UCLM
UNIVERSIDAD DE CASTILLA-LA MANCHA

MAEB

VIII CONGRESO ESPAÑOL SOBRE METAHEURÍSTICA, ALGORÍTMOS EVOLUTIVOS Y BIOINSPIRADOS

ALBACETE, 8 al 10 FEBRERO, 2012

EDITORES:

José Antonio Gámez Martín
José Miguel Puerta Callejón
Francisco Parreño Torres
Luis de la Ossa Jiménez

ISBN: 978-84-615-6931-1

AYUNTAMIENTO DE ALBACETE
CONCEJALÍA DE CULTURA

MINISTERIO DE CIENCIA E INNOVACIÓN

I3A Instituto de Investigación en Informática de Albacete

VIAJES El Corte Inglés

DSI Departamento de Sistemas Informáticos

Título:	VIII CONGRESO ESPAÑOL SOBRE METAHEURÍSTICAS, ALGORITMOS EVOLUTIVOS Y BIOINSPIRADOS.
ISBN:	978-84-615-6931-1.
Patrocina:	Ministerio de Ciencia e Innovación (TIN2011-15220-E).
Edita:	José A. Gámez, José M. Puerta, Francisco Parreño y Luis de la Ossa. Universidad de Castilla-La Mancha.

Algoritmos GRASP para el MMSP-W con estaciones en serie y libre interrupción de operaciones

Joaquín Bautista, Alberto Cano, Rocío Alfaro

Resumen— **Se presenta una variante del problema de secuenciación en líneas de montaje de productos mixtos (*MMSP-W: Mixed-Model Sequencing Problem with Workoverload Minimization*) con estaciones en serie y sin restricciones en los instantes de interrupción de las operaciones, con el objetivo de minimizar la sobrecarga de trabajo. Para resolver el problema, se proponen 7 algoritmos basados en el esquema metaheurístico *Greedy Randomized Adaptive Search Procedure* (*GRASP*), que se aplican a una colección de 225 ejemplares recogidos de la literatura. Los resultados obtenidos mediante *GRASP* son computacionalmente competitivos con los conseguidos a través de la programación lineal entera mixta y se acercan a los proporcionados por la programación dinámica acotada.**

Palabras clave— **MMSP-W, Secuenciación, Sobrecarga, GRASP, Programación lineal.**

I. Introducción

Las líneas de fabricación de productos mixtos, muy frecuentes en los entornos *Just-in-time* (*JIT*) y *Douki Seisan* (*DS*), permiten tratar diversas variantes de uno o más productos. Esta flexibilidad condiciona el orden en que se han de tratar las unidades para, por una parte, conseguir la reducción drástica de stocks intermedios, y, por otra, aprovechar al máximo el tiempo disponible para la fabricación.

En estos entornos, podemos encontrar dos categorías de objetivos básicos [1]:

A. *Vinculados al sobreesfuerzo o trabajo perdido*: Reducción al mínimo de las sobrecargas de trabajo que pueden aparecer por programas de producción con productos mixtos, debido a la duración no homogénea de los tiempos de proceso de las operaciones asociadas a distintos tipos de producto incluidos en dichos programas.
B. *Vinculados a JIT*: Reducción al mínimo de los niveles de stock en el sistema.

En cuanto a la categoría A de objetivos, además del enfoque relativo a maximizar el trabajo total completado [2], cabe la posibilidad de modular el esfuerzo adicional que debe aplicarse a lo largo del tiempo sobre algunas operaciones [3].

Bajo esta perspectiva, los problemas de secuencias pueden ser agrupados en tres categorías:

1. *Mixed-model sequencing*
2. *Car sequencing*
3. *Level scheduling*

Nissan Chair UPC, Barcelona, España. E-mail: joaquin.bautista@upc.edu, alberto.cano-perez@upc.edu, rocio.alfaro@upc.edu .

El presente trabajo se puede enmarcar en la categoría A.1. y adopta como criterio de optimización la minimización de la sobrecarga total de trabajo.

La sobrecarga, o sobreesfuerzo, es una medida, en unidades de tiempo, del trabajo que no se puede completar, al ritmo de la actividad estándar establecida, en el tiempo (*ciclo*) concedido a las estaciones de trabajo. Esta sobrecarga puede aparecer cuando el tiempo de proceso de una unidad, en una estación de trabajo, es mayor que el tiempo de ciclo [2]; incluso, la sobrecarga puede aparecer cuando concedemos a cada estación algo más de tiempo para procesar cada unidad de producto; se tiene entonces un ciclo ampliado, llamado ventana temporal, que es el tiempo máximo que puede permanecer una unidad en cada estación de trabajo.

Ante una sobrecarga previsible en una estación, se pueden adoptar, al menos, tres tipos de medidas:

I. Parar la línea y completar el trabajo pendiente con algún refuerzo [4], [5].
II. Dejar pasar la unidad y concluir, posteriormente, el trabajo pendiente en una línea final [2], [6], [7], [8].
III. Incrementar la actividad productiva por encima de la estándar, mediante la asistencia de operarios de refuerzo o sistemas robotizados programados previamente.

Este trabajo contempla las medidas de las categorías II y III para tratar la sobrecarga.

El *MMSP-W* es un problema *NP-hard* [2] para cuya resolución se han propuesto diferentes alternativas; que incluyen procedimientos exactos basados en *Branch and Bound* [9], programación dinámica [2], [10], procedimientos heurísticos basados en búsqueda local [6], [11], algoritmos *Greedy* con reglas de prioridad [6], [12], metaheurísticas [13], [14], o basados en *Beam Search* [15].

