DITS (2006/01).
Working Paper del Departament d'Organització D'empreses de la Universitat Politècnica de Catalunya.

Programación de la Producción en un sistema flow shop híbrido sin esperas y tiempos de preparación dependientes de la secuencia

Imma Ribas Vila[1], Ramon Companys Pascual[1]

[1] Departament d'Organització d'Empreses. Escola Tècnica Superior d'Enginyers Industrials de Barcelona. Avinguda Diagonal, 647. 08028 Barcelona. imma.ribas@upc.edu, ramon.companys@upc.edu

Resumen

Se estudia la programación de la producción en sistemas flow shop híbrido con tiempos de preparación dependientes de la secuencia de piezas a fabricar. Las piezas pueden pertenecer a diferentes familias y las máquinas requerirán un tiempo de preparación cada vez que se deba cambiar de familia. Se han desarrollado procedimientos heurísticos para el caso monocriterio en el que el objetivo buscado en la programación de la producción es la minimización del retraso medio, equivalente a minimizar la suma de retrasos de las piezas, y para el caso bicriterio en el que se tendrá en cuenta tanto la minimización de una función objetivo formada por la suma ponderada del retraso medio más la suma de los tiempos medios de proceso. Además se han adaptado los métodos implementados para trabajar bajo la restricción nowait.

Palabras clave: Secuenciación, tiempos de preparación, heurísticas

1. Introducción

El objeto de la investigación es la programación de la producción en sistemas productivos compuestos por varias etapas en serie cada una de las cuales puede constar de más de una estación o máquina en paralelo. Este tipo de sistemas es conocido, en la literatura, mediante el nombre de ***flow shop* híbrido o sistemas de flujo híbrido**. En cada etapa, las máquinas disponibles para realizar una operación pueden ser idénticas, proporcionales o diferentes, siguiendo la clasificación establecida en la literatura pero esta clasificación puede ser confusa cuando se consideran los tiempos de preparación y, por consiguiente, en el siguiente apartado se propone una nueva clasificación de las máquinas cuando existen tiempos de preparación.

Bajo esta perspectiva, la asignación de máquinas a las operaciones, repercutirá significativamente en la eficiencia del programa. Así pues, en un programa para un flow shop híbrido, es tan importante la secuenciación de piezas en cada máquina como la asignación de éstas a una de las diferentes máquinas de la etapa (carga).

Este tipo de sistemas permite modelizar diferentes procesos industriales comunes en el mundo industrial. En el curso de esta investigación se ha visitado varias empresas de sectores diferentes y se ha podido contrastar la pertinencia de este enfoque de modelización para el sistema productivo de una fábrica de helados, de una empresa que fabrica cacao en polvo, y de una empresa de etiquetas adhesivas, entre otros. A partir de la problemática detectada en cada uno de estos casos, se ha estimado justificado el interés del estudio de procedimientos que permitan programar la producción de forma eficiente. Inicialmente se ha elegido como sistema prototipo, el de la fábrica de helados cuyo sistema productivo (Figura 1), se puede

modelizar como un *flow shop* híbrido sin esperas, es decir, un sistema con varias etapas, más de una máquina en alguna de las etapas y sin almacenajes intermedios en el que, además, se considerarán tiempos de preparación dependientes de la secuencia de productos a fabricar.

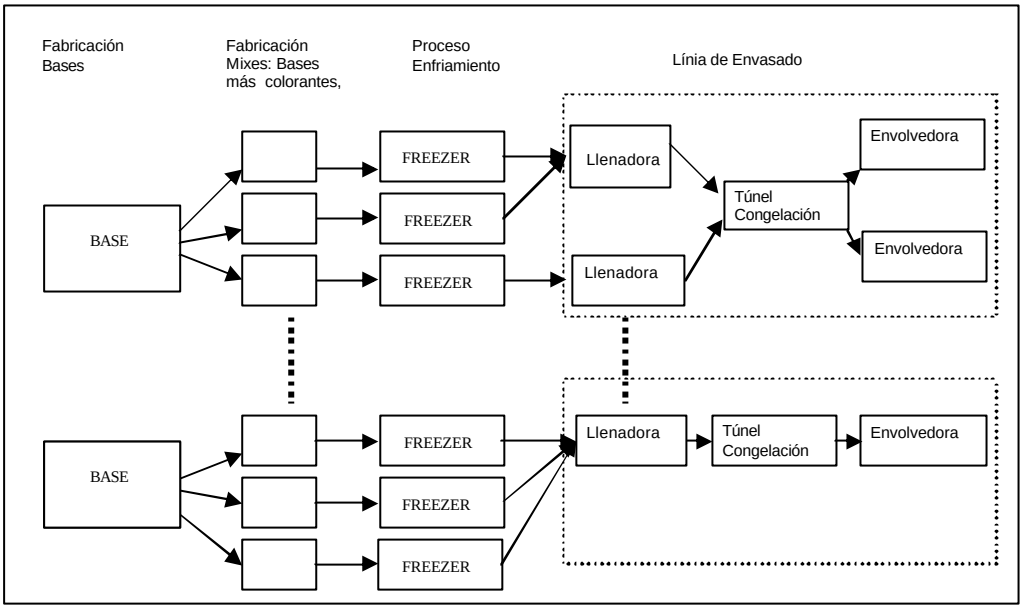

Figura 1. Modelización del sistema productivo de la fábrica de helados

Los programas de producción se elaboran con una misión, para alcanzar un objetivo. La medición de la cuantía en que se ha logrado el resultado apetecido se denomina la eficiencia del programa (aunque normalmente se mide la ineficiencia, es decir la desviación del programa respecto al objetivo). En esta investigación, inicialmente, se ha estudiado la programación de la producción en el caso monocriterio en el que el objetivo buscado es la minimización del retraso medio, equivalente a minimizar la suma de retrasos de las piezas. A continuación se han adaptados los procedimientos implementados al caso bicriterio, dado que en la mayoría de los entornos reales, la programación de la producción persigue un objetivo complejo que considera, implícita o explícitamente, más de un criterio simple de eficiencia. Con ello se pretende tener en cuenta los diversos puntos de vista, no coincidentes en sus intereses, en especial el del fabricante y el del cliente. Una de sus preocupaciones del fabricante suele ser el exceso del nivel de stock que se puede medir a través del plazo medio de fabricación (que obviamente deseará minimizar). En cambio, un cliente exigirá un nivel de servicio alto que se puede medir a través de los retrasos y de los incumplimientos, que se desearán minimizar.

1.1. Clasificación de las máquinas considerando tiempos de preparación

Las máquinas disponibles en cada etapa, siguiendo la clasificación propuesta en la literatura, 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 distintas si 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 (Tabla 1).

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

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.

2. Definición e hipótesis del problema tratado

Se consideran un conjunto de *n* piezas con *S* operaciones cada una que se deben procesar en una de las *m* máquinas idénticas que pueden existir en cada una de las *s* etapas. 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 para cada etapa s tiene asociado un tiempo de proceso p_{i_s}, la fecha de vencimiento d_i y la familia $g(i)$ a la que pertenece. El número de familias es $q \le n$, 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*; entre dos piezas sucesivas pertenecientes a la misma familia el tiempo de preparación es nulo. Se supone que la matriz de tiempos de cambio cumple la desigualdad triangular. Se conoce además, el instante, t_{j_s}, en que cada máquina de la etapa *s* queda libre (o disponible) para realizar las operaciones sobre las *n* piezas, el estado inicial de cada máquina, j_{j_s}, que corresponde a la familia de la última pieza procesada por esta y el instante r_{i_s} en que está disponible la pieza i para iniciar su operación en una máquina de la etapa *s* (la preparación de la misma, si la hubiera, puede haber empezado antes).

Siendo c_i el instante en que una pieza termina su proceso, el retraso T_i viene dado por:

$$T_i = \max\{0, c_i - d_i\} \qquad (1)$$

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

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

En 1999, ***Vignier et al***, proponen una notación unificada que permita establecer la tipología de los problemas en estudio. Los autores proponen seguir la notación de ***Lawler et al*** (1993), $a \mid b \mid g$, pero desglosando el campo a en cuatro parámetros según la siguiente forma: $a = a_1 a_{2,} \left(a_3 a_4{}^{(1)}, a_3 a_4{}^{(2)}, ... a_3 a_4{}^{(a_2)} \right)$. Donde,

- a_1 indica si el problema en consideración es un *Flow Shop* híbrido, que se designa con FH, o de otro tipo (0, O, J, F).
- a_2 indica el número de etapas
- a_3 y a_4 se repiten para cada una de las a_2 etapas.
  - a_3 valdrá 0 si hay únicamente una máquina en la etapa l, P si hay máquinas en paralelo, Q si las máquinas son proporcionales o R si hay máquinas en paralelo no relacionadas.
  - a_4 indicará el número de máquinas en la etapa l.

Los autores proponen agrupar $\left((a_3 a_4)^l \right)_{l=i}^{k}$ según el siguiente esquema, $\left((a_3 a_4)^l \right)_{l=i}^{k}$, para indicar que de la etapa i a la k hay el mismo número de máquinas.

Siguiendo esta notación, el problema objeto de estudio se conoce en la literatura como $FHk, ((PM^{(1)}, PM^{(2)}, ... PM^{(k)}) \left| s_{j,k} \right| \Sigma \mathrm{T}.$

Podemos considerar que un programa, en este caso, se compone de los siguientes elementos,

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

En lo que sigue consideraremos únicamente programas semiactivos en los que las máquinas no están ociosas (están siendo preparadas para la familia de la siguiente pieza o realizando una operación) mientras queden piezas asignadas por realizar.

3. Políticas de programación en sistemas flow shop híbrido

Para construir un programa de producción en un sistema *flow shop* híbrido, ya sean las máquinas de una misma etapa, iguales o diferentes, se puede seguir dos estrategias básicas: la programación por etapas o por piezas.

En la estrategia de programación por etapas se coloca una secuencia (sarta) frente a la primera etapa, y se asignan las piezas sucesivamente, por ejemplo, a la máquina que termina antes. Una vez asignadas todas las piezas a la primera etapa se crea una nueva sarta, ordenando las piezas según orden creciente de su tiempo de fin en la etapa anterior, frente a la segunda etapa y se procede de la misma forma hasta llegar a la última etapa.

En la estrategia de programación por piezas se coloca una sarta frente a la primera etapa y se van asignando las piezas, una a una, a todas las etapas teniendo en cuenta las asignaciones anteriores. En este caso, si no se cumple las restricciones del problema, por ejemplo una restricción del tipo *nowait*, se puede hacer un *backtracking*, que no es otra cosa que la programación hacia delante y hacia atrás conocida en la literatura anglosajona por *scheduling forward and backward.* Pero para llevar a cabo el *backtracking* se deben definir las reglas de juego, es decir, siempre se pueden utilizar las mismas máquinas que ya estaban asignadas variando únicamente el inicio y fin de la operación o se puede valorar la conveniencia de cambiar la máquina asignada. Por tanto, en la definición del procedimiento, y según la medida de eficiencia considerada, se deberá definir la política de *backtracking* a utilizar.

En el caso de sistemas *nowait* la programación por etapas no es posible y se debe proceder con la estrategia de programación por piezas.

4. Procedimientos de resolución

Llamamos **fila** (*row*), a un conjunto ordenado de k piezas, 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 la secuencia son diferentes.

