra programando todas las piezas en las máquinas de categoría alta.

Fase 2: cambio de categoría de varias piezas para encontrar nuevas soluciones que sean no dominadas. Se busca un conjunto de piezas candidatas a cambiar de nivel (JC). Se considera candidatas todas aquellas piezas que, por su complejidad, pueden producirse en otra categoría diferente a donde están programadas en la solución en curso. Cambiando de categoría una de estas piezas ($j_m \in JC$) se encontrará una nueva solución. Esta se evaluará mediante c_{max} (c_{max}^c). En caso de ser no dominada respecto a las *s* soluciones existentes, definidas por (c_{max}^v ,w_{tot}^v) v=1,...,s pasará a formar parte del conjunto de soluciones ρ.

A continuación se describen las características, la formalización y los puntos fuertes y débiles de la heurística diseñada (reasignación por criterios de tiempo). Dicha heurística consta de tres partes, donde cada una corresponde a un cambio de categoría diferente entre los que se pueden realizar en este problema:

- Reasignación de una pieza desde máquina de categoría alta a una de categoría media.
- Reasignación de una pieza desde máquina de categoría alta a una de categoría baja.
- Reasignación de una pieza desde máquina de categoría media a una de categoría baja.

4.1 CARACTERÍSTICAS DE LA HEURÍSTICA

La principal característica que presenta esta heurística es el método que se utiliza para escoger cuál será la pieza a cambiar de nivel, o "mover", entre todas las candidatas en cada iteración ($j_m \in JC$). En este caso se han elegido varios criterios según sus tiempos de producción (r_j, p_j y q_j) para determinar cuál es la pieza seleccionada en cada caso. Los criterios probados han sido:

$j_m = \max_{j \in JC} (p_j)$ $\qquad$ $j_m = \min_{j \in JC} (r_j+p_j, q_j+p_j)$

$j_m = \min_{j \in JC} (p_j)$ $\qquad$ $j_m = \max_{j \in JC} (r_j, q_j)$

$j_m = \max_{j \in JC} (r_j+p_j, q_j+p_j)$ $\qquad$ $j_m = \min_{j \in JC} (r_j, q_j)$

En caso de que se produzca un empate, se escoge la pieza programada más hacia inicio de la secuencia ya que se considera que tiene mayor repercusión en el conjunto, por el hecho de provocar el movimiento de un número mayor de piezas en toda la secuencia.

En un estudio previo realizado se determinó que el criterio de selección de la pieza más adecuado para escoger la pieza a cambiar de nivel es $j_m = \max_{j \in JC} (p_j)$. El tiempo de proceso de una pieza (p_j) es justamente el único tiempo que las piezas pasan dentro de la máquina. Por este motivo, parece claro que las piezas que pasen más tiempo en la máquina sean las que provocarán una mayor reducción en c_{max} si cambian de categoría.

4.2 FORMALIZACIÓN DE LA HEURÍSTICA

Seguidamente se describe el procedimiento utilizado:

Fase 1 - Encontrar la solución inicial, dados $J^H=\{1,...,n\}$; $J^M=\varnothing$; $J^L=\varnothing$.
Para programar todas las piezas en el nivel alto, aplicar las heurísticas $m_k=1$ (inspirada en Johnson, 1954) o $m_k>1$ (Gharbi y Haouari, 2002) para k=1, según convenga.
1. Encontrar solución inicial (s = 1).
2. $\rho_1 = (c_{max}^1, 0)$

