Adaptación de heurísticas para la secuenciación de piezas en una máquina al problema de secuenciación en máquinas en paralelo

Imma Ribas Vila, Ramon Companys

Dpto. de Organización de empresas. Escuela Técnica Superior de Ingenieros Industriales de Barcelona. Universidad Politécnica de Cataluña. Avda. Diagonal, 647. 08028 Barcelona. imma.ribas@upc.edu, ramon.companys@upc.edu

Resumen

En esta comunicación se propone un procedimiento de programación de piezas en un sistema formado por máquinas en paralelo con tiempos de preparación dependientes de la secuencia y una nueva clasificación para los problemas de máquinas en paralelo que tiene en cuenta la existencia de tiempos de preparación. El procedimiento de programación propuesto permite adaptar fácilmente las heurísticas diseñadas para la programación de piezas en un sistema formado por una única máquina con tiempos de preparación dependientes de la secuencia al problema de secuenciación en un sistema con máquinas en paralelo con tiempos de preparación dependientes de la secuencia. Para comprobar la factibilidad del procedimiento propuesto se han implementado dos heurística: la primera es una adaptación de una heurística diseñada para el caso de una máquina y la segunda se ha diseñado para el problema en cuestión. A través de la experiencia computacional se ha comprobado que el procedimiento propuesto es muy eficiente.

Palabras clave: Secuenciación, máquinas en paralelo, tiempos de preparación, retraso total

1. Introducción

El objeto del trabajo es presentar un procedimiento de determinación de programas que permite tratar la secuenciación de piezas en un sistema formado por máquinas en paralelo con tiempos de preparación dependientes de la secuencia mediante heurísticas similares a las utilizadas en la secuenciación de piezas en un sistema formado por una única máquina, en Ribas (2007), también con tiempos de preparación dependientes de la secuencia. La aplicación de este procedimiento, que no explora la totalidad del dominio de los programas factible, permite, en cambio, adaptar fácilmente heurísticas eficientes conocidas para el caso de una máquina.

La literatura, usualmente, al establecer la tipología de máquinas las clasifica en iguales, uniformes o diferentes, pero al tener en cuenta los tiempos de preparación esta clasificación puede ser confusa y por ello proponemos una nueva clasificación.

Cada máquina puede procesar un único trabajo a la vez y cada trabajo es procesado por una única máquina. Una vez un trabajo es iniciado debe finalizar sin interrupción. Consideramos que las piezas a fabricar pertenecen a diferentes familias y existe únicamente tiempos de preparación cuando la secuencia de piezas a fabricar implica un cambio de familia. Esta hipótesis no introduce ninguna restricción adicional.

El objetivo buscado en la secuenciación de las piezas es la minimización del retraso medio, o, en forma equivalente, la minimización de la suma de los retrasos de las piezas, siendo el retraso la diferencia entre el instante de finalización de la pieza y su fecha de vencimiento. De acuerdo con Koulamas (1994) este problema es como mínimo NP-hard ya que Du y Leung (1990) demostraron que el problema con una única máquina lo es.

2. Definición e hipótesis del problema tratado.

Se consideran *n* piezas, de una sola operación cada una, que pueden procesarse en *m* máquinas idénticas. Las máquinas no pueden procesar más de una pieza a la vez y una vez iniciada la operación no se interrumpe hasta que haya finalizado.

Cada pieza i tiene asociado un tiempo de proceso p_i, una fecha de vencimiento d_i, una familia $g(i)$ a la que pertenece y está disponible en el instante r_i para iniciar su operación (la preparación de la misma, si la hubiera, puede haber empezado antes); generalmente $r_i = 0$.

El número de familias es $q \leq n$, y $s_{h,g(i)}$ es el tiempo de preparación cuando se realiza una pieza de la familia $g(i)$ después de una de la familia *h*; si dos piezas sucesivas son de la misma familia el tiempo de preparación intermedio es nulo. Se supone que los tiempos de cambio cumplen la desigualdad triangular. Se conoce además, el instante, τ_j, en que cada máquina queda libre (o disponible) para realizar las operaciones sobre las *n* piezas, estos instantes pueden ser diferentes para cada máquina y así como el estado inicial de cada máquina, φ_j, que corresponde a una familia, en principio la familia de la última pieza procesada por esta.