Llamamos **pelotón** (*squad* o *array)* a un grupo de m filas sin piezas comunes. Un pelotón es equivalente a una asignación más una secuenciación. Al conjunto de todos los pelotones los denominamos P.

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 (Figura 2).

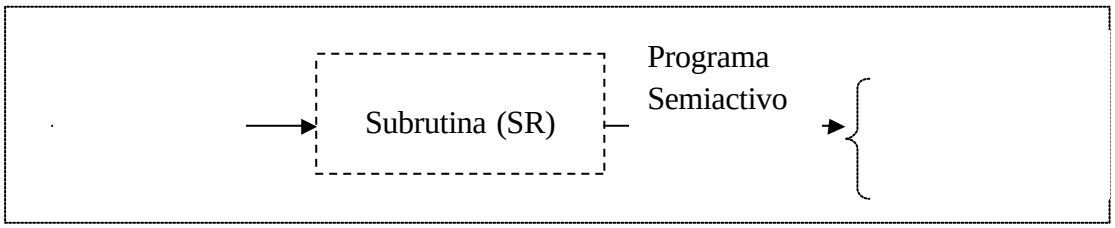

Figura 2. Subrutina básica

Sean dos máquinas idénticas j_1, j_2 ambas disponibles en el instante t para procesar la familia f. Diremos que dos pelotones p_1, p_2 son equivalentes (Figura 3) si p_1 procesa la fila A en j_1 y la fila B en j_2, y p_2 procesa la fila B en j_1 y la fila A en j_2.

Llamamos **sarta** (*string)* a una permutación de las n piezas. Al conjunto de sartas lo denominamos **?**. Existen n! sartas, $|\,?\,| = n!$

A partir de una **sarta** podemos generar un pelotón mediante un algoritmo adecuado (Algoritmo 1 o Algoritmo 2), que llamaremos **grinder**.

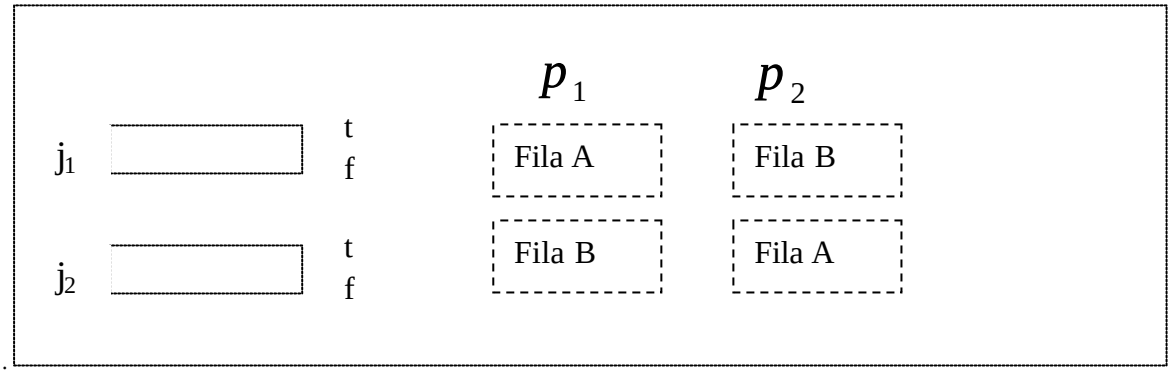

Figura 3. Pelotones equivalentes

Algoritmo 1 (constructor_1): Transforma una sarta σ en un pelotón π. Se procede a la asignación, secuenciación y temporización de las piezas progresivamente, una después de otra y para ello,

paso 1: Hacer k = 1 y $f_{i,j} = \{t_i + TP_{h,g_i}, r_i\}$ (instante de disponibilidad para asignar la pieza i a la máquina j) para j = 1, 2, ..., m, y $i=1,2,\ldots N$.

paso 2: Sea k el valor en curso, corresponde asignar la pieza [k] que ocupa la posición k en σ. Calcular f-min = $\min_j \{f_j\}$; sea $\bar{j}$ la máquina que proporciona f-min (en caso de empate se elige la de menor j).

paso 3: Asignar [k] a la máquina $\bar{j}$ y determinar el nuevo valor de $f_{\bar{j}}$; 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

Algoritmo 2 (constructor_2): Transforma una sarta σ en un pelotón π. Se procede a la asignación, secuenciación y temporización de las piezas progresivamente, una después de otra y para ello,

paso 1: Hacer k = 1 y $f_j = t_j$ (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, …, m

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

paso 4: Asignar [k] a la máquina $\bar{j}$ 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 2. Transforma una sarta en pelotón

Proposición 2: A la función asociada al Algoritmo 1, la denominamos **grinder1** y a la asociada al Algoritmo 2, **grinder2**.

grinder1(σ) = π_1
grinder1(σ) = π_2

Para una misma σ generalmente $\pi_1 \neq \pi_2$. Llamamos P_1 a la imagen de **?** dada por grinder1 y P_2 la dada por grinder2. Obviamente $P_1 \in P$ y $P_2 \in P$. Normalmente $P_1 \neq P_2$.

Proposición 3: $|P_1| = n!$ y $|P_2| = n!$, ya que **?** tiene n! elementos.

Es posible que $|P_1| = n!$ (bastará demostrar que dos elementos de **?** distintos tienen imágenes diferentes) pero no ocurre así con P_2 ya que es fácil construir ejemplos en los que dos elementos de **?** distintos tienen la misma imagen.

Proposición 4: Ni P_2 ni P_1 pueden coincidir con P.

En efecto el número de pelotones es $n! \cdot S(n,m)$ donde $S(n,m) > 1$ si $n > 0$ y $m > 1$. $S(n,m)$ es el número de formas distintas de descomponer n en m sumandos números naturales, incluyendo el 0 (el orden de los sumandos se tiene en cuenta).

Si colocamos las filas de un pelotón una detrás de otra obtenemos una sarta, para reconstruir el pelotón necesitamos conocer el número de piezas de cada máquina.

Llamamos a cada descomposición de n en m sumandos números naturales **perfil** o **esquema de corte** (*outline*). Al conjunto de esquemas lo denominamos **E**.

Proposición 5: $S(n,m) = \sum_{i=0}^{n} S(i, m-1)$ para m > 1. $S(n,1) = 1$ para todo $n = 0$. $S(0,m) = 1$ para todo $m = 1$.

Sería deseable que existiese la función inversa $(\text{grinder1})^{-1}$ o $(\text{grinder2})^{-1}$ ya que la relación entre ellos sería biunívoca y, por lo tanto, podría trabajar con sartas o pelotones indistintamente. Más aún, permitiría diseñar procedimientos heurísticos que combinaran la exploración del vecindario de la sarta y de su pelotón asociado aumentando el número de vecinos a evaluar y, por consiguiente, las opciones de mejora de una solución dada. Sin la existencia de las funciones inversas el procedimiento no es tan directo ya que se debe trabajar simultáneamente con sartas y pelotones si se quiere combinar mejoras a través del vecindario de ambos ya que después de una modificación en el pelotón, la reconversión de éste en sarta no garantiza una correspondencia única.

Se ha estudiado la existencia de ambas funciones $(\text{grinder1})^{-1}$ y $(\text{grinder2})^{-1}$ concluyendo que no existen, ya que las aplicaciones grinder1 y grinder2 no son aplicaciones biyectivas (Figura 4). Se puede comprobar fácilmente que a cada punto del conjunto de sartas le corresponde una imagen del conjunto de pelotones pero que una imagen de este conjunto puede corresponderse con más de un punto o con ningún punto del conjunto de sartas.

Obviamente podemos elegir una sarta σ cualquiera, obtenida al azar o construida mediante otro procedimiento. Utilizando el grinder adecuado, le corresponde un pelotón. Por tanto dada una sarta podemos obtener un valor de la función objetivo, por ejemplo del retraso total. Intuitivamente $R = F(\sigma)$ donde σ es la sarta.

NOTA: Al procedimiento que a partir de una sarta σ proporciona un pelotón (asignación y secuenciación de cada máquina) lo denominamos *molinillo* (***grinder***) y el que a partir de un

programa proporciona la sarta *molinillo inverso* (***ungrinder***).

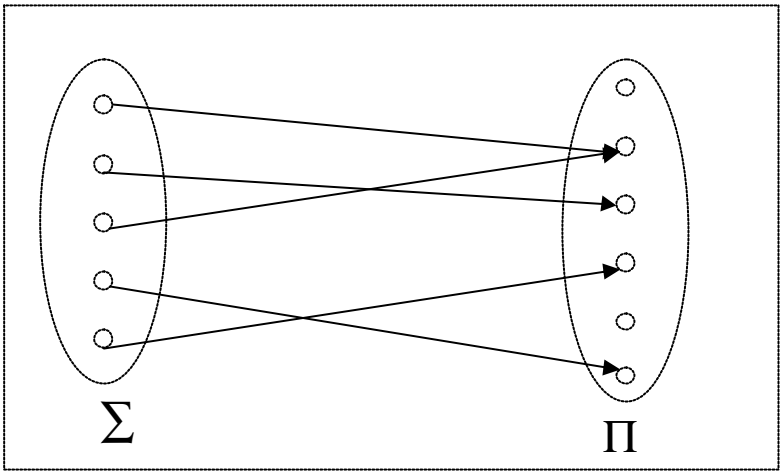

Figura 4. Aplicación grinder

4.1. Estrategia de programación por etapas

El algoritmo de programación diseñado para secuenciar un conjunto de *n* piezas en un sistema *flow Shop* híbrido con *s* etapas se puede ver como un sistema cerrado en el que entra una sarta (s) y sale la asignación y secuenciación de las piezas en las máquinas de cada etapa (?). A este algoritmo le llamaremos **Algoritmo Básico (AB)**.

El Algoritmo AB (figura 5) está constituido por un conjunto, que se repetirá *s* veces, formado por un algoritmo **grinder2**, que transforma la sarta de entrada en un pelotón y un algoritmo que llamaremos **Unificador** que convierte el pelotón en sarta.

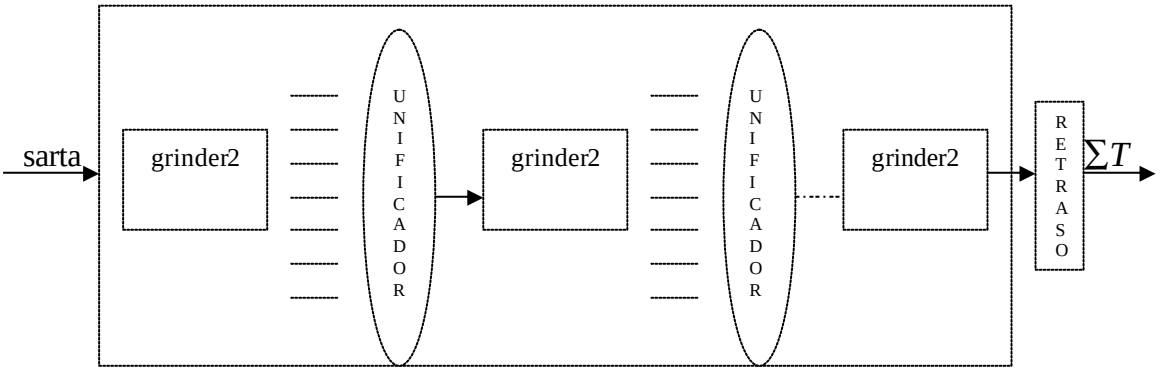

Figura 5. Algoritmo Básico

El algoritmo **Unificador** crea una sarta ordenando las piezas que salen del algoritmo **grinder2** en orden no decreciente de su tiempo de disponibilidad (tiempo de finalización de la operación en curso).