Fase 2 - Reasignación de las piezas a diferentes niveles.
$c_{max}^c = 0$
MIENTRAS $c_{max}^c < c_{max}^s$
1. Reasignación de una pieza desde la categoría alta (h) hasta la categoría media (m).
1.1. Determinar el conjunto de piezas candidatas a mover (JC).
1.2. SI JC = $\varnothing \rightarrow$ Fase 2-2 (si no hay candidatas, pasar al segundo tipo de movimiento)
En caso contrario:
1.2.1. Elegir la pieza a reasignar (j_m) según el criterio prefijado anteriormente.
1.2.2. Reprogramar las piezas asignadas a la categoría alta ($J_H = JH-\{j_m\}$).
1.2.3. Reprogramar las piezas asignadas a la categoría media ($J_M = J_M + \{j_m\}$).
1.2.4. Determinar la c_{max} global de la solución candidata (c_{max}^c).
1.2.5. SI $c_{max}^c > c_{max}^s \rightarrow$ Fase 2-2
En caso contrario, añadir solución candidata al conjunto de soluciones (s=s+1):
$\rho_{s+1} = (c_{max}^{s+1} = c_{max}^c, w^{s+1} = w^s + 1)$
fin MIENTRAS

MIENTRAS $c_{max}^c < c_{max}^s$
2. Reasignación de una pieza desde la categoría alta (h) hasta la categoría baja (l).
2.1. Determinar el conjunto de piezas candidatas a mover (JC).
2.2. SI JC = $\varnothing \rightarrow$ Fase 2-3 (si no hay candidatas, pasar al tercer tipo de movimiento)
En caso contrario:
2.2.1. Elegir la pieza a reasignar (jm) según el criterio prefijado anteriormente.
2.2.2. Reprogramar las piezas asignadas a la categoría alta ($J_H = J_H-\{j_m\}$).
2.2.3. Reprogramar las piezas asignadas a la categoría media ($J_L = J_L + \{j_m\}$).
2.2.4. Determinar la c_{max} global de la solución candidata (c_{max}^c).
2.2.5. SI $c_{max}^c > c_{max}^s \rightarrow$ Fase 2-3
En caso contrario, añadir la solución candidata al conjunto de soluciones (s = s +1):
$\rho_{s+1} = (c_{max}^{s+1} = c_{max}^c, w^{s+1} = w^s + 2)$
fin MIENTRAS

MIENTRAS $c_{max}^c < c_{max}^s$
3. Reasignación de una pieza desde la categoría media (m) hasta la categoría baja
3.1. Determinar el conjunto de piezas candidatas a mover (JC).
3.2. SI JC = $\varnothing \rightarrow$ FIN (si no hay candidatas, final)
En caso contrario:
3.2.1. Elegir la pieza a reasignar (j_m) según el criterio prefijado anteriormente.
3.2.2. Reprogramar las piezas asignadas a la categoría alta ($J_M = J_M-\{j_m\}$).
3.2.3. Reprogramar las piezas asignadas a la categoría media ($J_L = J_L + \{j_m\}$).
3.2.4. Determinar la c_{max} global de la solución candidata (c_{max}^c).
3.2.5. SI $c_{max}^c > c_{max}^s \rightarrow$ FIN
En caso contrario, añadir la solución candidata al conjunto de soluciones (s = s +1):
$\rho_{s+1} = (c_{max}^{s+1} = c_{max}^c, w^{s+1} = w^s + 1)$
fin MIENTRAS

4.3 PUNTOS FUERTES Y PUNTOS DÉBILES DE LA HEURÍSTICA

Explicada la heurística, se cree conveniente realizar un análisis de qué puntos fuertes y débiles presenta.

Puntos fuertes

- Sencillez de la heurística: se trata de un método muy sencillo, fácil de comprender y de programar.

Puntos débiles

- Orden estricto de los tres movimiento de piezas entre categorías.
- Rigidez del orden de reasignaciones de la heurística: Una vez realizados los tres movimientos propuestos, sería recomendable seguir realizando la Fase 2 hasta que no hubiese mejora en las soluciones obtenidas.
- La penalización aumenta rápidamente: los movimientos con $w^{s+1}=w^{s+2}$ no se realizan en el tramo final del algoritmo y eso provoca que enseguida aparezcan soluciones con una alta penalización.