Para este trabajo se ha diseñado un procedimiento *GRASP extendido* (*GRASP-x*) que admite diversas variantes en función de los valores asignados a tres parámetros. Entre dichas variantes se encuentran las heurísticas *Greedy* constructivas con posterior optimización local, los procedimientos *Multistart*, los algoritmos *GRASP clásicos* [16], [17] y los algoritmos *GRASP* en los que la probabilidad de selección de los candidatos se hace depender de la aptitud de éstos, la cual puede medirse a través de una función de calidad dependiente de cada candidato. Una extensa recopilación de trabajos sobre aplicaciones de estos

algoritmos se puede encontrar en [18], [19].

Los resultados que se han obtenido con el procedimiento *GRASP extendido* son computacionalmente competitivos con la programación lineal entera mixta y, en menor grado, con la programación dinámica acotada. Nuestra propuesta contiene lo siguiente:

1. Un modelo para el *MMSP-W* con estaciones en serie y libre interrupción de las operaciones, que parte de la base proporcionada por los modelos de Yano [2] y Scholl [13].
2. El diseño e implementación de cotas parciales y globales para el problema, basadas en la programación lineal.
3. Siete algoritmos basados en el procedimiento *GRASP* que actúan como maestros para dirigir la exploración en el espacio de búsqueda y cuentan con la asistencia de la programación lineal para determinar los índices de selección de candidatos. Para ello, se ha empleado el solver *Gurobi versión 4.5.0*.
4. Una experiencia computacional, con ejemplares de la literatura, para comparar los resultados obtenidos mediante los procedimientos implementados en este trabajo con los resultados ofrecidos por la programación lineal entera mixta y la programación dinámica acotada.

Este trabajo se organiza de la siguiente forma. La sección II presenta un modelo para el *MMSP-W* con estaciones de trabajo en serie y libre interrupción de operaciones. La sección III contiene el diseño de cotas parciales y globales para el problema. La sección IV muestra un ejemplo ilustrativo. En la sección V se describe el procedimiento *GRASP-x* adaptado para resolver el *MMSP-W*. La sección VI se centra en la descripción y comparación de resultados de la experiencia computacional realizada, que explota siete algoritmos (*derivados del procedimiento GRASP-x tras fijar siete conjuntos de valores a los tres parámetros*) sobre ejemplares de la literatura. Finalmente, la sección VII muestra algunas conclusiones sobre el presente trabajo.

II. Modelo para el *MMSP-W* con estaciones de trabajo en serie y sin reglas de interrupción de las operaciones

Para el *MMSP-W* con vínculos entre estaciones de trabajo, libre interrupción de las operaciones y con el objetivo de minimizar la sobrecarga de trabajo, tomamos como referencia el modelo *M4*, propuesto en [10].

El modelo extendido *M4'*, propuesto en [20], se centra en minimizar la sobrecarga total (*es decir, maximizar el trabajo total completado*), utiliza los instantes relativos de inicio de las unidades y, además, considera más de un procesador homogéneo en cada estación de trabajo.

Los parámetros y las variables de este modelo se presentan a continuación.

Parámetros

- K Conjunto de estaciones de trabajo ($k = 1, .., |K|$).
- b_k Número de procesadores homogéneos en cada estación de trabajo ($k = 1, .., |K|$).
- I Conjunto de tipos de producto ($i = 1, .., |I|$).
- d_i Demanda programada del tipo de producto i.
- $p_{i,k}$ Tiempo de proceso requerido a cada procesador homogéneo (*en actividad normal*), por una unidad de producto de tipo i en la estación k.
- T Demanda total. Obviamente $\sum_{i=1}^{|I|} d_i = T$.
- t Indice de posición en la secuencia $t = 1, .., T$.
- c Tiempo de ciclo. Tiempo estándar asignado a cada procesador de las estaciones de trabajo, $k = 1, .., |K|$, para tratar cualquier unidad de producto.
- l_k Ventana temporal. Tiempo máximo que se le permite, a cada procesador de la estación k, trabajar en cualquier unidad de producto; siendo $l_k - c > 0$ el tiempo máximo que se puede retener una unidad de producto, en la estación k, una vez concluido el ciclo.

Variables

- $x_{i,t}$ Variable binaria que adopta el valor 1 si una unidad del producto i ($i = 1, ..., |I|$) se asigna a la posición t ($t = 1, ..., T$) de la secuencia, y valor 0 en caso contrario.
- $s_{k,t}$ Instante de inicio del trabajo aplicado a la *t-ésima* unidad de la secuencia de productos en la estación k ($k = 1, ..., |K|$).
- $w_{k,t}$ Sobrecarga generada en cada procesador homogéneo (*en actividad normal)*, medida en tiempo, por la *t-ésima* unidad de producto secuenciada en la estación de trabajo k.
- $\hat{s}_{k,t}$ Diferencia positiva entre el instante de inicio real y el mínimo instante de inicio de la *t-ésima* operación en la estación de trabajo k. $\hat{s}_{k,t} = [s_{k,t} - (t-1)c]^+$ (con $[x]^+ = max\{0, x\}$).
- $\rho_{k,t}$ Tiempo de proceso requerido a cada procesador por la *t-ésima* unidad de la secuencia de productos en la estación de trabajo k.