Dado una secuencia de las piezas realizadas por cada máquina, en un orden concreto, podemos asociarle un programa semiactivo (sin máquinas ociosas cuando existan preparaciones u operaciones disponibles para realizar) y calcular las fechas correspondientes de inicio y fin de las operaciones: En lo sucesivo consideraremos sinónimo secuencia para cada máquina (que más adelante llamaremos pelotón) y programa. Siendo c_i el instante en que una pieza termina su proceso, el retraso T_i viene dado por:

$$T_i = \max \{0, c_i - d_i\} \tag{1}$$

El objetivo es encontrar una secuencia de las piezas que minimice la suma del retraso de éstas:

$$[MIN\ ]Z = \sum_{i=1}^{n} T_i \tag{2}$$

Este problema es conocido en la literatura, siguiendo la nomenclatura propuesta por Lawler et al. (1993), como $Pm\,|s_{j,k}|\Sigma T$.

En virtud de lo anterior, consideramos que un programa, en este caso, se compone de los siguientes elementos,

- una asignación de cada pieza a una máquina,
- una secuencia o permutación de las piezas asignadas en cada máquina,
- un calendario de realización (o tempística).

3. Clasificación de problemas con máquinas en paralelo considerando tiempos de preparación.

Según la clasificación propuesta en la literatura las máquinas disponibles en cada etapa, pueden ser: idénticas, proporcionales o distintas. Dos máquina son idénticas si para todo trabajo el

tiempo de fabricación es el mismo en cualquiera de ellas, son proporcionales si cada máquina tiene asociado un valor *v(j)* de forma que si *p(j,i)* es el tiempo de proceso de la operación *i* en la máquina *j* se cumple que $p(k,i) = \frac{p(i,j) * v(j)}{v(k)}$, y son diferentes o no están relacionadas cuando el tiempo de proceso varia en función de la máquina utilizada. Esta clasificación es confusa cuando se tienen en cuenta los tiempos de preparación ya que dos máquinas idénticas pueden requerir un tiempo de preparación diferente debido, por ejemplo, a estructuras o diseños diferentes, mientras que dos máquinas distintas pueden requerir el mismo tiempo de preparación. Por lo tanto, una clasificación más ajustada sería aquella que combinara las diferentes posibilidades que se pueden dar con respecto al tiempo de proceso y al tiempo de preparación. Tal y como se ha comentado anteriormente, el tiempo de proceso puede ser idéntico, proporcional o diferente. Esta misma clasificación puede ser válida para el tiempo de preparación aunque con el fin de simplificar, se considerará únicamente que los tiempos de preparación para las máquinas de una misma etapa pueden ser iguales o diferentes. En esta línea, se propone la siguiente clasificación:

Diremos que dos máquinas de una misma etapa son *idénticas* cuando el tiempo de proceso y el tiempo de preparación son iguales, son *uniformes* cuando el tiempo de proceso es proporcional pero su tiempo de preparación es igual y son *distintas* cuando el tiempo de proceso es diferente pero los tiempos de preparación son iguales. En cambio, diremos que dos máquinas de una misma etapa son *parecidas* cuando el tiempo de proceso es igual pero los tiempos de preparación son diferentes, son *relacionadas* cuando los tiempos de proceso son proporcionales pero los tiempos de preparación son diferentes y son *arbitrarias* cuando tanto el tiempo de proceso como el tiempo de preparación son diferentes.

Tabla 1. Clasificación de máquinas considerando tiempos de preparación

Tiempo preparación	Tiempo de proceso		
	Igual	Proporcional	Diferente
Igual	Idénticas	Uniformes	Distintas
Diferente	Parecidas	Relacionadas	Arbitrarias

4. Procedimiento de programación para máquinas en paralelo y tiempos de preparación dependientes de la secuencia

4.1. Nomenclatura

Llamamos *fila*, a un conjunto ordenado de *k* piezas o trabajos, con $0 \le k \le n$. Una fila puede ser vacía ($k = 0$). Dos filas formadas por las mismas *k* piezas son diferentes si el orden o secuencia es diferente.