5. PROCEDIMIENTO GRASP

Un GRASP (*Greedy Randomized Adaptive Search Procedure*) consiste en realizar varias ejecuciones de una heurística sobre una instancia, introduciendo un componente aleatorio para ampliar el número de soluciones exploradas. Generalmente se aplica en la optimización de problemas combinatorios. Este procedimiento presenta las siguientes características:

- Buscar M soluciones en cada iteración: en este caso ya no se trabajará sólo con la mejor solución posible de entre las exploradas.
- En cada iteración se establece la lista de M candidatas, como se ha hecho hasta ahora en la heurística. La diferencia es que ahora no se seleccionará la solución que sea la mejor según algún criterio común (mayor reducción de makespan, por ejemplo), sino una al azar siguiendo una lista restringida.
- Finalmente se elegirá una solución no dominada de manera aleatoria entre las M soluciones obtenidas.

Introduciendo este componente de aleatoriedad en las soluciones obtenidas se puede llegar a obtener un mayor número de soluciones y de mejor calidad debido a que se exploran otras soluciones vecinas que con el procedimiento heurístico nunca se tratarían.

5.1 PROCEDIMIENTO PROPUESTO

Las características concretas para aplicar el GRASP en el problema presentado son:

- La heurística 1 determina la pieza que cambiará de máquina y categoría de entre todas las piezas que por su complejidad y la máquina donde se encuentran pueden cambiar de categoría($j_m \in JC$) por medio del parámetro $\max_{j \in JC}(p_j)$. Esta vez , en vez de elegir sólo una pieza, se elegirá entre varias piezas con mayor p_j. De entre las piezas escogidas, la finalmente candidata a cambiar de nivel ($j_m \in JC$) se escogerá de forma aleatoria.
- Para explorar el mayor número de soluciones posibles, se aplicará la heurística anterior un elevado número de veces para obtener una mayor variedad de soluciones. Posteriormente se analizarán las soluciones obtenidas en todas las iteraciones y se tomarán las que sean no-dominadas.

Después de varias experimentaciones, se ha decidido 500 iteraciones y elegir aleatoriamente entre 4 piezas.

6. EXPERIENCIA COMPUTACIONAL

Con el fin de automatizar los cálculos, se programó el procedimiento expuesto anteriormente mediante el Microsoft Visual Basic for applications. Se han creado los ejemplares utilizados para la obtención de resultados y también se ha desarrollado un programa que compara las soluciones obtenidas con ambos procedimientos.

6.1 GENERACIÓN DE LOS EJEMPLOS

Los ejemplos han sido generados de manera similar a Gharbi y Haouari (2002) y los que Garriga (2009) utilizó para su Proyecto Fin de Carrera. Se trata de 1000 ejemplares de 20 y 50 piezas y 200 ejemplares de 100 y 200 piezas. Se ha considerado que con 200 ejemplares los resultados obtenidos son significativos. En cuanto a los diferentes tiempos de producción, se han generado mediante el uso de distribuciones discretas uniformes como se muestra en la Tabla 1: *n* es el número de piezas, *m* es el número de máquinas y *K* es un entero positivo (según Gharbi y Haouari, 2002) tal que, K=3 para la primera mitad de los ejemplares y K=5 para el resto de ejemplares. El parámetro *K* afecta, por tanto, las distribuciones de los tiempos de preproceso y de postproceso.

Tabla 1: Distribución de las piezas según la complejidad en un juego

Tiempos de producción	Tiempos de pre-proceso (r_j)	Tiempos de proceso (p_j)	Tiempo de post-proceso (q_j)
Distribución discreta uniforme	[1, k(n/m)]	[1,10]	[1,k(n/m)]

Finalmente, hay que considerar cuál será la proporción de piezas para cada complejidad en cada uno de los juegos de piezas generados. En la Tabla 2 se muestra los diferentes porcentajes establecidos para las piezas de las diversas complejidades. Estos valores también se generarán mediante el uso de distribuciones discretas uniformes.