Modelo *M4'*

$$\text{mín } W = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right) \quad (1)$$

sujeto a:

$$\sum_{t=1}^{T} x_{i,t} = d_i \qquad \forall i = 1, .., |I| \quad (2)$$

$$\sum_{i=1}^{|I|} x_{i,t} = 1 \qquad \forall t = 1, .., T \quad (3)$$

$$\rho_{k,t} = \sum_{i=1}^{|I|} p_{i,k} x_{i,t} \qquad \forall k = 1,..,|K|; \ \forall t = 1,..,T \qquad (4)$$

$$\rho_{k,t} - w_{k,t} \geq 0 \qquad \forall k = 1,..,|K|; \ \forall t = 1,..,T \qquad (5)$$

$$\hat{s}_{k,t} \geq \hat{s}_{k,t-1} + \rho_{k,t-1} - w_{k,t-1} - c \qquad \forall k = 1,..,|K|; \ \forall t = 2,..,T \qquad (6)$$

$$\hat{s}_{k,t} \geq \hat{s}_{k-1,t} + \rho_{k-1,t} - w_{k-1,t} - c \qquad \forall k = 2,..,|K|; \ \forall t = 1,..,T \qquad (7)$$

$$\hat{s}_{k,t} + \rho_{k,t} - w_{k,t} \leq l_k \qquad \forall k = 1,..,|K|; \ \forall t = 1,..,T \qquad (8)$$

$$\hat{s}_{k,t} \geq 0 \qquad \forall k = 1,..,|K|; \ \forall t = 1,..,T \qquad (9)$$

$$w_{k,t} \geq 0 \qquad \forall k = 1,..,|K|; \ \forall t = 1,..,T \qquad (10)$$

$$x_{i,t} \in \{0,1\} \qquad \forall i = 1,..,|I|; \ \forall t = 1,..,T \qquad (11)$$

$$\hat{s}_{1,1} = 0 \qquad (12)$$

En el modelo, la función objetivo (1) representa la sobrecarga total (W). La restricciones (2) requieren que la demanda programada quede satisfecha. La restricciones (3) indican que sólo una unidad de producto se puede asignar a cada posición de la secuencia. La restricciones (4) vinculan el tiempo de proceso requerido por el tipo de producto con los tiempos de proceso requeridos por las unidades de la secuencia. La restricciones (5) establecen límites superiores para los valores de la sobrecarga a través de los tiempos de proceso requeridos por las unidades de la secuencia. Las restricciones (6) - (9) constituyen el conjunto de los instantes iniciales relativos de las operaciones en cada estación de trabajo y los tiempos de proceso aplicados a los productos. La restricciones (10) indican que las sobrecargas generadas no pueden ser negativas. La restricciones (11) establecen que las variables de asignación son binarias. Por último, la restricción (12) fija el inicio de las operaciones.

III. Cotas globales y parciales para *M4'*

Dada la secuencia parcial $\pi(t) = \{\pi_1, \pi_2, ..., \pi_t\}$, se podrá determinar una cota global de W y una cota parcial del complemento $R(\pi(t))$ asociado a la secuencia o segmento $\pi(t)$ según el esquema de la figura 1.

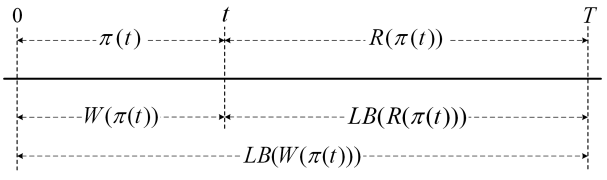

Fig. 1. Esquema de cotas para una secuencia parcial $\pi(t)$

Para obtener las cotas de las sobrecargas asociadas a $\pi(t)$ y $R(\pi(t))$, se imponen las siguientes condiciones a *M4'*:

1. Se fijan los valores de las variables $x_{i,\tau}$ ($i = 1, ..., |I|$; $\tau = 1, ..., t$) en consonancia con $\pi(t)$; esto es:

$$x_{i,\tau} = \left\{ \begin{array}{lll} 1, & si\ \pi_\tau = i & \Leftrightarrow \quad x_{\pi_\tau,\tau} = 1 \\ 0, & si\ \pi_\tau \neq i & \Leftrightarrow \quad x_{i,\tau} = 0, si\ i \neq \pi_\tau \end{array} \right\} \qquad (13)$$

2. Se relaja la condición de binariedad en las variables $x_{i,\tau}$ ($i = 1, ..., |I|; \tau = t+1, ..., T$)

$$0 \leq x_{i,\tau} \leq 1 \qquad i = 1, ..., |I|;\ \tau = t+1, ..., T \qquad (14)$$

El resultado es el siguiente programa lineal *LB_M4'*

$$\text{mín } LB(W(\pi(t))) = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right) \qquad (15)$$

sujeto a:

(2)-(10) *y* (12) *de M4'*

$$x_{\pi_\tau,\tau} = 1 \qquad \forall \tau = 1, ..., t \qquad (16)$$

$$0 \leq x_{i,\tau} \leq 1 \qquad \forall i = 1, ..., |I|; \forall \tau = t+1, ..., T \qquad (17)$$

El programa lineal anterior nos ofrecerá una cota global de la sobrecarga total ($LB(W(\pi(t)))$), así como un valor de la sobrecarga asociada al segmento $\pi(t)$ (o sea, $W(\pi(t))$) y una cota de la sobrecarga asociada al complemento $R(\pi(t))$ (es decir, $LB(R(\pi(t)))$). Estos valores pueden calcularse de la siguiente manera:

$$W(\pi(t)) = \sum_{k=1}^{|K|} \left(b_k \sum_{\tau=1}^{t} w_{k,\tau} \right) \qquad (18)$$

$$LB(R(\pi(t))) = \sum_{k=1}^{|K|} \left(b_k \sum_{\tau=t+1}^{T} w_{k,\tau} \right) \qquad (19)$$

Los instantes relativos de finalización ($\hat{e}_{k,t}$) de la última operación de la secuencia parcial $\pi(t)$, en cada estación de trabajo, se pueden obtener a partir de *LB_M4'*, de la forma: $\hat{e}_{k,t} = \hat{s}_{k,t} + \rho_{k,t} - w_{k,t}$, $\forall k$.