Llamamos *pelotón* (π) a un grupo ordenado de *m* filas sin piezas comunes que contienen las *n* piezas. Un pelotón es equivalente a una asignación más una secuenciación de todas las piezas a programar. Al conjunto de todos los pelotones lo denominamos Π.

Dado un *pelotón* y las condiciones iniciales de las máquinas podemos determinar un calendario de realización (y eventualmente el retraso global).

Proposición 1: Un pelotón está asociado biunívocamente a un programa semiactivo.

Llamamos *sarta* (σ) a una permutación de las *n* piezas. Al conjunto de sartas lo denominamos

Σ. Existen n! sartas, $|\Sigma| = n!$.

Llamamos Π_1 a la imagen de Σ que proporciona el algoritmo *grinder* implementado**.**

4.2. Procedimiento de programación

A partir de una *sarta* σ cualquiera, obtenida al azar o construida mediante otro procedimiento podemos generar un pelotón mediante un algoritmo adecuado que llamamos *grinder*. Por lo tanto, *grinder*(σ) = π.

Se pueden definir diferentes algoritmos *grinder* y la elección del algoritmo a implementar puede depender del criterio de eficiencia considerado. En este trabajo la implementación utilizada es la que muestra el Algoritmo 1.

Algoritmo 1: Transforma una sarta σ en un pelotón π. Para construir un pelotón se procede a la asignación, secuenciación y temporización de las piezas progresivamente, una después de otra. Para ello,

paso 1: Hacer $k = 1$ y $f_j = \tau_j$ (instante de disponibilidad inicial de la máquina j, tal vez 0) para $j = 1, 2, ..., m$,

paso 2: Sea k el valor en curso, corresponde asignar la pieza [k] que ocupa la posición k en σ. Calcular c_j el instante de terminación de la pieza [k] si fuera asignada a la máquina j, $j = 1, 2, \ldots, m$

paso 3: Determinar c-min = $\min_j \{c_j\}$; sea $\overline{j}$ la máquina que proporciona c-min (en caso de empate se elige la de menor j),

paso 4: Asignar [k] a la máquina $\overline{j}$ y actualizar los parámetros de dicha máquina; hacer $k = k+1$, si $k > m$ se han programado todas las piezas, en caso contrario volver al **paso 2**

Algoritmo 1. Transforma una sarta en pelotón

De esta forma, dada una sarta y un algoritmo *grinder* podemos asociar un valor de la función objetivo, por ejemplo del retraso total. Sería deseable que existiese la función inversa $(grinder)^{-1}$ ya que la relación entre la *sarta* y el *pelotón* sería biunívoca y, por lo tanto, se podría trabajar con sartas o pelotones indistintamente.

La aplicación *grinder* no es biyectiva ya que $|\Pi_1| \leq |\Sigma| < |\Pi|$ y, en consecuencia, no existe la función $(grinder)^{-1}$. No obstante, puede ser de utilidad tener un algoritmo que dado un pelotón proporcione una sarta, aunque éste no se corresponda con la funciones inversa del algoritmos *grinder*. A este algoritmo le llamaremos *Ungrinder*. Al igual que en los algoritmos *grinder*, los algoritmos *ungrinder* no tienen una definición única. A continuación se presenta una de las definiciones posibles.