Tabla 2: Distribución de las piezas según la complejidad en un juego

Complejidad	Alta (h)	Media (m)	Baja (l)
Proporción (%)	20-30	20-50	20-60

El número de máquinas (m) es de 3, 4, 5, 6, 8 y 10. Cada una de estas máquinas puede pertenecer a una categoría (k) alta, media o baja. Las diferentes distribuciones que se han estudiado de las máquinas indican en la Tabla 3.

Tabla 3: Distribución de las máquinas

	3A	4A	4B	4C	5A	5B	5C	6A	6B	6C	6D	6E	6F
$\mathbf{m_h}$	1	2	1	1	1	2	2	3	3	2	2	1	1
$\mathbf{m_m}$	1	1	2	1	2	1	2	2	1	3	1	3	2
$\mathbf{m_l}$	1	1	1	2	2	2	1	1	2	1	3	2	3
	6G	8A	8B	8C	8D	8E	8F	10A	10B	10C	10D	10E	10F
$\mathbf{m_h}$	2	4	2	2	2	3	3	4	3	3	2	4	4
$\mathbf{m_m}$	2	2	4	2	3	2	3	3	4	3	4	2	4
$\mathbf{m_l}$	2	2	2	4	3	3	2	3	3	4	4	4	2

Para poder comparar de manera adecuada los ejemplos, se crean unas piezas con sus características correspondientes y, estas piezas se utilizan para las diversas combinaciones de máquinas definidas anteriormente.

Para los casos con 20 piezas, se han descartado los estudios con 8 y 10 máquinas y para los casos con 50 piezas se han descartado los estudios con 10 máquinas. Se obtiene así que se han realizado aproximadamente 45.000 ejecuciones para cada una de las heurísticas planteadas.

6.2 PRESENTACIÓN DE LOS RESULTADOS

6.2.1 ESTUDIO DE LA DIFERENCIA DE C_{MAX} EXTREMAS

Las siguientes tablas presentan los resultados de aplicar la heurística presentada en la Sección 4 y el GRASP de la Sección 5 a ejemplares de 20, 50, 100 y 200 piezas . En primer lugar se ha realizado un estudio de la diferencia

que proporciona en el valor de c_{max} de sus soluciones extremas (v=0 y v=s) cada heurística en función del número de piezas por ejemplar y del número de máquinas.

Los valores se presentan en forma porcentual y no en unidades de tiempo debido a que la cantidad de piezas influye considerablemente en esta variación. La diferencia de la c_{max} se calcula como muestra la ecuación (1):

$$\frac{c^0_{max} - c^s_{max}}{c^0_{max}} \quad (1)$$

Un elevado porcentaje en diferencia de c_{max} indica un abanico más amplio de tiempo en las soluciones obtenidas para cada ejemplar, es decir, mayor flexibilidad para decidir la programación del trabajo al proceso productivo.

Estudio en función del número de piezas

La Tabla 4 muestra el resultado del estudio en función del número de piezas después de la aplicación de los procedimientos en todas las combinaciones de máquinas mostradas en la Tabla 4. En la primera columna está el número de piezas que tienen los ejemplares tratados. En las dos columnas siguientes se indica se indica el porcentaje de mejora de la c_{max} obtenida con la heurística y el GRASP.

Tabla 4: Diferencia de la c_{max} extremas según el número de piezas

n	Heurística	GRASP
20	39,40%	45,01%
50	38,25%	45,56%
100	37,08%	50,13%
200	37,21%	51,08%

Se puede comprobar que el número de piezas no conlleva una variación significativa en la diferencia de la c_{max}. Esto puede ser debido a que en esta heurística se aplica para todas las soluciones el procedimiento de Gharbi y Haouari (2002) que reparte la carga de las máquinas de una misma categoría. En términos generales, se puede decir que la diferencia de la $c_{max,}$ para el caso de la heurística, se encuentra entre un 37% y un 40% desde la primera solución en la que todas las piezas se encuentran en la categoría alta hasta la última solución. También se observa que los resultados aplicando el procedimiento GRASP alcanzan el 50% de media a mayor número de piezas.