Las heurísticas implementadas se dividen en dos partes: un procedimiento para obtener una solución inicial, y un procedimiento de optimización local sobre la sarta inicial.

Un ejemplar se describe mediante:

- S matrices cuadradas (de dimensión n) de tiempos de preparación ST, una matriz para cada una de las etapas. Se supone que las matrices de tiempos de cambio cumplen la desigualdad triangular.
- el número de máquinas en paralelo en cada etapa, m_s
- el número de piezas *n*, el número de familias *q*
- la familia de la última pieza procesada en cada una de las máquinas de cada etapa j_{j_s},
- un conjunto de piezas de los que se conoce los tiempos de proceso por etapa $p_{i_s}\ (i=1,...,N)$, la fecha de entrega $d_i\ (i=1,...,n)$ y la familia a la que pertenece $g(i)\in\{1,2,...,q\}$ y $i=1,2,...,n$.

4.1.1 Heurística CR

Para hallar una solución inicial, se calcula de forma dinámica un índice crítico para la primera etapa, combinación de pieza y máquina. El índice se utiliza como criterio de asignación de las piezas a las máquinas de la primera etapa de forma que se asigna la pieza a la máquina con cuya combinación se obtenga el índice de menor valor.

Se define el índice crítico para cada pieza i en cada máquina j de la etapa 1 mediante la expresión (1), que se aplica de forma iterativa:

$$CR_{i,j}(t) = a \cdot d_i + (1-a)\cdot\left(\max\left\{t_{j_1} + ST_{(j_j,g_i)_s}, r_{i_1}\right\} + \sum_{s=1}^{S} p_{i_s} + \sum_{s=1}^{S} ST_{med}\right) \qquad (3)$$

donde a puede tomar cualquier valor entre 0 y 1; d_i es la fecha de entrega de la pieza i, p_{i_s} es el tiempo de operación de la pieza i en la etapa s, ST_{hg_i} es el tiempo de preparación necesario para realizar la familia g_i si la pieza anterior era de la familia h, t_{j_1} es el instante en que la máquina j de la etapa *1* queda libre y r_{i_1} es el instante de disponibilidad de la pieza i para la etapa *1*.

Posteriormente, para asignar y secuenciar las piezas en las máquinas de las etapas posteriores, se procesa mediante el algoritmo **Unificador** el pelotón obtenido en la primera etapa y la sarta resultante entra en el algoritmo **AB** desde la etapa 2 hasta la etapa *S*.

Solución Inicial CR (dado un valor a)
  $p = 0$
  Mientras no se haya programado todos los lotes
    Desde $i = 1$ hasta $N- p+1$
      Desde $j=1$ hasta m
        Calcular $CR_{i,j}$
        Si $CR_{i,j} <$ min
          Min= $CR_{i,j}$
          i_{min}=i
          j_{min}=j
        Fin si
      Fin desde

Fin desde
Asignar i_{min} a la máquina j_{min}
Borrar i_{min} del vector de piezas no programadas
p=p+1
Fin Mientras

Unificador
Desde s=2 hasta S
Grinder2
Unificador
Fin desde
Retraso (T)
Fin solución Inicial CR

Algoritmo 3. Solución inicial CR

Procedimiento de Mejora

A partir de la solución inicial se generan y evalúan sus vecinos. Un vecino se obtiene permutando dos piezas, dentro de la secuencia disponible y valorando la suma del retraso de las piezas. Para recorrer el espacio de soluciones de forma distinta en cada iteración y poder acceder a vecindarios no explorados se utiliza un vector de posiciones, REV, que permite codificar los punteros de posición que se usan durante la exploración del vecindario. El vector REV contiene, inicialmente, de forma ordenada, las diferentes posiciones que una pieza puede ocupar en la secuencia. A continuación se mezclan las posiciones a través del azar y, durante la aplicación del procedimiento de mejora de la heurística SSA, se codifican, a través del vector REV, los punteros de posición utilizados para explorar el vecindario de una solución. Es decir, para una i, j determinada se busca su equivalente, i_{rev} y j_{rev}, en el vector REV siendo $i_{rev} = rev(i)$ y $j_{rev} = rev(j)$, y se aplica el procedimiento con estos nuevos punteros. Si la permutación entre ambas consigue reducir el retraso, se mantiene el cambio y se evalúa el vecindario de la nueva solución (sin terminar la generación de los vecinos de la solución primitiva). En cambio, si la nueva permutación no mejora el retraso anterior, se mantiene la secuencia original y, si es oportuno, se sigue explorando el vecindario. Cuando no se produce ninguna mejora en la evaluación de un cierto vecindario, se finaliza. Los empates se resuelven mediante un procedimiento aleatorio que simula el lanzamiento de una moneda: si sale cara, se queda con la nueva permutación y si es cruz, se mantiene la antigua.

Procedimiento de Mejora SSA2-HFS
$T_{tot}^* = T_{tot}(B)$; *empates = 0;* $q = 0$; $g = 1$;
Mientras $g = 1$ y *empates < 100*
$g = 0$
Para $i_{rev} = 1$ hasta el número de posiciones del vector B
$rev(i_{rev}) = i_{rev}$
Fin para
Para $i_{rev} = 1$ hasta el número de posiciones de B
$j_{rev} = 1 + Int($ número de posiciones de $B \cdot aleatorio(\ ))$
Intercambiar las posiciones (i_{rev}, j_{rev})
Fin para

Mientras $j \le N$

$j_{rev} = rev(j)$

Mientras $i \le N$

$i_{rev} = rev(i)$

Intercambiar las posiciones (i_{rev}, j_{rev}) en vector $B : B'$

Algoritmo Básico(s : ?, $T_{tot}(B')$)

Si $T_{tot}(B') < T_{tot}$ *

$g = 1; B^* = B'; T_{tot}^* = T_{tot}(B')$; *empates = 0;* $q = 1$

Fin Si

Si [$T_{tot}(B') = T_{tot}$ *] y [Aleatorio( ) < 0,5];

$g = 1; B^* = B'; T_{tot}^* = T_{tot}(B')$; *empates = empates + 1*

$q = 1$

Fin Si

Si $q = 0$

Deshacer intercambio posiciones *(* i_{rev}, j_{rev}*)* en B':B

Sino $q = 0$ *;*

Fin Si

$i = i + 1$

Fin Mientras

i = 1; j = j+1

Fin Mientras

Fin Mientras

Fin Procedimiento de Mejora CR2-HFS

Algoritmo 4. Procedimiento de mejora SSA2 –HFS

4.1.2. Heurística GRASP

Para disponer de una solución inicial, se construye una secuencia de fabricación a partir de la ordenación de las piezas según un orden no decreciente de su índice crítico que se calculará de forma dinámica según la fórmula (3). A continuación, se toma como valor de referencia el índice menor con un incremento del 20% y se selecciona, al azar, una de las piezas cuyo índice tiene un valor menor que el valor de referencia colocándola como primera pieza a procesar. Una vez fijada la primera pieza se recalculan los índices y se ordena de nuevo la secuencia en función de estos.

A continuación, se ejecuta sobre la solución inicial el mismo procedimiento de mejora que en la heurística CR para intentar mejorar la solución.

4.1.3. Heurística Multistar (MS)

Inicialmente se construye una secuencia de fabricación de forma aleatoria 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 en la posición *i* de la secuencia el valor de *aux(i)*. La secuencia creada se procesa a través del Algoritmo Básico para obtener la solución inicial

A continuación, se ejecuta sobre la solución inicial el mismo procedimiento de mejora que en la heurística CR para intentar mejorar la solución.

4.2. Estrategia de programación por pìezas

La estrategia de programación por piezas es la estrategia a seguir cuando se debe programar la producción en un sistema *nowait*. En particular, un sistema *flow shop* híbrido *nowait* es un sistema productivo que consta de varias etapas en las que puede haber más de una máquina en paralelo y en el que las piezas deben abandonar inmediatamente, una vez terminada la operación, la máquina que las procesa en una etapa para pasar a la siguiente etapa. Dado que las piezas que abandonan una máquina no pueden esperar a que haya una máquina disponible en la siguiente etapa para empezar a procesarse, se deberá comprobar, previamente, que existe una máquina disponible en cada etapa en el instante requerido. Si además se tienen en cuenta los tiempos de preparación, se deberá comprobar que, al menos una máquina en cada etapa, no sólo está disponible en el instante requerido, sino también está preparada para procesar el tipo de pieza en curso.

Para poder tener en cuenta esta restricción se ha modificado el **Algoritmo Básico** desarrollado para el caso general. A este nuevo algoritmo que dada una sarta (s) de entrada devuelve tanto el valor del retraso acumulado (?) como la asignación y secuenciación de las piezas en las máquinas de cada etapa (?) respetando la restricción *nowait*, le llamaremos **Algoritmo Básico *nowait* (ABNW),**

El funcionamiento del **ABNW** es el siguiente:

En un instante *t*, se tienen *k-1* piezas programadas y se debe proceder a la programación de la pieza que ocupa la posición *k*. La programación se lleva a cabo desde la última etapa hacia delante. Así pues, empezando por la última etapa, *S*, se coloca tentativamente la pieza *k* en la máquina con la que acabe antes. Si hay empate se elige la máquina de menor número. Conocido el tiempo de finalización de la pieza en la etapa *s*, f_{k_s} se puede calcular el instante de inicio, e_{k_s}, restándole a éste el tiempo de proceso, p_{k_s}. El instante de inicio en la etapa s, e_{k_s}, es el instante de fin en la etapa $s-1$ (condición *nowait*). Si, para cualquier máquina, e_{k_s} es menor que la suma del instante de disponibilidad de la máquina, t_j, más el tiempo de preparación $ST_{(h,g_k)_s}$ necesario, se elige la máquina con la que termine antes fijando, de esta forma, el instante de fin, y por consiguiente el instante de inicio de la operación ($f_{k_s} - p_{i_s}$), y se reprograma hacia atrás las etapas posteriores, manteniendo la máquina que tenían asignada y teniendo en cuenta la condición *nowait*. Esta reprogramación implica un desplazamiento temporal de los instantes de inicio y fin de las operaciones en las etapas siguientes. Se procede siguiendo este esquema hasta la etapa 1.

```
Inicio ABNW
        Desde i =1 hasta N
          S=num_etapas
          Mientras s>0
              Si s  = última etapa
                      min=M          (valor muy grande)
```

Desde j=1 hasta m_s

$f_{i_s} = t_j + ST_{(j_j, g_i)_s} + p_{i_s}$

Si min> f_{i_s}

min= f_{i_s} ; min_maq=j

Fin si

Fin desde

Asignar (*i: min_maq)*

Programar (*i*: e_{i_s} , f_{i_s})