Ungrinder: Transforma un pelotón π en una sarta σ. Para construir la sarta se procede a la secuenciación de las piezas progresivamente, una después de otra y para ello,

> **paso 1**: Hacer i=1, $k = 1$ para cada j_k (posición k de la máquina j) y $f_j = \tau_j$ (instante de disponibilidad inicial de la máquina j, tal vez 0) para $j = 1, 2, ..., m$,
>
> **paso 2**: Sea k el valor en curso de cada máquina j, si $j_k > 0$, determinar c-min = $\min_j \{c_j\}$;

sea $\overline{j}$ la máquina que proporciona *c*-min (en caso de empate se elige la de menor *j*).

paso 3: Asignar [*k*] de la máquina $\overline{j}$ en la posición *i* de la sarta y actualizar los parámetros de dicha máquina; hacer *i* = *i*+1, si i > *N* se han programado todas las piezas, en caso contrario volver al **paso 2**

Algoritmo 2. Transforma un pelotón π en una sarta σ

De esta forma podemos concluir que $sarta \cdot grinder = pelotón$ y que $pelotón \cdot ungrinder = sarta'$.

Podemos elegir una *sarta* σ cualquiera, obtenida al azar o construida mediante otro procedimiento. Utilizando el *grinder* adecuado, le corresponderá un *pelotón*. Por tanto dada una *sarta* y un *grinder* podemos obtener un valor de la función objetivo, por ejemplo del retraso total. Intuitivamente $R = \Phi(\sigma)$ donde σ es la sarta.
El vecindario de σ es fácil de generar. Por tanto podemos aplicar heurísticas de mejora eficientes conocidas para el caso de una máquina con lo que se explora el subconjunto de Π_1.

Tabla 2. Procedimientos heurísticos implementados

	Soluciones iniciales		
Procedimientos de Mejora	S1	S2	S3
PM1	H1	H2	H3
PM2	H4	H5	H6

Las heurísticas implementadas se dividen en dos partes: un procedimiento para obtener una solución inicial, y un procedimiento de optimización local. Se han implementado nueve heurísticas (Tabla 2) que son combinación de tres procedimientos de obtención de una solución inicial (S1, S2, S3) y dos heurísticas de mejora (PM1y PM2).

Un ejemplar se describe mediante:

– el número de piezas n, el número de familias q,una matriz cuadrada (de dimensión q) de tiempos de preparación S, el número de máquinas en paralelo m, la familia de la última pieza procesada en cada una de las máquinas φ_j y una terna de datos para cada pieza compuesta por el tiempo de proceso $p_i \ (i = 1,...,n)$, la fecha de entrega $d_i \ (i = 1,...,n)$ y la familia a la que pertenece la pieza $g(i) \in \{1,2,...,q\}$, $i = 1,2,...,n$.

4.2.1 Procedimiento S1

Se calcula de forma dinámica un índice crítico que tiene en cuenta la combinación de pieza y máquina. El índice se utiliza como criterio de asignación de las piezas a las máquinas de forma que se asigna la pieza a la máquina con cuya combinación se obtenga el índice de menor valor.

Para cada pieza y máquina se calcula $CR_{j,i.}$ según la expresión (3).

$$CR_{j,i}(t) = \alpha \cdot d_i + (1-\alpha) \cdot \left(\max\left\{\tau_j + s_{h,g(i)}, r_i\right\} + p_i\right) \qquad (3)$$

donde α puede tomar cualquier valor entre 0 y 1; d_i es la fecha de entrega de la pieza i, p_i es el tiempo de operación de la pieza i, $s_{h,g(i)}$ es el tiempo de preparación para realizar la familia

$g(i)$ que depende de la familia de la pieza anterior h, τ_j es el instante en que la máquina j queda libre y r_i es el instante de disponibilidad de la pieza.

El parámetro α permite al planificador fijar su política según quiera prioriza las piezas que tengan un plazo de entrega más crítico o un tiempo de proceso menor.

Para cada pieza se retiene el valor de $\overline{CR_i}$ calculado según (4) y se asigna a la máquina que proporciona dicho valor, actualizando su instante de disponibilidad y la familia para la que queda preparada, si la pieza asignada pertenece a una familia diferente a la que estaba preparada la máquina.

$$\overline{CR_i} = \min \{CR_{j,i}\} \tag{4}$$

Se procede de esta forma hasta haber generado un pelotón. Posteriormente, se calcula la temporización de las piezas y su retraso asociado.