IV. Un ejemplo ilustrativo

Con el propósito de ilustrar el problema a través del modelo anteriormente formulado, presentamos el siguiente ejemplo.

Se dispone de 6 unidades de producto ($T = 6$), de las cuales 3 son de tipo A, 1 es de tipo B y 2 son de tipo C, con un trabajo total requerido igual a $V_0 = 104$. Las unidades de producto se procesan en 3 estaciones de trabajo ($|K| = 3$), con diferente número de procesadores (b_k), siendo los tiempos de proceso para cada procesador (*en actividad normal*), para cada unidad de tipo de producto (A, B, C) y en cada estación de trabajo (m_1, m_2, m_3), los recogidos en la tabla I.

Se considera, además, $c = 4$ (*tiempo de ciclo*) y $l_k = 6$ para $k = 1, ..., 3$ (*ventana temporal*). La figura 2 muestra el diagrama de Gantt de la solución óptima dada por el modelo *M4'*. La secuencia de productos que presenta la mínima sobrecarga total es $C - B - A - C - A - A$. El trabajo total completado es $V = 101$, y la sobrecarga, concentrada en las estaciones de trabajo m_1 y m_2, es $W = 3$ (el área gris en la figura 2).

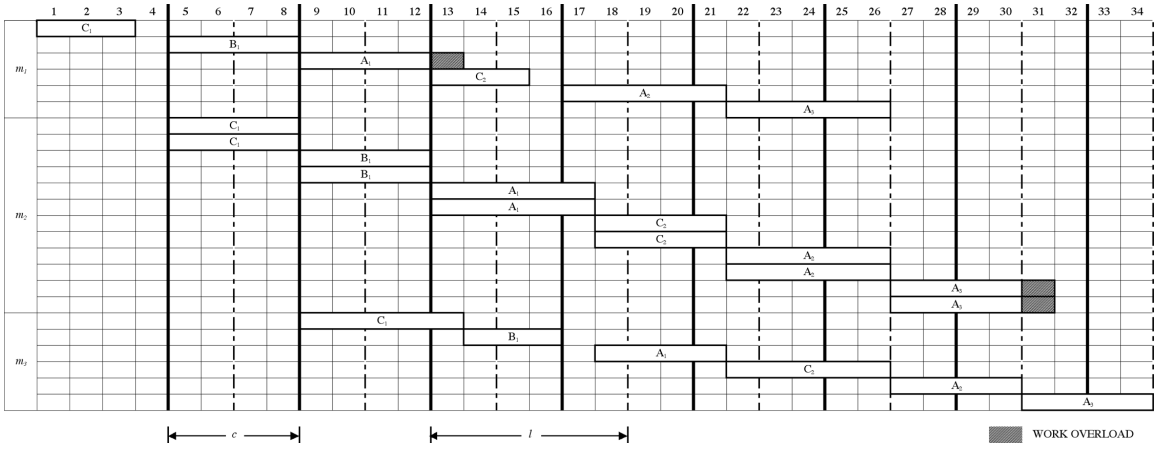

Fig. 2. Solución óptima ofrecida por *M4'*

TABLA I
TIEMPOS DE PROCESO ($p_{i,k}$), NÚMERO DE PROCESADORES HOMOGÉNEOS (b_k) Y TRABAJO TOTAL (V_0) REQUERIDO POR CADA TIPO DE UNIDAD DE PRODUCTO EN CADA ESTACIÓN DE TRABAJO

	A ($d_A = 3$)	B ($d_B = 1$)	C ($d_C = 2$)	b_k
m_1	5	4	3	1
m_2	5	4	4	2
m_3	4	3	5	1
Total	19 $V_{0_A} = 57$	15 $V_{0_B} = 15$	16 $V_{0_C} = 32$	$V_0 = 104$

V. ALGORITMOS *GRASP* PARA EL *MMSP-W*

Esta sección presenta los elementos básicos del procedimiento *GRASP* aplicado a la resolución del *MMSP-W* con estaciones de trabajo en serie y con libre interrupción de las operaciones.

A. Preliminares

La complejidad del *MMSP-W* con vínculos entre estaciones de trabajo y el interés de obtener soluciones para ejemplares del problema con dimensiones industriales, hace recomendable el uso de procedimientos heurísticos capaces de ofrecer soluciones aceptables con bajo esfuerzo de computación.

En la literatura se pueden encontrar numerosas metaheurísticas dedicadas a la resolución del *MMSP-W* original [6], [11], [12], [14]; en cuanto a la variante del problema que nos ocupa, ha sido tratada con técnicas basadas en programación lineal entera mixta y con programación dinámica.

La metaheurística *GRASP* [16], [17] es de tipo multi-arranque y está provista de 2 fases en cada iteración: (1) un procedimiento *Greedy* que sirve para construir una solución aceptable sin que sea preciso alcanzar el óptimo global; y (2) una segunda fase para obtener un óptimo local dentro de un vecindario y teniendo como punto de partida la solución que resulta al aplicar el procedimiento *Greedy* de la fase_1. Obviamente, la solución ofrecida por el *GRASP* es la mejor entre las obtenidas en el conjunto de iteraciones.