Sino

Dif1=M

Dif2=-M

Desde j=1 hasta nmaq(s)

dif= $t_j + ST_{(j_j, g_i)_s} - (e_{i_{s+1}} - p_{i_s})$

Si Dif <0

Si dif > Dif1

Dif1= dif; min_maq1= j

Fin si

Sino

Si dif < Dif2

Dif2 = dif; min_maq2 = j

Fin si

Fin Desde

Si Dif1 = -M Then

min_dif = Dif2; min_maq = min_maq2

Sino

min_dif = Dif1; min_maq = min_maq1

Fin si

Si min_dif>0

$f_{i_s} = e_{i_{s+1}} + \min\_ dif$

Sino

$f_{i_s} = e_{i_{s+1}}$

Fin si

Asignar (*i: min_maq)*

Programar (*i*: e_{i_s})

Si min_dif>0

k=s+1

Mientras k = num_etapas

$e_{i_k} = e_{i_k} + \min\_ dif$

$f_{i_k} = f_{i_k} + \min\_ dif$

k=k+1

Fin mientras

Fin si

Fin si

s=s-1

Fin mientras

Fin desde

Retraso (T)
Fin ABNW

Algoritmo5. Algoritmo Básico *nowait* (ABNW)

En el ABNW, cuando se reprograma las operaciones hacia atrás se mantiene la máquina asignada durante la programación hacia delante. Pero está máquina puede no ser la más adecuada con respecto al nuevo instante de inicio de la operación y, por lo tanto, si se reevalúa la disponibilidad de las máquinas de la etapa puede existir una, cuyo instante de disponibilidad más preparación, sea más próximo al instante de inicio requerido. Esta nueva política se ha implementado en un algoritmo que llamaremos ABNW2. Este algoritmo tiene la misma filosofía que el ABNW pero, al reprogramar hacia atrás, evalúa nuevamente la máquina a asignar. De esta forma, se programa desde la última etapa hasta la primera calculando los instantes de inicio y fin de la etapa, como si se asignara a la máquina con la que termina antes y al llegar a la primera etapa, no sólo se temporiza la pieza, sino que además se hace una reasignación de máquina, si es necesario, actualizando los valores de ésta (instante de disponibilidad y familia para la que está preparada). A continuación se programa hacia atrás (desde la etapa 2 hasta la última etapa) asignando la pieza, de entre las máquinas que pueden estar preparadas para el instante de inicio requerido, a la máquina cuya diferencia, entre el instante de inicio requerido por la operación y el instante en que la máquina puede quedar preparada (instante de disponibilidad más tiempo de preparación necesario), sea mínima.

Inicio ABNW2
Desde i =1 hasta N
S=*num_etapas*
Mientras s>0
Si s = *S*
Min=M (valor muy grande)
Desde j=1 hasta nmaq(s)
$f_{i_s} = t_j + ST_{(h,g_i)_S} + p_{i_s}$
Si min> f_{i_s}
min= f_{i_s} ; min_maq=j
Fin si
Fin desde
Programar $(i, min: e_{i_s}, f_{i_s})$
Sino
Dif1=M
Dif2=-M
Desde j=1 hasta nmaq(s)
dif= $t_j + ST_{(j_j,g_i)_s} - (e_{i_{s+1}} - p_{i_s})$
Si Dif <0
Si dif > Dif1
Dif1= dif; Min_maq1= j
Fin si
Sino
Si dif < Dif2
Dif2 = dif; Min_maq2 = j

```
                    Fin si
                Fin si
            Fin Desde
            Si      Dif1 = -M Then
                    min_dif = Dif2; min_maq = min_maq2
            Sino
                    min_dif = Dif1; min_maq = min_maq1
            Fin si
            Si      min_dif>0
                    f_{i_s} = e_{i_{s+1}} + min_ dif
            Sino
                    f_{i_s} = e_{i_{s+1}}
            Fin si
                    k=s
                    Programar  (i, min_dif:  e_{i_k} , f_{i_k} )
            Si      k=1
                    Asignar (i: min_maq)
            Fin si
                    k=k+1
            Si      k=2
                    Mientras k<= S
                        Min=-1000
                        Desde   j=1 hasta nmaq(k)
                            Si t_j + ST_{(j_j,g_i)_s} - f_{i_{k-1}} < min
                                min= t_j + ST_{(j_j,g_i)_s} - f_{i_{k-1}} ; min_maq = j
                            Fin si
                        Fin desde
                        Asignar (i: min_maq)
                        Programar  (i:  e_{i_s} )
                        k=k+1
                    Fin mientras
            Fin si
        Fin si
        s=s-1
    Fin mientras
Fin desde
Retraso (T)

Fin ABNW2
```

Algoritmo 6. ABNW2

Podríamos haber probado otras políticas en la reprogramación hacia delante que fueran menos ciegas (greedy), como por ejemplo, tener en cuenta la siguiente pieza a programar comprobando la eficacia de la asignación a una máquina y su repercusión en la siguiente pieza a programar optando por la asignación que sea más eficaz teniendo en cuenta ambas piezas. Se puede discutir que se entiende por asignación eficaz ya que se pueden establecer diferentes

criterios. Dado que el campo de investigación, en este sentido, puede ser muy amplio, se dejará como futuras investigaciones.

Las heurísticas implementadas en este caso son la adaptación de las implementadas para el caso general. Esta adaptación implicará la substitución del **Algoritmo Básico** por el **ABNW** o el **ABNW2** tanto en los procedimientos de mejora como en los procedimientos para la obtención de la solución inicial que lo requieran. Para experimentar la eficiencia del **ABNW** y del **ABNW2** se han probado las heurísticas desarrolladas con ambos algoritmos.

4.3 Adaptación de las estrategias de programación al caso bicriterio

Uno de los objetivos de esta tesis es estudiar procedimientos de programación en sistemas *flow shop* híbrido cuando existe más de un objetivo de programación. Hasta el momento se ha tenido en cuenta un único criterio, en particular, los procedimientos desarrollados pretenden minimizar el retraso total.

A continuación se han desarrollado procedimientos de programación multicriterio cuya función objetivo pretende dar respuesta a una de las demandas de los clientes, como puede ser el nivel de servicio, y a uno de los objetivos del programador a través de la minimización del stock en curso. El primer objetivo se ha modelizado mediante la minimización de la suma del retraso total de las piezas y el segundo a través de la minimización de la suma de los tiempos de proceso de las piezas. De esta forma, la nueva función objetivo en los procedimientos de programación viene dada por la suma ponderada de ambos criterios (4).

$$[MIN]z = l \cdot \sum T + (1 - l) \cdot \sum C \qquad (4)$$

siendo ? el parámetro que permitirá la sintonización de ambos criterios en función de las necesidades del programador.

Dado que la suma del retraso total y el tiempo de proceso total responden a unidades diferentes, los factores de la ecuación (13) se deben normalizar. Para ello, se calcula inicialmente, el retraso máximo y tiempo de proceso máximo para una secuencia formada por las piezas ordenadas según orden creciente del número de pieza. Estos valores permiten formular la ecuación (14) que actúa como función objetivo en los procedimientos desarrollados.

$$[MIN]z = l \cdot \frac{\sum T}{T_{\max}} + (1 - l) \cdot \frac{\sum C}{C_{\max}} \qquad (5)$$

Los valores T_{max} y C_{max} se calculan para cada ejemplar una única vez y son independientes del procedimiento de resolución utilizado.

Así pues, los procedimientos desarrollados son una adaptación de los métodos implementados hasta el momento para el caso bicriterio. En este caso la adaptación significa el cálculo de C_{max} y T_{max} para cada ejemplar y la incorporación de la ecuación (14) cuando se tenga que valorar una secuencia dada.

5. Experiencia Computacional

Se ha analizado de forma separada los procedimientos desarrollados según si varia o no el número de máquinas en las diferentes etapas, tanto para el caso monocriterio como para el caso bicriterio. Se muestran además, para cada uno de los casos, los resultados obtenidos teniendo en cuenta o no la restricción *nowait*.

5.1. Flow shop híbrido con igual número de máquinas por etapa

La experiencia computacional se ha realizado sobre 3 colecciones, una formada por 40 ejemplares con 15 piezas cada uno que pueden pertenecer a 4 familias, otra formada por 70 ejemplares de 20 piezas que pueden pertenecer a 4 o 5 familias diferentes, y el tercero lo forman 90 ejemplares con 25 piezas que también pueden pertenecer a 4 o 5 familias diferentes, según el ejemplar. En los ejemplares de las diferentes colecciones las piezas deben procesarse en 3 etapas con 4 máquinas idénticas en cada una de ellas. Para cada ejemplar, se dispone de 3 matrices, una para cada etapa, con los tiempos de preparación necesarios para pasar de fabricar un tipo de pieza a otro.

La comparación entre los procedimientos se ha llevado a cabo en un ordenador Pentium IV, a 2800 Mhz y 512 Mb de Ram. Se ha considerado para cada uno de los métodos el retraso total, según (2).

5.1.1 Tiempos de Ejecución

En la Tabla 2 se muestra los tiempos medios, en segundos, para resolver un ejemplar usando cada uno de los procedimientos.

Tabla 2. Tiempos medios de ejecución, en segundos, por ejemplar.

$FH3,(P3)_{l=1}^{3} \mid s_{jk} \mid \sum T$			
	Número de piezas		
Heurísticas	**n=15**	**n=20**	**n=25**
CR	0.42	0.97	1.81
GRASP	0.47	0.94	1.87
Multistart	0.57	0.97	1.78

La Tabla 3 muestran los tiempos medios, en segundos, para resolver un ejemplar diferenciando según la política de backtracking utilizada (ABNW o ABNW2).

Tabla 3. Tiempos medios de ejecución, en segundos, por ejemplar.

Heurísticas	$FH3,(P3)_{l=1}^{3} \mid s_{jk}, nowait \mid \sum T$ **(ABNW)**			$FH3,(P3)_{l=1}^{3} \mid s_{jk}, nowait \mid \sum T$ **(ABNW2)**		
	Número de piezas			**Número de piezas**		
	N=15	**N=20**	**N=25**	**N=15**	**N=20**	**N=25**
CR	0.5	1.21	2.35	0.8	1.85	3.24
GRASP	0.5	1.18	2.25	0.72	1.72	3.27
Multistart	0.55	1.24	2.31	0.87	1.84	3.46

5.1.2. Calidad de la Solución

Se compara entre sí las heurísticas implementadas, con el fin de determinar que procedimiento es el más adecuado a esta tipología de problemas. La comparación se lleva a cabo concediendo a las heurísticas el mismo tiempo de ejecución lo que permite al programa ejecutarse varias veces y explorar así nuevos vecindarios que, debido a la resolución de empates, pueden haberse descartado anteriormente. Por esta razón y a la vista de los resultados obtenidos en los diferentes casos se ha fijado los siguientes tiempos:

Para el caso general, $FH3,(P3)^{3}_{l=1} \mid s_{jk} \mid \sum T$, se ha fijado un tiempo de cómputo de 10 minutos para la colección formada por 15 trabajos. El tiempo fijado para la colección de 20 trabajos se ha obtenido multiplicando el incremento medio del tiempo computacional al pasar de 15 a 20 trabajos siendo el tiempo total de 35 minutos. Finalmente, a la colección de 25 trabajos el tiempo se ha fijado en 86 minutos siguiendo el mismo procedimiento que en el caso anterior.