4.2.2 Procedimiento S2

Se calcula para cada pieza su $\overline{CR_i}$ asociado según (4). A continuación se calcula el valor y se toma $\overline{CR}_{\min} = \min \overline{CR_i}$ como valor umbral el resultado de (5).

$$Umbral = (1 + \beta) \cdot \overline{CR}_{\min} \tag{5}$$

En la implementación realizada b=0,2. Posteriormente se elige al azar una de las piezas cuyo valor $\overline{CR_i} \leq umbral$ y se asigna a la máquina que ha proporcionado dicho valor. El procedimiento continua hasta que se ha generado todo el pelotón.

4.2.3 Procedimiento S3

Este procedimiento genera una secuencia de fabricación inicial (sarta) al azar. Para ellos se construye una secuencia a través de un vector auxiliar, *aux*, que inicialmente contiene las *n* piezas ordenadas según numeración creciente. A continuación se barajan las piezas de forma aleatoria y se crea la secuencia inicial asignando a la posición *i* de la secuencia el valor de *aux(i)*. Para obtener el pelotón asociado y valorar el retraso global se aplica el algoritmo *grinder*.

4.2.4 Procedimiento PM1

Este procedimiento es una variante del algoritmo no exhaustivo de descenso (ANED). Esta variante incorpora dos herramientas que permiten hacer frente a dos problemas existentes en el espacio de las soluciones: La existencia de mesetas, zonas en las que el valor de la función objetivo es la misma, y el orden en que se explora el vecindario. Para el primer inconveniente se aceptan empates con cierta probabilidad y para el segundo, se incorpora un vector de posiciones (que llamamos técnica revolver) que permite recorrer el vecindario de forma aleatoria.

4.2.5 Procedimiento PM2

Se ha querido estudiar la eficiencia de un procedimiento de mejora que actúe sobre el vecindario del pelotón inicial. En este caso los movimientos de las piezas mantienen la estructura del pelotón. Un vecino se puede generar a través de diferentes movimientos de las piezas. Estos movimientos se pueden producir dentro de la máquina a la que está asignada o entre máquinas.

El procedimiento de mejora PM2 combina tres movimientos. Inicialmente se aplica el procedimiento PM1 sobre cada una de las máquinas con el fin de encontrar la mejor secuencia dentro de cada máquina. A continuación se aplica un procedimiento de intercambio de una pieza de una máquina con todas las demás piezas de las demás máquinas y, si éste produce algún cambio, se aplica de nuevo la mejora PM1 sobre cada máquina. Finalmente se aplica un procedimiento que estudia la inserción de una pieza de una máquina en cualquier posición de las demás máquinas y, de nuevo, si éste produce cambios, se aplica la mejora PM1 sobre cada máquina.

4.2.6 Experiencia computacional

Se ha construido una colección de ejemplares dividida en 4 grupos de 100 ejemplares cada uno. Cada grupo es una combinación de 15 y 20 piezas que se deben secuenciar en 3 o 4 máquinas diferentes. Las piezas pertenecen a una de las 4 familias activas.

El generador de ejemplares se basa en tres parámetros $\lambda_1, \lambda_2, \lambda_3$ usados para calcular S_{max}, d_{max} y d_{min}, respectivamente. El tiempo de proceso de las piezas sigue una distribución uniforme entre $[1, P_{max}]$, con P_{max}=25; el tiempo de preparación está uniformemente distribuido entre $[1, S_{max}]$, donde $S_{\max} = \lambda_1 P_{\max}$; las fechas de vencimiento siguen una distribución uniforme entre $[d_{min}, d_{max}]$, con d_{max}-$\lambda_2 fv$, d_{min}-$\lambda_3 fv$ y $fv = \frac{1}{m}\sum_{i=1}^{n} p_i + \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} s_j}{q}$ para cada ejemplar. Una vez fijados *n* y *q*, se elige al azar, entre 1 y *q*, la familia para la que está preparada la máquina y lo mismo se hace para fijar la familia de las diferentes piezas. En este conjunto de ejemplares se han fijado los siguientes valores: $\lambda_1 = 0.5, \lambda_2 = 1$ y $\lambda_3 = 0.3$. Una vez creado un ejemplar se aplica una heurística sencilla que permita descartarlo si su retraso es cero.