Estudio según número de máquinas

La Tabla 5 muestra el resultado del estudio en función del número de máquinas después de la aplicación de los diferentes procedimientos. Con estos resultados se quiere comprobar si hay algún procedimiento que sea especialmente eficiente, para un número de máquinas determinado.

Observando los datos de la Tabla 5 se puede comprobar que a medida que aumenta el número de máquinas, el porcentaje de diferencia de c_{max} extremas disminuye en el caso de la heurística. Esto puede ser debido a que cuando más máquinas hay, al equilibrarlas mediante el procedimiento de Gharbi y Haouari (2002), se plantean más opciones que dos categorías tengan el mismo valor de c_{max} y provoqueo así el final de la heurística.

Tabla 5: Diferencia de la c_{max} extremas según el número de máquinas

m	Heurística	GRASP
3	42,82%	48,44%
4	41,75%	48,87%
5	40,09%	48,79%
6	39,06%	47,12%
8	35,78%	48,94%
10	28,07%	50,47%

En cuanto a los resultados del GRASP, cabe destacar que no se cumple que al aumentar el número de máquinas disminuya la diferencia de c_{max} extremas. Esto se debe a que con este procedimiento se exploran muchas soluciones vecinas diferentes y conlleva estudiar muchas vías diferentes. Para 10 máquinas, eto puede ser especialmente útil.

6.2.2 NÚMERO DE SOLUCIONES OBTENIDAS

Obtener un número de soluciones elevado puede ser importante ya que da una mayor variedad al encargado que toma la decisión de determinar cuál es la programación más adecuada en cada caso. Para algunos ejemplares, puede ser más importante tener muchas soluciones donde poder escoger entre c_{max} y w_{tot} diferentes para poder realizar la programación ajustándose lo máximo posible a restricciones impuestas por la dirección de la empresa como: un período de entrega muy estricto o un costo de producción determinado.

Estudio en función del número de piezas

En la Tabla 6 se observa cuál es el número soluciones medio para cada uno de los procedimientos según el número de piezas que tengan los ejemplares. De acuerdo con el análisis de la Tabla 6 se puede concluir que al aumentar el número de piezas por ejemplar aumentan también el número de soluciones obtenidas. Este hecho se podría deducir sin necesidad de los resultados, ya que al aumentar el número de piezas aumenta el número de piezas candidatas para realizar intercambios entre máquinas. Observando que el número de soluciones cambia tanto según el número de piezas, no se considera necesario realizar un estudio del número de soluciones según las máquinas.

Tabla 6: Número de soluciones según el número de piezas procesadas

n	Heurística	GRASP
20	7,76	9,74
50	15,33	19,73
100	27,42	39,58
200	54,19	77,04

Para cada solución obtenida, el peso global (w_{tot}) de la solución puede aumentar en 1 ó 2 unidades según el cambio de categoría realizado. Conocer el número de soluciones da una idea muy aproximada del valor que tendrá w_{tot}, ya que será muy cercano o igual al número de soluciones. Por lo tanto, un número elevado de soluciones implica también un amplio rango de costos en el que se puede mover la empresa para realizar los ejemplares , esto da una mayor flexibilidad e información a los encargados de decidir la programación del sistema productivo.

En cuanto a la metaheurística propuesta, el hecho de contar con más soluciones implica que el encargado de determinar la programación de la línea de producción tiene más variedad para elegir diferente programaciones con c_{max} y penalización diferentes. Es decir ofrece una mayor flexibilidad.