Para el caso $FH3,(P3)^{3}_{l=1} \mid s_{jk}, nowait \mid \sum T$ los tiempos fijados varían en función de la estrategia de programación seguida ya que hay una diferencia en los tiempos computacionales obtenidos. Así pues,

- Según la estrategia ABNW para la colección de 15 trabajos se ha fijado un tiempo de 10 minutos, para la de 20 trabajos se ha multiplicado el tiempo por 2,4 que corresponde al incremento medio de tiempo para pasar de 15 a 20 trabajos siendo el tiempo fijado de 42 minutos y para la colección de 25 trabajos se ha multiplicado por 3.75 siendo el tiempo fijado en 85 minutos.

- Según la estrategia ABNW2 para la colección de 15 trabajos se ha fijado un tiempo de 15 minutos que corresponde a la relación entre el tiempo requerido según ABNW y ABNW2, para la de 20 trabajos se ha multiplicado el tiempo por 2,3 que corresponde al incremento medio de tiempo para pasar de 15 a 20 trabajos siendo el tiempo fijado de 60 minutos y para la colección de 25 trabajos se ha multiplicado por 4,2 siendo el tiempo fijado en 141 minutos.

Para comparar los procedimientos desarrollados se ha ejecutado, 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 $I_{h,heuristica}$, siendo *h* el número de ejemplar analizado y *heurística* el procedimiento utilizado para obtener el retraso acumulado.

$$I_{h,heuristica} = (\mu_{h,\text{heurística}} - r_{h,min}) / r_{h,min} * 100 \qquad (6)$$

siendo $\mu_{i,\text{heurística}}$ la media del retraso obtenida para el ejemplar *h* mediante el procedimiento *heurística* y $r_{h,min}$ el retraso mínimo, para este ejemplar, obtenido con cualquiera de las heurísticas y réplicas.

Además, se ha calculado la media (μ) y la desviación estándar (s) del índice $I_{h,heuristica}$, cuyos valores de cada colección (Tabla 4), servirán para comparar los procedimientos entre si.

Tabla 4. Calidad de la solución según $I_{h,heuristica}$

$FH3,(P3)_{l=1}^{3} \mid s_{jk} \mid \sum T$						
Heurísticas	**Número de Piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	0.94	3.06	3.88	3.63	5.62	4.30
GRASP	**0.37**	**0.63**	**2.81**	**2.55**	**5.44**	**3.75**
Multistart	0.43	0.85	3.13	3.03	7.70	5.17

A través de los resultados obtenidos para el caso $FH3,(P3)_{l=1}^{3} \mid s_{jk} \mid \sum T$ se comprueba que los valores menores, tanto de la media como de la desviación estándar, del índice $I_{h,heuristica}$ se obtienen siempre a través del procedimiento GRASP. También cabe decir que la diferencia entre ellos no es demasiado significativa dado el valor de la desviación estándar pero aun así se aconsejaría utilizar la heurística GRASP implementada para resolver el problema de secuenciación en un sistema *flow shop* híbrido con tiempos de preparación dependientes de la secuencia.

Tabla 5. Calidad de la solución según $I_{h,heuristica}$ con estrategia ABNW

$FH3,(P3)_{l=1}^{3} \mid s_{jk}, nowait \mid \sum T$ **(ABNW)**						
Heurísticas	**Número de Piezas**					
	n=15		**n=20**		**n=25**	
	μ	S	μ	s	μ	s
CR	0.83	1.47	4.82	4.66	**6.46**	**4.37**
GRASP	**0.80**	**1.24**	**5.40**	**10.37**	6.58	3.96
Multistart	0.98	1.32	7.09	13.96	8.33	5.51

En la Tabla 5 se muestran los resultados obtenidos para el caso $FH3,(P3)_{l=1}^{3} \mid s_{jk}, nowait \mid \sum T$ según las dos estrategias de programación utilizadas. Los resultados muestran que las heurísticas CR y GRASP tienen un comportamiento similar y que son más eficientes que la heurística Multistar. Los dos primeros procedimientos obtienen una solución inicial dirigida al objetivo de programación utilizado, en cambio la heurística Multistar genera la solución totalmente al azar. Por lo tanto, parece que con la restricción *nowait* es mejor crear soluciones iniciales con significado.

La Tabla 6 muestra los resultados obtenidos para el mismo caso pero con la estrategia de programación ABNW2. En ella se comprueba que, como en el caso anterior, los mejores valores se obtienen con las heurísticas CR y GRASP. En este caso parece más evidente que la aleatoriedad de la solución inicial no es aconsejable.

Tabla 6. Calidad de la solución según $I_{h,heuristica}$ con estrategia ABNW2

$FH3,(P3)^3_{l=1} \mid s_{jk}, nowait \mid \sum T$ **(ABNW2)**						
Heurísticas	**Número de Piezas**					
	N=15		**N=20**		**N=25**	
	µ	s	?	s	µ	s
CR	0.78	1.23	**4.78**	**4.70**	**6.49**	**4.49**
GRASP	**0.44**	**0.73**	5.03	8.48	7.11	6.10
Multistart	0.68	1.06	5.85	8.38	8.13	5.97

Finalmente, para comparar entre sí las dos estrategias de programación utilizadas para el caso *nowait* se calcula el índice $I^*_{h,heuristica}$ que compara el valor de la media de cada uno de los procedimientos, para cada una de las estrategias, con respecto al valor mínimo obtenido mediante cualquiera de las réplicas de cualquier procedimiento. De esta forma,

$$I^*_{h,heuristica} = (m_{h,heuristica} - \min_h) / \min_h * 100 \qquad (7)$$

siendo $\min_h$ el mejor valor conocido para el ejemplar *h*.

Tabla 7. Comparación entre estrategias para el caso *nowait* según $I^*_{h,heuristica}$

Heurísticas	$FH3,(P3)^3_{l=1} \mid s_{jk}, nowait \mid \sum T$ **(ABNW)**						$FH3,(P3)^3_{l=1} \mid s_{jk}, nowait \mid \sum T$ **(ABNW2)**					
	Número de Piezas						**Número de Piezas**					
	N=15		**N=15**		**N=20**		**N=15**		**N=15**		**N=20**	
	µ	µ	s	µ	s	µ	µ	s	µ	s	µ	s
CR	4.52	0.79	1.23	0.79	4.84	7.84	0.79	1.23	**4.94**	**4.84**	**7.84**	**5.25**
GRASP	4.47	0.45	0.73	0.45	9.08	8.27	**0.45**	**0.73**	5.33	9.08	8.27	4.84
Multistart	4.58	0.60	0.94	0.60	8.75	9.67	0.60	0.94	5.88	8.75	9.67	4.69

De la Tabla 7 se deduce que los valores mínimos de la media se obtienen con la estrategia ABNW2 lo que indica que mediante ésta los valores del retraso obtenidos son menores que los obtenidos a través de la estrategia ABNW.

5.2 Flow shop híbrido con diferente número de máquinas por etapa

La experiencia computacional se ha realizado sobre 3 colecciones cuyos ejemplares contienen 15, 20 y 25 piezas a programar respectivamente. En este caso se ha realizado un primer estudio para evaluar el efecto, sobre el tiempo computacional por ejemplar, de considerar diferente número de etapas y familias. Para ello, se ha trabajado con una colección formada con 200 ejemplares en que el número de piezas puede se de 15, 20 o 25. Las piezas pueden ser de 4, 5 o 6 familias diferentes y se deben procesar en 3 o 4 etapas. En cada etapa el

número de máquinas puede ser diferente aunque esta variable no se ha tenido en cuenta en el análisis del tiempo requerido por la dificultad de valorar 4 dimensiones a la vez.

Los resultados obtenidos varían en función del caso considerado. Así pues, se han analizado cada uno de los casos que se tratará en los siguientes apartados.

5.2.1 Tiempos de ejecución

A continuación se muestra, para el caso general, en forma de tabla (Tabla 8, 9 y 10) y gráficamente (Figuras 3, 4 y 5) el tiempo medio requerido, en segundos, para encontrar una solución, en ejemplares que tengan igual características, a través de cada uno de los procedimientos desarrollados.

Según el procedimiento CR:

Tabla 8. Tiempo medio, en segundos, para CR

Piezas	etapas	familias	Tiempo medio
15	3	4	0.42
20	3	4	1.12
25	3	4	2.18
20	4	5	1.41
25	4	5	3.04
25	4	6	2.95

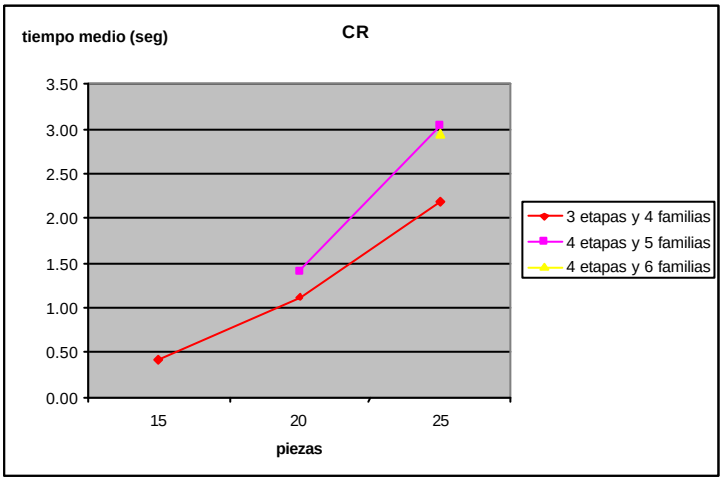

Figura 4. Tiempos medios (segundos) para CR

Según la Heurística GRASP:

Tabla 9. Tiempo medio (segundos) para GRASP

Piezas	etapas	familias	Tiempo medio
15	3	4	0.42
20	3	4	1.13
25	3	4	2.05
20	4	5	1.73
25	4	5	2.68
25	4	6	3.15

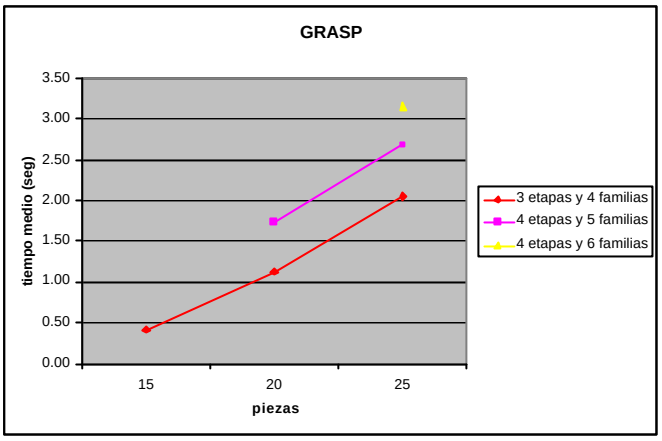

Figura 5. Tiempos medios (segundos)según GRASP

Según la Heurística Multistar:

Tabla 10. Tiempo medio, en segundos, según Multistar

Piezas	etapas	familias	Tiempo medio
15	3	4	0.43
20	3	4	1.23
25	3	4	2.04
20	4	5	1.83
25	4	5	3.19
25	4	6	3.08

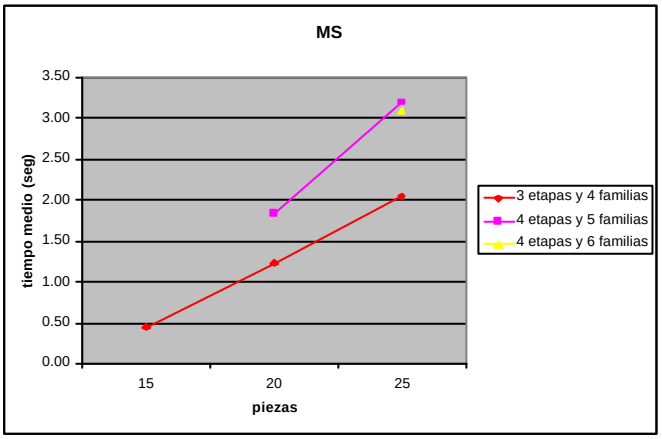

Figura 6. Tiempos medios (segundos) según Multistar

Analizando los resultados se observa que el número de piezas es la variable que repercute en mayor medida sobre el tiempo computacional ya que el incremento observado al variar el número de etapas o el número de familias es inferior. Por esta razón, la experiencia computacional se ha hecho separando los 200 ejemplares en 3 colecciones. Cada colección está formada por ejemplares con el mismo número de piezas variando, para cada ejemplar, el número de etapas (entre 3 y 4) y el número de familias a las que pueden pertenecer las piezas (4, 5 o 6).