Todos los test realizados se han hecho en un Pentium IV con 512 MB RAM y un procesador de 2.8 GHz.

A través de esta experiencia computacional se evalúan los diferentes procedimientos elaborados teniendo en cuenta el efecto producido por la variación del número de piezas y del número de máquinas.

En la Tabla 3 se muestra los tiempos medios, en segundos, para resolver un ejemplar usando cada uno de los procedimientos. En ella se puede observar que el tiempo que requieren las heurísticas para obtener una solución aumenta cuando el número de piezas a programar es mayor. Cabe destacar que las heurísticas que requieren un tiempo de cómputo son las que utilizan el procedimiento de mejora PM1 que actúa sobre el vecindario de la sarta.

Tabla 3. Tiempos medios, en segundos, de resolución de un ejemplar

Nº Maquinas	m=3		m=4	
Nº Piezas	n=15	n=20	n=15	n=20
Heurística H1	0.10	0.17	0.10	0.18
Heurística H2	0.12	0.25	0.11	0.23
Heurística H3	0.13	0.28	0.12	0.24
Heurística H4	0.06	0.14	0.09	0.15
Heurística H5	0.08	0.18	0.09	0.17
Heurística H6	0.09	0.17	0.10	0.19

Para determinar que procedimiento de los implementados es el más adecuado a esta tipología de problemas se ha optado por comparar entre sí las heurísticas. La comparación se ha llevado a cabo concediendo a las heurísticas el mismo tiempo de ejecución permitiendo así que puedan ejecutarse varias veces y explorar vecindarios, que debido a la resolución de empates pueden haberse descartado con anterioridad. Dado que el tiempo de cómputo requerido por las heurísticas aumenta con el número de piezas a programar, siendo más o menos el mismo aunque se tengan 3 o 4 máquinas en paralelo, se ha fijado un tiempo de 15 minutos para las colecciones con 15 piezas y 30 minutos para las colecciones de 20 piezas.

Los resultados a comparar se han obtenido ejecutando, sobre los ejemplares de cada colección, los algoritmos implementados. Se han realizado tres réplicas por ejemplar y procedimiento. Para cada ejemplar, se ha calculado la discrepancia relativa del retraso medio respecto a la mejor solución obtenida a través de cualquiera de los procedimientos en estudio. Esta discrepancia relativa se calcula mediante el índice $I_{h,heurística}$ según (6), siendo *h* el número de ejemplar analizado y *heurística* el procedimiento utilizado para obtener el retraso acumulado.

$$I_{h,heurística} = (\mu_{h'heurística} - r_{h,min}) / r_{h,min} * 100 \quad (6)$$

donde $\mu_{h'heurística}$ es el retraso medio obtenido para el ejemplar *h* mediante el procedimiento *heurística* y $r_{h,min}$ es el retraso mínimo, para este ejemplar, obtenido con cualquiera de las heurísticas y réplicas. Se ha calculado, además, la media (μ) y la desviación estándar (σ) del índice $I_{h,heurística}$, cuyos valores de cada colección (Tabla 4), sirven para comparar los procedimientos entre si.

De la Tabla 4 se deduce que el procedimiento que proporciona mejores soluciones es la heurística H3 (combinación de los procedimientos S3 y PM1) ya que es la que tiene un valor de la media menor lo que indica que con esta heurística casi siempre se obtienen las mejores soluciones.

Tabla 4. Análisis de los resultados comparando la μ y σ de $I_{h,heurística}$ (en %).

