GRASP para secuenciar modelos mixtos en una línea con sobrecarga, tiempo inerte y regularidad en la producción

Joaquín Bautista [1], Rocío Alfaro-Pozo [1], Cristina Batalla-García [1]

[1]Research Group OPE-PROTHIUS. Universitat Politècnica de Catalunya.
Avda. Diagonal, 647, 7th floor, 08028 Barcelona, Spain.
{joaquin.bautista; rocio.alfaro; cristina.batalla}@upc.edu

Resumen. Se presenta un algoritmo GRASP para resolver un problema de secuenciación de productos en una línea de montaje de modelos mixtos. El objetivo del problema es obtener una secuencia de fabricación de productos con máximo trabajo total completado y cumpliendo la propiedad de regularidad en la producción. El algoritmo GRASP implementado se compara con otros procedimientos de resolución, empleando para ello las instancias de un caso de estudio asociado a la planta de fabricación de motores de Nissan en Barcelona.

Keywords: GRASP; Secuenciación; Líneas de productos mixtos; Preservación del mix de producción.

1 Preliminares

Diremos que una línea de montaje es de modelos mixtos cuando esté capacitada para fabricar variantes distintas de un mismo producto (vg.- motores para 4x4 y furgonetas con distintas versiones), sin cambios físicos en las estaciones de trabajo y sin tiempos de preparación significativos al fabricar unidades consecutivas distintas. En este tipo de líneas, aparecen dos categorías de problemas: (1) problemas de equilibrado de líneas [1], y (2) problemas de secuenciación de productos [2]; siendo tradicional que éstos se resuelvan secuencialmente. El primero consiste en asignar, de la forma más eficiente posible y respetando una serie de condiciones, un conjunto de tareas (relativas al ensamblado de un producto) a un conjunto de estaciones de trabajo dispuestas en serie. El segundo problema, una vez resuelto el primero y dados un plan de demanda y un tiempo para ejecutarlo, consiste en establecer el orden de fabricación de los productos en función de uno o más criterios.

Los objetivos, no necesariamente excluyentes, que se tienen en cuenta a la hora de secuenciar las unidades, suelen responder a una o varias preocupaciones de carácter productivo [3]. Entre dichos objetivos están: (o.1) maximizar el tiempo productivo, completando el máximo número de unidades y reduciendo las esperas innecesarias [4]; (o.2) maximizar el grado de satisfacción de un conjunto de restricciones relacionadas con componentes especiales de los productos [5]; y (o.3) mantener constantes la tasa de fabricación de productos y la tasa de consumo de componentes con el propósito de reducir al mínimo los niveles máximos de stock de estos últimos [6].

Entre los problemas de secuenciación que atienden al objetivo (o.1), se encuentra el MMSP-W (*Mixed Model Sequencing Problem with Workload Minimisation*). El problema consiste en establecer una secuencia de T unidades de un conjunto de tipos de producto I que se fabrican en una línea de montaje compuesta por un conjunto K de estaciones en serie. Para completar una operación sobre una unidad de producto tipo i ($i = 1,..,|I|$) en la estación k ($k = 1,..,|K|$), se necesita un tiempo de proceso $p_{i,k}$ por parte de cada procesador asignado a la estación (operarios, robots, etc). Dicho tiempo se mide a una actividad normal (factor de actividad: $\alpha^N = 1$), la misma con la que se mide el tiempo estándar, denominado tiempo de ciclo c, que se concede a cada procesador para realizar una operación,. Los tiempos de proceso $p_{i,k}$ suelen ser distintos del ciclo c, por lo que es frecuente que los procesadores se encuentren ante dos situaciones: (1) que tengan que esperar un tiempo, llamado tiempo inerte, entre los instantes de finalización y de inicio de dos operaciones consecutivas, y (2) que no dispongan de tiempo suficiente para completar una operación. Para aliviar esta segunda situación, dada la estación k ($k = 1,..,|K|$), se concede, eventualmente y a cada procesador, un tiempo superior al del ciclo c, al que llamamos ventana temporal l_k ($l_k > c$), para completar una unidad de producto. Lógicamente, esta concesión temporal va en detrimento del tiempo disponible para trabajar sobre dicha unidad en la estación siguiente $k + 1$, y sobre la unidad siguiente en la propia estación k. Cuando la ventana temporal no es suficiente para completar todo el trabajo requerido sobre las unidades, diremos que hay sobrecarga de trabajo o trabajo perdido.