CONCLUSIONES

El problema tratado en este trabajo es la programación de un conjunto de piezas con tiempo de pre-proceso, tiempo de proceso y tiempo de post-proceso y clasificadas en 3 categorías en un conjunto de máquinas en paralelo clasificadas también en las mismas categorías o niveles. Los objetivos considerados simultáneamente han sido los de minimizar el instante de finalización de la última pieza (c_{max}) y la penalización total asociada a cambiar de nivel piezas (w_{tot}), que se encuentra relacionada con el costo que implica realizar una pieza en una máquina con tecnología menos eficiente. Se considera que las piezas sólo se pueden realizar en máquinas que tienen una categoría mayor o igual que la suya. La primera de las soluciones es aquella en que todas las piezas se programan en las máquinas con una categoría alta, ya que en esta situación la penalización es 0 (todas las piezas se fabrican con la mejor tecnología posible).

En este documento se propuso una heurística de reasignación por criterios de tiempo para obtener soluciones factibles para el problema planteado. Dicha heurística presenta unos resultados eficientes. La diferencia de las c_{max}

extremas es de un 38% aproximadamente. En cuanto al número medio de soluciones por ejemplar se han obtenido desde 8 soluciones de media para ejemplares con 20 piezas hasta 54 para ejemplares de 200 piezas, pasando por las 15 soluciones para ejemplares de 50 piezas y las 27 para los de 100. Posteriormente, se aplica un GRASP basado en la heurística propuesta para mejorar la eficiencia de las soluciones obtenidas. La diferencia de las c_{max} extremas alcanza entonces en media un 48% aproximadamente. En cuanto al número medio de soluciones por ejemplar, se han obtenido: 10 soluciones de media para ejemplares con 20 piezas, 20 soluciones para ejemplares de 50; 40 soluciones para ejemplares de 100 y 77 soluciones para ejemplares de 200.

Como futuras líneas de investigación se recomienda la incorporación de tiempo de preparación de las máquinas para cada tipo de pieza, así como la consideración de tiempos de proceso diferente de cada pieza en cada tipo de máquina, ya que normalmente en una planta real no se cumple el supuesto que todas las piezas tengan la misma velocidad de producción en todas las máquinas.

REFERENCIAS

Blazewicz J., Ecker K.H., Pesch E., Schmidt G. y J. Weglarz. (2001). Scheduling Computer and Manufacturing Processes, Springer-Verlag.

Carlier J. (1987). Scheduling jobs with release dates and tails on identical machines to minimize the makespan. *European Journal of Operational Research*; No. 29, p. 298 –306.

Centeno, G. y Armacost, R. (1997). Parallel Machine Scheduling with Release Time and Machine Eligibility Restrictions, *Computers and Industrial Engineering*, Vol. 33.

Centeno, G. y Armacost, R. (2004). Parallel Machine Scheduling with Release Dates, Due dates, and Machine Eligibility Restrictions for Minimizing Makespan, *International Journal of Production Research*, Vol. 42, No. 6, pp.1243-1256.

Dessouky M.M.(1998). Scheduling identical jobs with unequal ready times on uniform parallel machines to minimize the maximum lateness. *Computers and Industrial Engineering*, No. 34, pp. 793–806.

Ehrgott, M. y Gandibleux, X. (2002). Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys. *International Series in Operations Research and Management Science* : Volume 52.

Garriga, X.(2009). Programació de peces a múltiples nivells en un taller de línies de premses. Barcelona, Projecte de Fi de Carrera ETSEIB-UPC.

Gharbi, A., Haouari, M. (2002). Minimizimg makespan on parallel machines subject to release dates and delivery times. *Journal of Scheduling*. No. 5, pp. 329-355.

Johnson, S. M. (1954). Optimal two and three stage production schedules with setup times included. *Naval research Logistics Quarterly*, Vol 1, No. 4, pp. 281.

Lancia, G. (2000). Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan. *European Journal of Operational Research*, No. 120, pp. 277–288.

Leung, J. Y.-T. and C.-L. Li (2008). Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, No. 116, pp. 251-262.

Loukil, T., Teghem, J. and P. Fortemps,(2007). A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research*, No. 179 (3), pp. 709-722.

Pinedo, M.L., (2002). *Scheduling theory, algorithms, and systems*. Prentice-Hall.