Para la primera fase *Greedy* es importante definir un buen procedimiento que pueda ofrecer soluciones aceptables y una diversidad suficiente de soluciones que permitan explorar diferentes regiones en el espacio de soluciones. Para garantizar dicha diversidad se emplea el azar, de manera que, el siguiente elemento a añadir a una solución parcial se sortea entre los elementos de una lista restringida de candidatos (*RCL*); dicha lista contiene los candidatos que presentan los mejores valores en relación a una función (*una cota, por ejemplo*) diseñada para la selección.

Para resolver un problema de optimización mediante un procedimiento *GRASP* es preciso definir los siguientes elementos:
- El proceso aleatorio empleado en la selección entre candidatos y el procedimiento *Greedy*.
- El vecindario de una solución y, lógicamente, el procedimiento para explorar dicho vecindario.
- El criterio de finalización del algoritmo, normalmente vinculado a un máximo número de iteraciones o al tiempo de ejecución.

En la figura 3 se presenta un esquema general del algoritmo *GRASP*.

1. Inicialización
2. Para toda iteración $iter = 1, ..., iter_{max}$
   2.1. Solución ⟵ Fase_constructiva (*Semilla*)
   2.2. Solución ⟵ Mejora_local (*Solución*)
   2.3. Actualizar_Solución (*Solución, Mejor_solución*)
3. Salida: *Mejor_solución*

Fig. 3. Esquema general de la metaheurística *GRASP*.

En algunos casos, al procedimiento *GRASP*, se le puede añadir un post-proceso que permita combinar las soluciones generadas [21] o también un mecanismo de auto-adaptación de los parámetros del algoritmo durante el proceso de búsqueda [22].

B. Fase Greedy de construcción de una solución

El procedimiento implementado para esta fase del algoritmo (*ver figura 4*) construye progresivamente una secuencia seleccionando, en cada etapa asociada con el instante, $t = 1, ..., T$, un elemento candidato a partir de una lista restringida de éstos (*sea RCL*).

En efecto, llegados a la etapa t, en la que se dispone de una secuencia (*solución*) parcial $\pi(t)$, para cada producto i con demanda pendiente aún, se determina el índice f_i ($\forall i : x_i < d_i$) a partir del valor de la cota $LB(W(\pi(t) \cup \{i\}))$; tras ello, la lista de candidatos se construye en 2 pasos:

(a) En el primero, a partir del parámetro $Z \in [0, 1]$ denominado impedancia, se seleccionan todos los productos con un valor de cota no superior a $1/Z$ veces el valor de la menor de ellas (*cota correspondiente al mejor candidato*).

(b) En el segundo paso, se seleccionan como máximo los L mejores candidatos (*ordenados por cota, de menor a mayor*) incluyendo en la lista, claro está, los productos empatados en cota con el L-*ésimo* candidato.

La selección del tipo de producto a secuenciar se realiza aleatoriamente con una probabilidad de selección, g_i, que se hace depender de la calidad de las soluciones parciales (*medida a través del índice* f_i). Además, se ha incorporado un mecanismo para modular dichas probabilidades mediante la utilización de dos parámetros: (1) el coeficiente de elasticidad aditiva, f_0, que puede corresponderse con el valor de una solución inicial y permite la concentración relativa de los valores de la aptitud de los productos candidatos; y (2) el coeficiente de elasticidad potencial, η, que sirve para concentrar ($\eta \in [0, 1]$) o dispersar ($\eta > 1$) los valores de la aptitud en términos absolutos.

Nótese que la formulación propuesta en la figura 4 comprende como casos particulares los siguientes procedimientos: (1) los algoritmos *GRASP* tradicionales con tratamiento de empates incorporado, pues basta hacer $Z \to 0$, $f_0 \to \infty$, $\eta = 1$ y fijar un valor de L menor que el número de candidatos; (2) *Multistart*, haciendo $Z \to 0$, $f_0 \to \infty$, $\eta = 1$ y fijar un valor de L igual al número de candidatos; y (3) las heurísticas *Greedy* con tratamiento de empates con $Z = 1$. Nótese además que si $Z \to 0$ y L es suficientemente grande ($L = |I|$) todos los elementos compatibles son candidatos y toda solución del *MMSP-W* tiene una probabilidad no nula de ser generada por el procedimiento.

0. Inicialización

Leer: T, I, K, $d_i(\forall i)$, l_k, $b_k(\forall k)$, $p_{i,k}(\forall i, \forall k)$, c, Z, L, f_0, η

Hacer:

$t = 0$

$x_i = 0$ $\forall i \in I$, siendo x_i el número de unidades de tipo i secuenciadas hasta el instante t.

$\pi(t) = \{\varnothing\}$ siendo $\pi(t) = \{\pi_1, \pi_2, ..., \pi_t\}$ la secuencia parcial construída hasta el instante t.

1. Cálculo del índice f

$\forall i \in I$: $x_i < d_i$, determinar:

$f_i = LB^*(W(\pi(t) \cup \{i\}))$
siendo $LB^*(W(\pi(t) \cup \{i\}))$ el valor de la solución óptima ofrecida por el programa lineal *LB_M4'* dada la secuencia parcial $\pi(t) \cup \{i\}$.