La primera colección (colección 1) la forman 40 ejemplares de 15 piezas fijándose un tiempo de cómputo de 10 minutos que resulta a un tiempo medio por ejemplar de 15 segundos.

La segunda colección (colección 2) la forman 60 ejemplares de 20 piezas. Para conceder a los ejemplares un tiempo similar a los de la colección 1 se ha multiplicado por 2,7 el tiempo por ejemplar, que es aproximadamente la repercusión que tiene el aumento de 15 a 20 piezas, según los datos analizados. De esta forma, el tiempo concedido a la colección es de 40 minutos.

La tercera colección (colección 3) está formada por 100 ejemplares de 25 piezas y se ha multiplicado por 5 el tiempo medio concedido por ejemplar que, según los datos analizados, corresponde a la relación aproximada entre el tiempo medio para un ejemplar de 15 piezas y uno de 25. De esta forma el tiempo concedido a la colección 3 es de 125 minutos.

Caso Nowait con algoritmo de programación ABMW

A continuación se muestra el tiempo medio requerido, en segundos, para encontrar una solución, en ejemplares que tengan igual características, a través de cada uno de los procedimientos desarrollados (Tablas 11, 12 y 13 y Figuras 7, 8 y 9).

Heurística CR:

Tabla 11. Tiempo medio, en segundos, según CR

piezas	etapas	familias	Tiempo medio
15	3	4	0.33
20	3	4	0.73
25	3	4	1.69
20	4	5	0.68
25	4	5	1.45
25	4	6	2.07

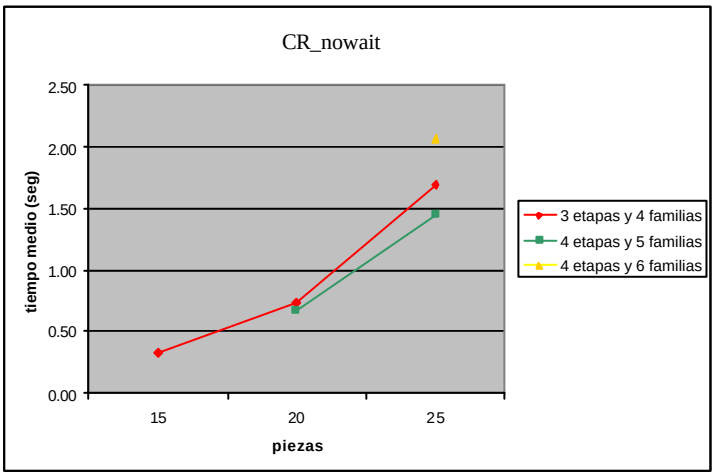

Figura 7. Tiempos medios (segundos) para CR con ABNW

Heurística GRASP:

Tabla 12. Tiempo medio, en segundos, según GRASP

piezas	etapas	familias	Tiempo medio
15	3	4	0.36
20	3	4	0.93
25	3	4	1.50
20	4	5	0.69
25	4	5	1.32
25	4	6	2.01

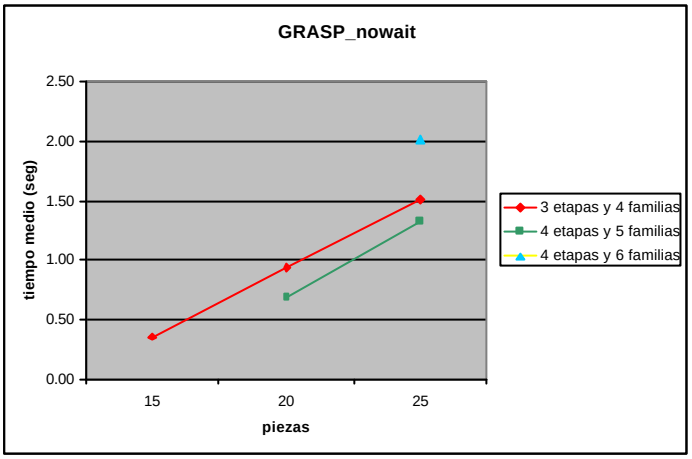

Figura 8. Tiempos medios (segundos), para GRASP con ABNW

Heurística Multistar:

Tabla 13. Tiempo medio, en segundos, según MS

piezas	etapas	familias	Tiempo medio
15	3	4	0.33
20	3	4	0.93
25	3	4	1.93
20	4	5	0.41
25	4	5	1.08
25	4	6	1.76

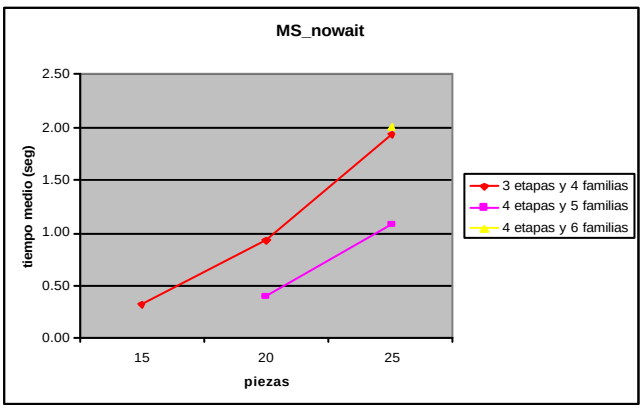

Figura 9. Tiempo medio, en segundos, para MS con ABNW

De las Tablas (Tablas 11-13) y gráficos (Figuras 7-9) se deduce que el número de piezas es la variable que repercute en mayor medida sobre el tiempo computacional ya que el incremento observado al variar el número de etapas o el número de familias es inferior. Por esta razón, a la colección 1 (40 ejemplares de 15 piezas) se ha fijado un tiempo de 10 minutos. A la colección 2 (60 ejemplares de 20 piezas) se ha multiplicado el tiempo por 2,5 que es el incremento de tiempo medio requerido para pasar de 15 a 20 piezas siendo el tiempo fijado de 35 minutos y para la colección 3 (100 ejemplares de 25 piezas) se ha multiplicado por 5 el tiempo medio concedido por ejemplar que corresponde al incremento medio respecto a un ejemplar de 15 piezas. De esta forma el tiempo concedido a la colección 3 es de 125 minutos.

Caso Nowait con algoritmo de programación ABMW2

En las tablas (Tabla 14, 15 y 16) y gráficos siguientes (Figura 10, 11 y 12) se muestra el tiempo medio requerido, en segundos, para encontrar una solución, en ejemplares que tengan igual características, a través de cada uno de los procedimientos desarrollados.

Heurística CR:

Tabla 14. Tiempo medio, en segundos, según CR

piezas	etapas	familias	Tiempo medio
15	3	4	0.60
20	3	4	1.84
25	3	4	5.15
20	4	5	1.49
25	4	5	3.60
25	4	6	4.25

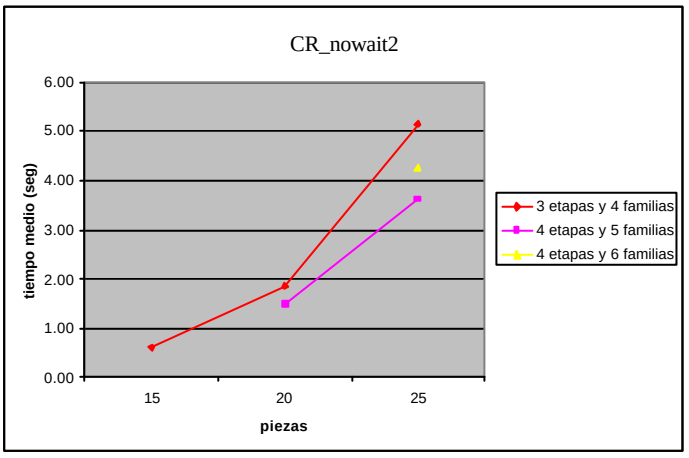

Figura 10. Tiempos medios (segundos) según CR con estrategia ABNW2

Heurística GRASP:

Tabla 15. Tiempo medio, en segundos, según GRASP

piezas	etapas	familias	Tiempo medio
15	3	4	0.62
20	3	4	1.97
25	3	4	4.51
20	4	5	1.78
25	4	5	3.13
25	4	6	4.79

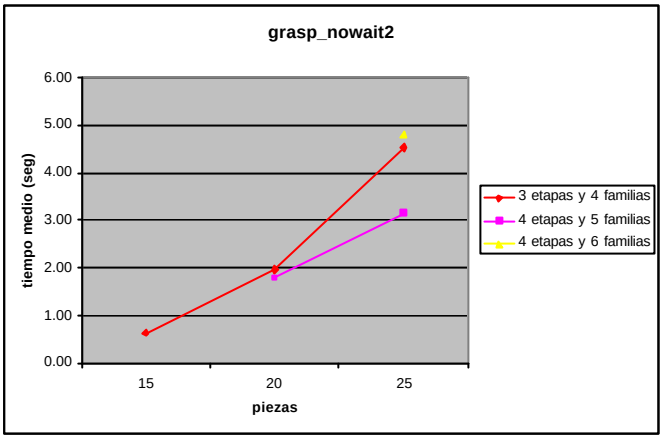

Figura 11. Tiempos medios (segundos) según GRASP con estrategia ABNW2

Heurísitca Multistar:

Tabla 16. Tiempo medio, en segundos, según Multistar

piezas	etapas	familias	Tiempo medio
15	3	4	0.54
20	3	4	1.86
25	3	4	4.63
20	4	5	1.45
25	4	5	3.92
25	4	6	5.13

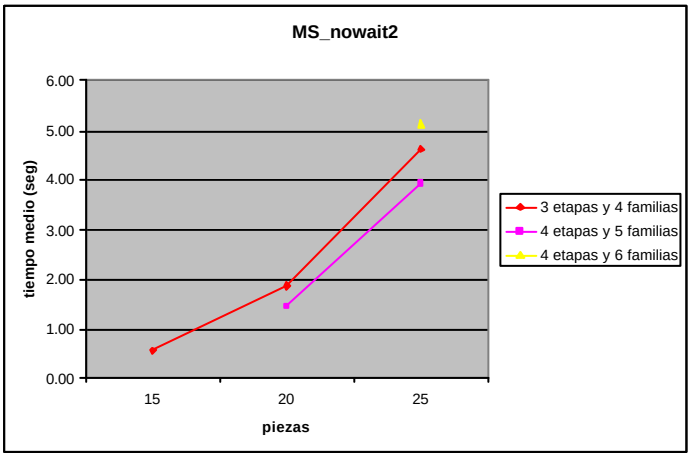

Figura 12. Tiempos medios (segundos) según MS con estrategia ABNW2

En las tablas (Tablas 14,15 y 16) y gráficos anteriores (Figuras 10, 11 y 12) se puede observar que, de nuevo, es el número de piezas la variable que repercute en mayor medida sobre el tiempo computacional. Por esta razón, a la colección 1 (40 ejemplares de 15 piezas) se ha fijado un tiempo de 10 minutos. A la colección 2 (60 ejemplares de 20 piezas) se ha multiplicado el tiempo por 3,2 que es el incremento de tiempo medio requerido para pasar de 15 a 20 piezas siendo el tiempo fijado de 48 minutos y para la colección 3 (100 ejemplares de 25 piezas) se ha multiplicado por 8,2 el tiempo medio concedido por ejemplar que corresponde al incremento medio respecto a un ejemplar de 15 piezas. De esta forma el tiempo concedido a la colección 3 es de 205 minutos.