Nª máquinas	m=3				m=4			
Nº piezas	n=15		n=20		n=15		n=20	
Heurísticas:	μ	σ	μ	σ	μ	σ	μ	σ
H1	7.12	16.21	17.80	25.06	3.54	10.43	11.62	20.36
H2	3.38	14.02	3.64	13.20	0.40	2.75	2.94	11.41
H3	0.01	0.08	0.10	0.50	0.00	0.00	0.05	0.28
H4	10.07	14.10	14.53	13.77	6.47	7.46	10.06	9.69
H5	8.69	13.33	10.20	11.79	6.96	8.65	8.63	11.74
H6	0.21	0.89	1.57	2.73	0.13	0.50	1.18	1.95

Se ha querido evaluar, además, la dispersión de los resultados en cada método para estudiar la estabilidad del procedimiento. Esta evaluación se ha realizado calculando el índice $D_{h,heurísitca}$ de la siguiente forma (7):

$$D_{h,heurísitca} = (retraso_{h,max} - retraso_{h,min}) / \mu_{h'heurística} * 100 \quad (7)$$

Donde $D_{h,heurísitca}$ es la dispersión de los resultados del ejemplar *h* obtenidos con el procedimiento *heurística*, calculado sobre las tres réplicas de cada ejemplar de una colección.

En la Tabla 5 se muestran la media y la desviación estándar de $D_{h,heurísitca}$ para cada procedimiento y colección. De ella se deduce que:

– Las heurísticas que utilizan el procedimiento de mejora PM1 son las que se comportan de forma más estable ya que la media del índice D es baja lo que significa que las soluciones obtenidas en las diferentes réplicas son similares.

– El índice D crece a medida que aumenta el número de máquinas aunque no lo hace respeto al número de piezas.

Tabla 5. Media y desviación estándar de D

Nª máquinas	m=3				m=4			
Nº piezas	n=15		n=20		n=15		n=20	
Heurísticas:	μ	σ	μ	σ	μ	σ	μ	σ
H1	0.18	1.34	0.00	0.00	0.45	2.73	0.09	0.53
H2	0.03	0.26	0.00	0.00	0.29	1.28	0.21	0.94
H3	**0.02**	**0.25**	**0.00**	**0.00**	**0.18**	**0.96**	**0.10**	**0.52**
H4	0.69	2.78	0.41	1.39	4.26	8.94	2.14	4.58
H5	1.79	5.07	1.01	2.73	4.67	13.11	3.62	6.93
H6	0.32	1.28	0.24	1.05	2.23	4.66	1.92	3.44

5. Conclusiones

Se presenta un procedimiento de programación que permite secuenciar piezas en máquinas en paralelo con tiempos de preparación dependientes de la secuencia mediante heurísticas similares a las utilizadas en el problema análogo con una única máquina. Para comprobar la eficiencia de este procedimiento se han implementado y comparado seis heurísticas donde los rasgos diferenciales son la mejora por exploración del vecindario de la sarta y la mejora por exploración del vecindario del pelotón. Hemos comprobado que a través de la exploración del vecindario de la sarta obtenemos mejores resultados. Hubiera sido interesante poder contrastar los resultados con la solución óptima pero carecemos de un algoritmo eficiente que nos la proporcione. Estamos trabajando en la adaptación de un algoritmo exacto diseñado para máquinas en paralelo sin tiempos de preparación, con el fin de poder obtener el óptimo de cada ejemplar. Nos proponemos adaptar nuevas heurísticas que sean eficientes para el caso de una máquina con el fin de estudiar su efectividad en el caso de máquinas en paralelo.

Se ha propuesto, además, una nueva nomenclatura para clasificar los problemas de máquinas en paralelo cuando se tienen en cuenta los tiempos de preparación, ya que la clasificación propuesta en la literatura no considera la existencia de dichos tiempos.

Referencias

Du, J., Leung, JYT. (1990). Minimizing total tardiness on one machine is NP-hard. Mathematics of Operations Research, Vol.15, pp.483-495.

Koulamas C. (1994). The total tardiness problem: review and extensions, Operational Research, Vol. 42, pp. 1025-1041.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B. (1993). Sequencing and scheduling: algorithms and complexity, in Graves, C. G., Rinnooy Kan, A. H. G., Zipkin, P. (eds) Handbooks in Operations Research and Management Science, 4: Logistics of Production

and Inventory, North-Holland, Amsterdam, pp. 445-522.

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. Universidad politécnica de Valencia.