2. Creación de la lista de candidatos *RCL*

Sea $f^* = \min_{x_i < d_i} \{f_i\}$;

$$RCL(f) = \{i \in I : (x_i < d_i) \wedge \wedge (f_i \leq \min\{f_0, f^*/Z\}\}$$

donde $Z \in [0, 1]$ es la impedancia sobre el conjunto de elementos compatibles y f_0 es la elasticidad aditiva que que se puede corresponder con el valor de una solución de referencia:
- Si $|RCL(f)| \leq L \Rightarrow RCL = RCL(f)$
- Si $|RCL(f)| > L$:

Sea $i_L \in RCL(f)$ el tipo de producto que ocupa la *L-ésima* posición en la lista $RCL(f)$ ordenada no-decrecientemente respecto al índice f. Hacer:

$$RCL = \{i \in RCL(f) : f_i \leq f_L\}$$

3. Selección del tipo de producto a secuenciar

$\forall i \in RCL$, determinar:

$$g_i = \frac{(f_0 - f_i)^\eta}{\sum_{j \in RCL} (f_0 - f_j)^\eta}$$

donde η es la elasticidad potencial.

Seleccionar, por sorteo, con probabilidades g_i $\forall i \in RCL$ un tipo de producto; sea i^* el resultado de esta selección.

4. Actualización

$x_{i^*} \leftarrow x_{i^*} + 1$; $t \leftarrow t + 1$; $\pi_t = i^*$

5. Finalización

Si $t < T = \sum_{i \in I} d_i$, ir a paso-1.

Si no, finalizar.

Fig. 4. Fase constructiva *GRASP* para el *MMSP-W*.

C. Fase de mejora local

Se propone una mejora local exhaustiva tipo 2-intercambio entre dos elementos de la secuencia en curso de mejora; el intercambio tentativo se produce, lógicamente, cuando dichos elementos corresponden a tipos de productos distintos en el problema *MMSP-W*. La exploración del vecindario es determinista y

se realiza de izquierda a derecha.

A partir de una secuencia en curso, $\pi_c(T)$, con valor $LB^*(W(\pi_c(T))$ obtenido a partir del programa lineal *LB_M4'*, se genera una secuencia vecina, $\pi_v(T)$, mediante un 2-intercambio tentativo entre los elementos que ocupan las posiciones t y t' de la secuencia en curso. Si $LB^*(W(\pi_v(T)) < LB^*(W(\pi_c(T))$, el intercambio se consolida, $\pi_v(T)$ se convierte en la nueva secuencia en curso y se reinicia el proceso de intercambios. En caso contrario, se prosigue con la generación de una nueva secuencia tentativa, una secuencia vecina, mediante otro 2-intercambio tentativo. El procedimiento finaliza cuando ninguna solución vecina tiene un valor de sobrecarga menor que el de la solución en curso.

VI. Experiencia computacional

Se ha realizado una experiencia computacional que emplea 225 ejemplares del problema *MMSP-W* recogidos en la literatura (*ver* [6], [10]). Dichos ejemplares se construyen a partir de 45 programas de producción y 5 estructuras de tiempos de proceso de las operaciones. Todos los ejemplares presentan 4 tipos de productos distintos ($|I| = 4$) y 4 estaciones de trabajo en serie ($|K| = 4$). Las soluciones óptimas de los 225 ejemplares se han obtenido empleando el solver *CPLEX v11.0* (*para un procesador*). Dichas soluciones se comparan con los resultados ofrecidos por 7 algoritmos derivados del procedimiento general *GRASP-x* para el que se han fijado los valores de los parámetros Z, L, f_0 y η (*ver tabla* II), siendo f_{BDP} el valor de referencia, para cada ejemplar, ofrecido por el procedimiento basado en la programación dinámica acotada descrito en [10].

Los algoritmos resultantes tras asignar valores a los 4 parámetros son: (*G*) una heurística *Greedy* constructiva con posterior mejora local; (*M*) un procedimiento *Multistart* tradicional con mejora local; (*GR-01/2*) un algoritmo *GRASP* tradicional con una lista *RCL* limitada a 2 candidatos; (*GR-5/2*) un algoritmo *GRASP* con lista limitada a 2 candidatos y con probabilidades de selección de éstos levemente dependientes de su aptitud; (*GR-9/2*) un algoritmo *GRASP* con alta dependencia entre las probabilidades de selección de los candidatos y su aptitud y con lista restringida a 2 candidatos; y dos algoritmos *Multistart*, (*GR-5/4*) y (*GR-9/4*) con probabilidades de selección de los candidatos idénticas a (*GR-5/2*) y (*GR-9/2*), respectivamente. En todos los algoritmos se fija el número máximo de iteraciones ($iter_{max} = 10$), impuesto a cada uno de los 7 algoritmos en el proceso de resolución de cada ejemplar. Para obtener los valores de las soluciones óptimas de *LB_M4'* y calcular el índice f (ver figura 4, paso-1), se ha empleado el solver *Gurobi v4.5.0.*

Los procedimientos han sido programados en *gcc v. 4.2.1*, en un ordenador Macintosh iMac con un procesador Intel Core i7, 2.93 Ghz. y 8 Gb de memoria RAM, usando MAC OS X 10.6.8 como sistema operativo. Ni la implementación ni el compilador hacen uso de *threads* ni de otra forma de código paralelo, y, por tanto, el ordenador actúa como un único procesador a 2.93 GHz.

TABLA II
Algoritmos derivados de *GRASP-x*. Características.

Alg.	Z	L	f_0	η
G	1	1	f_{BDP}	1
M	0.01	4	f_{BDP}/Z	1
GR-01/2	0.01	2	f_{BDP}/Z	1
GR-5/2	0.5	2	f_{BDP}/Z	1
GR-9/2	0.9	2	f_{BDP}/Z	1
GR-5/4	0.5	4	f_{BDP}/Z	1
GR-9/4	0.9	4	f_{BDP}/Z	1

Los resultados del experimento se resumen en las tablas III, IV y V.