5.2.2. Calidad de la solución

Para comparar los procedimientos desarrollados se ha ejecutado, 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 $I_{h,heuristica}$ según (6), siendo h el número de ejemplar analizado y *heurística* el procedimiento utilizado para obtener el retraso acumulado.

Además, se ha calculado la media (μ) y la desviación estándar (s) del índice $I_{h,heuristica}$, cuyos valores de cada colección, servirán para comparar los procedimientos entre si.

Tabla 17. Resultados para un el caso general

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)}) \mid\mid \sum T$						
Heurísticas	**Número de Piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	2.11	3.54	6.01	8.69	12.77	30.88
GRASP	**1.05**	**1.84**	**4.58**	**4.87**	**11.09**	**15.48**
Multistart	2.29	3.38	7.68	8.80	16.47	19.83

Los resultados mostrados en la Tabla 17 indican que el método más adecuado es la heurística GRASP. Además se puede apreciar que la desviación estándar aumenta con el número de piezas. Cabe destacar que la heurística Multistar, en este caso, no da buenos resultados quizás debido a que al partir de una solución inicial construida al azar requiera un tiempo mayor de estabilización.

En la Tabla 18 se muestran los resultados obtenidos para el caso *nowait* según la estrategia ABNW. En este caso, aunque la heurística GRASP sigue teniendo los valores menores de la media, la diferencia de éstos con los obtenidos por los procedimientos Multistar y CR es poca.

Tabla 18. Resultados con restricción *nowait* con estrategia ABNW

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum T$ **(ABNW)**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	4.41	6.39	10.13	7.31	10.57	7.32
GRASP	**2.52**	**3.20**	**9.31**	**6.62**	**8.61**	**5.40**
Multistart	2.72	3.04	9.33	7.29	11.41	8.01

Tabla 19. Resultados con restricción *nowait* con estrategia ABNW2

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum T$ **ABNW2**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	4.21	6.72	9.94	9.75	10.02	7.71
GRASP	**2.58**	**4.37**	**7.47**	**5.41**	**10.00**	**8.51**
Multistart	2.63	3.71	8.72	6.23	11.67	8.27

Según los resultados mostrados en la Tabla 19 para el caso *nowait* programando según la estrategia ABNW2 se deduce que la heurística más apropiada sería también la GRASP.

De las tabla 18 y 19 no se puede deducir cual de las dos estrategias es la más adecuada a seguir durante la programación. Para ello se ha calculado el índice $I^*_{h,heuristica}$ según (7) que permite calcular la discrepancia del valor de la media con respecto al valor mínimo conocido para el ejemplar. Además, como en el caso anterior, se ha calculado la media y la desviación estándar de $I^*_{h,heuristica}$ que servirá para detectar la estrategia a utilizar.

Tabla 20. Media y desviación estándar del índice $I^*_{h,heuristica}$ caso ABNW

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum T$ **ABNW**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	9.12	10.46	16.63	10.49	18.91	14.08
GRASP	7.10	7.84	15.94	11.45	16.78	12.08
Multistart	7.21	7.28	15.87	10.04	19.64	12.03

Comparando los resultados mostrados en las tablas 20 y 21 se comprueba que los valores menores de la media se obtienen con la estrategia ABNW2. Así pues la estrategia de

programación ABNW2 es más eficiente que la ABNW. Este resultado era previsible ya que en la estrategia ABNW2, cuando se programa hacia atrás se revisa de nuevo la disponibilidad de las máquinas reasignando el trabajo a la máquina que esté disponible antes, de esta forma que se minimiza los tiempos muertos.

Tabla 21. Media y desviación estándar del índice $I^*_{h,heuristica}$ caso ABNW2

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum T$ **ABNW2**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	4.48	6.75	11.44	11.46	**11.13**	**7.62**
GRASP	**2.85**	**4.43**	**8.90**	**6.43**	11.75	9.76
Multistart	2.90	3.81	10.29	7.27	12.91	8.62

5.3 Caso Bicriterio

A continuación se analiza la experiencia computacional realizada sobre las heurísticas adaptadas al caso bicriterio. En este apartado también se ha dividido el análisis en función de si el sistema productivo tiene o no el mismo número de máquinas en paralelo para cada etapa. Así mismo, dentro de cada tipo de sistema se ha considerado, también, la restricción *nowait* que ha sido tratada con cada una de las estrategias implementadas, ABNW y ABNW2.

Para realizar la experiencia computacional se han utilizado las mismas colecciones, para cada tipo de sistema, que las descritas en el apartado 5.1 y 5.2 respectivamente. El tiempo computacional concedido a las heurísticas se corresponde con el tiempo fijado para cada caso y colección en dichos apartados.

Para comparar los procedimientos desarrollados se ha ejecutado, 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 $I_{h,heuristica}$ según (6), siendo *h* el número de ejemplar analizado y *heurística* el procedimiento utilizado para obtener el retraso acumulado.

Además, se ha calculado la media (μ) y la desviación estándar (s) del índice $I_{h,heuristica}$, cuyos valores de cada colección, servirán para comparar los procedimientos entre si.

En todos los casos analizados se ha tomado ?=0.5

5.3.1 Flow shop híbrido con igual número de máquinas por etapa

La Tabla 22 muestra los resultados obtenidos para el caso general. En ella se puede observar que la heurística GRASP, para los ejemplares con 15 y 20 piezas, es la que proporciona un

valor de la media del índice $I_{h,heuristica}$ menor, en cambio los mejores resultados para los ejemplares con 25 piezas se obtienen con la heurística CR, aunque el valor obtenido con GRASP es muy similar y la diferencia a favor de CR puede haber sido causada por el azar. Cabe apreciar que la desviación estándar obtenida no es muy grande, a excepción de la obtenida en la colección con 25 piezas mediante la heurística Multistar, lo que indica que los valores que ofrecen todas ellas no discrepan demasiado del valor mínimo. La desviación aumenta a medida que aumenta el número de piezas a secuenciar.

Tabla 22. Resultados para el caso general

$FH3,(P3)_{l=1}^{3} \mid s_{jk} \mid l \cdot \sum T + (1-l) \cdot \sum C$						
Heurísticas	**Número de Piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	0.04	0.07	0.71	0.47	**1.00**	**0.63**
GRASP	**0.02**	**0.04**	**0.57**	**0.47**	1.09	0.56
Multistart	0.13	0.58	0.63	0.41	1.96	4.06

Las tablas 23 y 24 muestran los resultados obtenidos cuando se tiene en cuenta la restricción *nowait*. La tabla 22 corresponde a las heurísticas con la estrategia de programación ABNW y la tabla 23 con la estrategia ABNW2. Para ambos casos se aprecia que la heurística GRASP es la que obtiene mejores resultados.

Tabla 23. Resultados para el caso *nowait* con estrategia ABNW

$FH3,(P3)_{l=1}^{3} \mid s_{jk}, nowait \mid l \cdot \sum T + (1-l) \cdot \sum C$ **ABNW**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	0.43	0.59	1.63	0.92	2.40	1.33
GRASP	**0.39**	**0.46**	**1.59**	**0.94**	**1.79**	**1.10**
Multistart	0.39	0.62	1.47	0.81	2.83	3.93

Tabla 24. Resultados para el caso *nowait* con estrategia ABNW2

$FH3,(P3)_{l=1}^{3} \mid s_{jk}, nowait \mid l \cdot \sum T + (1-l) \cdot \sum C$ **ABNW2**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	0.34	0.49	1.41	0.80	**2.04**	**1.24**
GRASP	**0.33**	**0.42**	**1.24**	**0.76**	2.00	1.62
Multistart	0.34	0.70	1.36	0.73	3.43	3.89

Para compara las dos estrategias de programación utilizadas con la restricción *nowait*, ABNW y ABNW2, se calcula el índice $I^*_{h,heuristica}$ según (7). Los valores obtenidos de la media (μ) y la desviación estándar (s) para cada coso se muestran en las tablas 25 y 26. En esta ocasión los valores menores, en las colecciones de 15 y 20 piezas, se obtienen con la estrategia ABNW en cambio para los ejemplares con 25 piezas la estrategia ABNW2 funciona mejor, aunque este resultado puede ser debido al azar. Cabe remarcar que a medida que aumenta el número de piezas a secuenciar los valores obtenidos con ambas estrategias convergen, se debería estudiar el comportamiento para colecciones de piezas mayores. Estos resultados no son los esperados, ya que con la estrategia ABNW2 se revisa la máquina asignada y por lo tanto se tiene la opción de mejorar la solución obtenida.

Tabla 25. Media y desviación estándar del índice $I^*_{h,heuristica}$ caso ABNW

$FH3,(P3)^3_{l=1} \mid s_{jk}, nowait \mid l \cdot \sum T + (1-l) \cdot \sum C$ **(ABNW)**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	1.52	1.46	2.55	1.41	3.67	2.16
GRASP	**1.48**	**1.53**	**2.50**	**1.56**	3.85	2.01
Multistart	1.40	1.52	2.38	1.56	3.91	2.39

Tabla 26. Media y desviación estándar del índice $I^*_{h,heuristica}$ caso ABNW2

$FH3,(P3)^3_{l=1} \mid s_{jk}, nowait \mid l \cdot \sum T + (1-l) \cdot \sum C$ **ABNW2**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	2.54	3.58	3.25	2.73	3.74	2.33
GRASP	2.53	3.63	3.10	2.85	**3.61**	**2.49**
Multistart	2.45	3.67	3.21	2.81	3.96	2.39

5.3.2 Flow shop híbrido con diferente número de máquinas por etapas

La Tabla 27 muestra los resultados obtenidos en el caso general. Se puede observar que la heurística GRASP es la que proporciona un valor de la media del índice $I_{h,heuristica}$ menor. Cabe apreciar que la desviación estándar obtenida no es muy grande lo que indica que los valores que ofrecen todas ellas no discrepan demasiado del valor mínimo. La desviación aumenta a medida que aumenta el número de piezas a secuenciar.