En la tabla III se recoge: (1) el número de óptimos alcanzado (*#opt*) por cada uno de los 7 algoritmos sobre los 225 ejemplares del *MMSP-W*; (2) el promedio de la *desviación porcentual relativa* (RPD=((*Solución* - *Óptimo*)/*Óptimo*)×100), para los ejemplares y cada algoritmo, tanto para la fase_1 *Greedy* constructiva del *GRASP* ($\overline{RPD_1}$) como para el proceso completo ($\overline{RPD_2}$) que incluye el procedimiento de mejora local; y (3) el tiempo medio de *CPU*, por ejemplar, requerido por una iteración de cada uno de los 7 algoritmos *GRASP* ($\overline{CPU}$).

La tabla IV muestra, para cada uno de los 5 bloques de ejemplares formados a partir de familias de los 45 programas de producción (*ver* [6]), los valores de $\overline{RPD_2}$, promediados por bloque, que ofrece cada uno de los 7 algoritmos.

En la tabla V se recogen, para cada una de las 5 estructuras de tiempos de proceso de las operaciones, los valores de $\overline{RPD_2}$, promediados por estructura, que ofrece cada uno de los 7 algoritmos.

TABLA III
Número de óptimos, $\overline{RPD_1}$, $\overline{RPD_2}$ y $\overline{CPU}$ obtenidos a partir de los 7 algoritmos

	#opt	$\overline{RPD_1}$	$\overline{RPD_2}$	$\overline{CPU}$
G	103	9.92	2.46	0.29
M	151	28.81	1.06	0.30
GR-01/2	*159*	15.71	*0.87*	0.31
GR-5/2	150	14.34	1.00	0.31
GR-9/2	155	*7.67*	0.95	0.28
GR-5/4	155	24.89	0.89	0.31
GR-9/4	152	8.68	0.91	0.27

En la tabla III se observa que el algoritmo que obtuvo peores resultados en cuanto a óptimos alcanzados es el *G* (*103 óptimos sobre 225 ejemplares*), mientras que *GR-01/2* fue la heurística con mayor

TABLA IV
$\overline{RPD_2}$ OBTENIDA POR LOS 7 ALGORITMOS PARA CADA BLOQUE DE PROGRAMAS DE PRODUCCIÓN

	B1	*B2*	*B3*	*B4*	*B5*
G	0.37	2.02	4.02	2.47	2.46
M	0.22	0.65	2.37	1.02	0.93
GR-01/2	0.04	*0.54*	*1.61*	0.74	0.91
GR-5/2	0.04	0.89	2.53	*0.55*	0.81
GR-9/2	*0.00*	0.71	2.04	0.72	0.89
GR-5/4	0.04	0.87	1.93	0.56	*0.79*
GR-9/4	*0.00*	0.70	1.85	0.90	0.85

TABLA V
$\overline{RPD_2}$ OBTENIDA POR LOS 7 ALGORITMOS PARA CADA ESTRUCTURA DE TIEMPOS DE PROCESO DE LAS OPERACIONES

	E1	*E2*	*E3*	*E4*	*E5*
G	4.18	3.26	1.67	0.08	3.10
M	2.19	*1.07*	0.53	*0.00*	1.52
GR-01/2	1.79	1.18	*0.33*	*0.00*	1.07
GR-5/2	2.32	1.10	0.42	*0.00*	1.16
GR-9/2	1.87	1.18	0.42	*0.00*	1.26
GR-5/4	1.76	1.29	0.43	*0.00*	*0.98*
GR-9/4	*1.74*	1.31	0.35	*0.00*	1.17

número de éxitos (*159 sobre 225*). En cuanto a la calidad de las soluciones, medida a través de $\overline{RPD_2}$, se puede observar que el comportamiento de todos los algoritmos, exceptuando el *G*, fue similar, con unos valores de $\overline{RPD_2}$ alrededor del 1 %, e inferiores a la mitad del valor correspondiente al algoritmo *G*. En cuanto a los valores de $\overline{RPD_1}$ correspondientes a la fase constructiva de los algoritmos, se puede observar que *GR-9/2* es el que ofrece mejor comportamiento, seguido de muy cerca por *GR-9/4* y *G*; en este caso, el algoritmo constructivo que dio peores resultados es *M*. Por otra parte, los tiempos de *CPU* requeridos por cada ejemplar en cada iteración son similares para todos los procedimientos (*alrededor de 0.30s*). Dichos tiempos son muy inferiores a los empleados por el solver *CPLEX* y ligeramente inferiores a los del procedimiento *BDP* (*recogidos en* [20]) que son del orden de 81.7*s* y 3.3*s*, respectivamente.

Examinando los resultados recogidos en la tabla IV podemos concluir que el peor procedimiento para todos los bloques fue *G* (*en sintonía con los valores de* $\overline{RPD_2}$ *para los 225 ejemplares*); el resto de algoritmos se comporta de forma irregular en todos los bloques, aunque ofrecen mejores resultados que *G*. En cuanto a resultados por bloques, las mejores soluciones se concentraron en el bloque 1, mientras que las peores soluciones lo hicieron en el bloque 3.

Si observamos los resultados por estructuras (*tabla V*), la estructura, sobre la que se obtuvieron mejores resultados, fue la 4; siendo la estructura 1 la más difícil de resolver para todos los algoritmos. De nuevo, el algoritmo que presentó peores promedios en este caso fue *G*, comportándose el resto de algoritmos de una forma similar.

VII. Conclusiones

Se formula el problema *MMSP-W* con vínculos entre estaciones de trabajo dispuestas en serie y sin restricciones en los instantes de interrupción de las operaciones. El resultado de la formulación es un programa lineal entero mixto.

Se propone una formulación para algoritmos *GRASP* que permite su generalización. Nuestra extensión (*GRASP-x*) comprende, a través de la definición de tres parámetros adicionales (*impedancia y elasticidades, aditiva y potencial*), los procedimientos *Greedy* constructivos con tratamiento de empates y posterior mejora local y los algoritmos *GRASP* y *Multistart* tradicionales; también comprende la posibilidad de seleccionar los elementos candidatos de la lista *RCL* con una probabilidad de selección dependiente de la calidad aportada por éstos a la secuencia parcial en fase de construcción.

La propuesta *GRASP-x* se concreta en 7 algoritmos maestros que se aplican a la variante del *MMSP-W*, objeto de estudio. Bajo la capa de la fase constructiva de los algoritmos maestros, se determinan los valores de la función de aptitud correspondientes a la incorporación, a la secuencia parcial en curso, de cada uno de los tipos de producto candidatos. La determinación de dichos valores, a través del solver *Gurobi* (*v.4.5.0*), se realiza mediante un programa lineal asociado al problema *MMSP-W*, relajado en las condiciones binarias para las variables de secuenciación.

La experiencia computacional sobre 225 ejemplares del problema, presentes en la literatura, pone de manifiesto la competencia de *GRASP* (*en tiempos de computación*) frente a procedimientos exactos de resolución basados en la programación lineal entera mixta (*tales como los incorporados al solver CPLEX*) y un peor comportamiento ante el uso de la programación dinámica acotada.

Agradecimientos

Los autores agradecen la colaboración prestada por *Nissan Spanish Industrial Operations* (NSIO), la Cátedra Nissan UPC y al Gobierno Español por la financiación parcial de este trabajo a través del proyecto PROTHIUS-III: DPI2010-16759, incluyendo fondos FEDER.

Referencias

[1] N. Boysen, M. Fliedner y A. Scholl, *Sequencing mixed-model assembly lines: survey, classification and model critique*, European Journal of Operational Research, 192/2, 349-373, 2009.

[2] C. A. Yano y R. Rachamadugu, *Sequencing to minimize work overload in assembly lines with product options*, Management Science, 37/5, 572-586, 1991.

[3] J. Bautista, J. Pereira y B. Adenso-Díaz, *A GRASP approach for the extended car sequencing problem*, Journal of Scheduling, 11/1, 3-16, 2008.

[4] K. Okamura y H. Yamashina, *A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor*, International Journal of Production Research, 17/3, 233-247, 1979.

[5] Z. Xiaobo y K. Ohno, *Algorithms for sequencing mixed models on assembly line in a JIT production system*, Computers & Industrial Engineering, 31/1, 47-56, 1997.

[6] J. Bautista y J. Cano *Minimizing work overload in mixed-model assembly lines*, International Journal of Production Economics, 112/1, 177-191, 2008.

[7] A. Bolat, *Efficient methods for sequencing minimum job sets on mixed model assembly lines*, Naval Research Logistics, 44/5, 419-437, 1997.

[8] L.H. Tsai, *Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage*, Management Science, 41/3, 485-495, 1995.

[9] A. Bolat, *A mathematical model for sequencing mixed models with due dates*, International Journal of Production Research, 41/5, 897-918, 2003.

[10] J. Bautista y A. Cano, *Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules*, European Journal of Operational Research, 210/3, 495-513, 2011.

[11] C. A. Yano y A. Bolat, *Survey, development, and application of algorithms for sequencing paced assembly lines*, Journal of Manufacturing and Operations Management, 2/3, 172-198, 1989.

[12] A. Bolat y C.A. Yano, *Scheduling algorithms to minimize utility work at a single station on paced assembly line*, Production Planning and Control, 3/4, 393-405, 1992.

[13] A. Scholl, R. Klein y W. Domschke, *Pattern based vocabulary building for effectively sequencing mixed-model assembly lines*, Journal of Heuristics, 4/4, 359-381, 1998.

[14] J. Cano, R. Ríos y J. Bautista, *A scatter search based hyper-heuristic for sequencing a mixed-model assembly line*, Journal of Heuristics, 6/6, 749-770, 2010.

[15] E. Erel, Y. Gocgun y I. Sabuncuoglu, *Mixed-model assembly line sequencing using beam search*, International Journal of Production Research, 45/22, 5265-5284, 2007.

[16] T.A. Feo y M.G.C. Resende *Greedy randomized adaptive search procedures*, J. of Global Optimization, vol. 6, 109-133, 1995.

[17] M.G.C. Resende y C.C. Ribeiro, *Greedy randomized adaptive search procedures*, Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, 219-249, 2003.

[18] P. Festa y M.G.C. Resende, *An annotated bibliography of GRASP–Part I: Algorithms*, International Transactions in Operational Research, vol. 16, 1-24, 2009.

[19] P. Festa y M.G.C. Resende, *An annotated bibliography of GRASP–Part II: Applications*, International Transactions in Operational Research, vol. 16, pp. 131-172, 2009.

[20] J. Bautista, A. Cano y R. Alfaro, *A bounded dynamic programming algorithm for the MMSP-W considering workstation dependencies and unrestricted interruption of the operations*, 11th International Conference on Intelligent Systems Design and Applications (ISDA), accepted, 2011.

[21] M. Laguna y R. Martí, *GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization*, INFORMS Journal on Computing, 11/1, 44-52, 1999.

[22] M. Prais y C.C. Ribeiro, *Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment* , INFORMS Journal on Computing, 12, 164-176, 2000.