Tabla 27. Resultados para el caso general

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)}) \mid\mid l\sum T+(1-l)\sum C$						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	0.57	0.78	1.69	1.09	2.17	1.59
GRASP	**0.42**	**0.44**	**1.45**	**0.93**	**1.90**	**1.33**
Multistart	0.43	0.67	1.56	1.05	2.06	2.14

En las Tablas 28 y 29 se muestran los resultados obtenidos cuando se tiene en cuenta la restricción *nowait*, con cada una de las estrategias de programación implementadas, ABNW y ABNW2. En la Tabla 28 se observa que los resultados menores de la media se obtienen con la heurística GRASP. Cabe apreciar que con la heurística CR los resultados obtenidos no son muy diferentes, en cambio, con la heurística Multistar éstos son mucho peores.

Tabla 28. Resultados para el caso nowait con la estrategia ABNW

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum l\sum T+(1-l)\sum C$ **ABNW**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	1.63	2.99	2.29	1.36	2.40	1.74
GRASP	**1.31**	**2.70**	**1.83**	**1.12**	**2.17**	**1.37**
Multistart	12.93	8.48	19.71	10.69	22.32	9.61

En la Tabla 29 observamos unos resultados parecidos a los mostrados en la Tabla 27. Por lo tanto, se puede concluir que la heurística GRASP sería la más aconsejable.

Tabla 29. Resultados para el caso nowait con la estrategia ABNW2

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum l\sum T+(1-l)\sum C$ **ABNW2**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	4.00	3.90	2.10	1.46	2.55	1.85
GRASP	**2.02**	**5.00**	**2.03**	**1.35**	**2.26**	**1.36**
Multistart	13.21	8.55	17.43	9.98	21.22	9.92

Para poder compara entre sí las dos estrategias de programación utilizadas con la restricción *nowait* se ha calculado el índice $I^*_{h,heuristica}$ según (7). Los valores de la media y desviación estándar obtenidos se muestran en las tablas 30 y 31. En ellas se pueden apreciar que los mejores valores de la media se obtienen con la heurística GRASP cuando utiliza la estrategia

ABNW2. Para la colección con 20 piezas se obtiene un valor de la media ligeramente menor con la estrategia ABNW pero puede ser debido al azar ya que la discrepancia entre ellos es poca. Se puede apreciar, además, que a medida que aumenta el número de piezas la estrategia ABNW2 se comporta mejor que la ABNW.

Tabla 30. Resultados con restricción *nowait* con estrategia ABNW

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum l \sum T + (1-l)\sum C$ **ABNW**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	3.60	5.65	3.86	2.54	4.34	2.70
GRASP	3.08	6.09	**3.39**	**2.57**	4.11	2.50
Multistart	15.93	8.27	21.76	10.95	24.60	10.05

Tabla 31. Resultados con restricción *nowait* con estrategia ABNW2

$FH3,(PM^{(1)},PM^{(2)},PM^{(3)} \mid s_{jk}, nowait \mid \sum l \sum T + (1-l)\sum C$ **ABNW2**						
Heurísticas	**Número de piezas**					
	n=15		**n=20**		**n=25**	
	μ	s	μ	s	μ	s
CR	4.71	3.91	3.51	3.37	3.23	2.16
GRASP	**2.64**	**5.04**	3.46	3.59	**2.32**	**1.33**
Multistart	14.46	8.24	19.16	9.81	21.91	9.90

6. Conclusiones

Se ha estudiado la programación de la producción en un entorno productivo formado por diferentes etapas en las que puede haber diferentes máquinas idénticas en paralelo (sistemas *flow shop* híbrido). Se ha considerado la existencia de tiempos de preparación dependientes del tipo de pieza a fabricar y se ha analizado, además, la programación de la producción bajo la restricción *nowait*.

Nos hemos interesado por la implementación de procedimientos heurísticos tanto para el caso en que exista un único criterio de programación, aquí el retraso total, como por su adaptación al caso multiobjetivo, aquí suma ponderada del retraso total y el tiempo de finalización de las piezas. Se ha desarrollado un procedimiento de programación general para este tipo de sistemas de forma que dada una secuencia es capaz de asignar y programar las piezas en cada etapa y máquina. Esta forma de proceder ha permitido poder aprovechar aquellos procedimientos heurísticos que han proporcionado mejores resultados en los casos anteriores.

Para ambos caso se han implementado tres heurísticas (CR, GRASP y Multistar) y se ha concluido que la heurística GRASP es la que proporciona mejores resultados. La mezcla de

aleatoriedad y significado de la solución inicial permite a la heurística de mejora comportarse de forma más eficiente.

La restricción *nowait* se ha abordado a través de dos estrategias de programación: ABNW y ABNW2. En el primer caso la máquina asignada en una etapa se mantiene durante la programación hacia atrás en cambio en la segunda estrategia se revisa la máquina por si puede existir alguna otra disponible en el momento adecuado. La segunda estrategia, en general, ha resultado mejor como cabía esperar ya que al tener un grado más de libertad se tiene opción a mejorar la solución obtenida.

Se plantea como futuras líneas de investigación la aplicación de este esquema sobre otro tipo de heurísticas como pueden ser la búsqueda Tabú o los Algoritmos genéticos. Se pretende además estudiar casos en el que las máquinas de una misma etapa no sean idénticas para analizar su repercusión y extender el estudio a casos generales.

Referencias

Aldowaisan, T.; Allahverdi, A. (2004). New heuristics for m-machine no-wait flowshop to minimize total completion time, *Omega*, 32:345-352.

Bertel, S.; Billaut, J.C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation, *European Journal of Operations Research*, 159:651-663.

Brah, S.A.; Hunsucker, J.L. (1991). Branch and Bound algoritm for the flow shop with multiple processors, *European Journal of Operational Research*, 51: 88-99.

Brah, S.A.; Loo, L.L. (1999). heuristics for scheduling in a flow shop with multiple processors, *European journal of Operational Research*, 113: 113-122.

Buten, R.E, Shen, V.Y. (1973). A scheduling model for computer systems with two classes of processors, *Sagomore Computer Conference on Parallel Processing.*

Campbell, H.G.; Dudek, R.A; Smith, M.L. (1970). *A heuristic algorithm for the n-job, m-machine sequencing problem, Management Science*, 16:630-637.

Chen, C.L.; Neppalli, R.V.; Aljaber, N. (1996). Genetic algorithms applied to the continuous flowshop problem. *Computers & Industrial Engineering*, 30(4):919-929.

Fortemps, Ph.; Ost, Ch.; Pirlot, M.; Teghem, J.; Tuyttens, D. (1996). Using metaheuristics for solving a production scheduling problem in a chemical firm. A case study, *International Journal of Production Economics*, 46-47: 13-26.

Grageon, N.; Tangut, A.; Tchernev, N. (1999). Generic simulation model for hybrid flow-shop, *Computers & Industrial Engineering*, 37: 207-210.

Guirchoun, S.; Martineau, P.; Billaut, J.C. (2005). Total completion time minimization in a computer system with a server and two parallel processors, *Computers & Operations Research*, 32:599-611.

Gupta, J.N.D.(1988). Two-stage hybrid flowshop scheduling problem, *Operational Research Society*, 39(4): 359-364.

Kuriyan, k.; Reklaitis, G.V. (1985). Approximate scheduling algorithmd for network flowshops, *Symposium Series No.92*, 79-91.

Kurinyan, K.; Reklaitis, G.V. (1989). Scheduling network flowshops so to Minimize Makespan, *Computers & Chemical Engineering*, 13(1-2):187-200.

Lee, G.C.; Kim, Y.D.; Choi, S.W. (2004). Bottleneck-focused scheduling for a hybrid flowshop, *International Journal of Production Research*, 42(1):165-181.

Lin, H.; Liao C. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines, *International Journal of Production Economics*, 86:133-143
Linn, R.; Zhang, W. (1999). Hybrid flow shop scheduling: a survey, *Computers & Industrial Engineering*, 37: 57-61.
Narasimhan, S.L.; Panwalkar, S.S. (1984). Sacheduling in a two stage manufacturing process, *International Journal of Production Research*, 22(4):555-564.
Negenman, E.G. (2001). Local search algorithms for the multiprocessor flow shop scheduling problem, *European journal of operational research*, 128:147-158
Oguz, C.;Lin, B.M.T.;Cheng T.C.E. (1997).Two-stage flowshop scheduling with a common second-stage machine, *Computers and Operations Research*, 24(12): 1169-1174.
Oguz C.; inder, Y., Do, VH.;Janiak, A. (2004).Hybrid flow-shop scheduling problems with multiprocessor task systems, *European Journal of Operational Research*, 152:115-131.
Paul, R.J. (1979).A production scheduling problem in the glass-container industry, *Operations Research*, 27(2):290-302.
Portmann, M.C.;Vignier, A.;Dardilhac, D.;Dezalay, D. (1998).Branch and Bound crossed with GA to solve hybrid flowshops, *European Journal of Operational Research*, 107: 389-400.
Riane, F.; Meskens, N.; Artiba, A. (1997). Bicriteria scheduling hybrid flowshop problems. In *International Conference on Industrial Engineering and Productions Management (IEPM'97), Fucam*, Pages 34-43, Lyon, France.
Ruiz, R.; Maroto, C. (2006). A genetic Algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of operational Research*, 169:781-800.
Salvador, M.S. (1973). *Proceedings of Symposium on the Theory of scheduling and Its Applications*, 83-91. Springer-Verlag. Berlin.
Santos D.L.; Hunsucker, J.L.; Deal, D.E. (1995). Global lower bounds for flow shops with multiple processors, *European Journal of Operational Research*, 80: 112-120
Santos D.L.; Hunsucker, J.L.; Deal, D.E. (1996). An evaluation of sequencing heuristics in flow shops with multiple processors, *Computers & Industrial Engineering*, 30(4):681-692
Sawik, T.J. (1995). Scheduling flexible flow line with no in-process buffers, *International Journal of productions research*, 33(5):1357-1367.
Shen, V.Y.; Chen, Y.E. (1972). A scheduling strategy for the flowshop problem in a systen with two classes of processors. *In Conference on Information and Systems Science*, 645-649.
Sherali, H.D.; Sarin, S.C.; Kodialam, M.S.(1990). Models and algorithm for a two stage production process, *Production Planning and Control*, 1(1):27-39.
Sriskandarajah, C.; Sethi, S.P. (1989). Scheduling algorithms for flexible flowshops: Worst and average case performance, *European Journal of Operational Research*, 43: 143-160.
Sriskandarajah, C. (1993). Performance of scheduling algorithms for no-wait flowshops with parallel machines, *European Journal of Operational Research*, 70; 365-378.
Thornton, H.W.; Hunsucker, J.L. (2004). A new heuristic for minimal make span in flow shops with multiple processors and no intermediate storage, *European Journal of Operational Research*, 152:96-114
T'kindt, V.; Billaut, J.C. (2002), Multicriteria Scheduling theory, Models and Algorithms, *Springer-Verlag*, pages 263-269, Berlin, Heidelberg.
Vignier, A.; Billaut, J.C.; Proust, C. (1999). Les problèmes d'ordonnancement de type flowshop hybride : État de l'art, *RAIRO Recherche operationnelle*, 33(2):117-183.
Wardono, B.; Fathi, Y. (2004). A tabu search algorithm for the multi-satage parallel machine problem with limited buffer capacities, *European Journal of Operational Research*, 155:380-401.