En sus primeras formulaciones, el MMSP-W tiene por objetivos maximizar el trabajo completado [4] o minimizar la sobrecarga de trabajo [7]; posteriormente, se demuestra que ambos objetivos son equivalentes y se establece la vinculación, carente en los trabajos pioneros, entre los instantes de finalización y de inicio de las operaciones que fluyen en la cadena de estaciones que componen la línea [3].

En cuanto a las propiedades que ofrece el objetivo (o.3) a las secuencias de fabricación, podemos afirmar que son del todo deseables en los entornos Justo a Tiempo (*Just in Time*) [8] o Fabricación Ajustada (*Lean Manufacturing*) [9], propios del sector que nos ocupa: el sector de automoción. Por ello, hemos incorporado al MMSP-W genuino un conjunto de restricciones que propician la regularidad en producción, esto es: preservar el mix de producción en la secuencia de fabricación [6,10].

En el presente trabajo nos centramos en una variante del MMSP-W que combina el objetivo (o.1) con el (o.3), quedando representado el primero por funciones objetivo (sobrecarga W y tiempo inerte U) y el segundo por restricciones sobre el mix de producción (*pmr*). Llamaremos MMSP-W/U/pmr a esta versión del problema.

Para resolver un caso de la planta de motores de Nissan BCN, hemos implementado un algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*), cuyos resultados se comparan con los obtenidos en trabajos previos con Programación Dinámica Acotada (BDP) [3] y con Programación Lineal Entera Mixta (MILP) [10].

El texto que sigue se estructura así: formulamos el MMSP-W/U/pmr sin libre interrupción de las operaciones en la sección 2; la sección 3 la dedicamos al algoritmo GRASP implementado; en la sección 4 describimos el caso de estudio y mostramos los resultados ofrecidos por los procedimientos utilizados; finalmente, recogemos las conclusiones del trabajo en la sección 5.

2 MMSP-W/U con preservación del mix de producción

El MMSP-W/U/pmr básico lo enunciamos como sigue.

Dados:

- El conjunto de tipos de producto ($I: i = 1,..,|I|$) y el conjunto de estaciones de trabajo ($K: k = 1,..,|K|$).
- El tiempo de ciclo c y las ventanas temporales l_k ($k \in K$), que se concede a cada procesador para trabajar sobre una unidad de producto en su estación, y el número de procesadores asignados a cada estación b_k ($k \in K$).
- Los tiempos de proceso $p_{i,k}$ ($i \in I \wedge k \in K$) de las operaciones, medidos a actividad normal ($\alpha^N = 1$), y los vectores demanda $\vec{d} = (d_1, \dots, d_{|I|})$ y mix de producción $\vec{\lambda} = (\lambda_1, \dots, \lambda_{|I|})$ - donde d_i es el número de unidades de producto tipo $i \in I$ contenidas en el plan de producción-demanda, y λ_i es la proporción del modelo $i \in I$ en el plan, cumpliéndose: $\vec{\lambda} = \vec{d}/D$ y $T \equiv D = \sum_{\forall i} d_i$ -.

El problema consiste en hallar una secuencia de T productos $\pi(T) = (\pi_1, \dots, \pi_T)$ con mínima sobrecarga W, mínimo tiempo inerte U, y satisfaciendo el plan de demanda representado por el vector $\vec{d}$ y las restricciones de preservación del mix de producción. Por consiguiente, las condiciones del problema son:

$$W(\pi(T)) = \sum_{t=1}^{T} \sum_{k=1}^{|K|} b_k w_{k,t}(\pi_t) \quad (1)$$

$$U(\pi(T)) = \sum_{t=1}^{T} \sum_{k=1}^{|K|} b_k u_{k,t}(\pi_t) \quad (2)$$

$$w_{k,t}(\pi_t) = max(0, s_{k,t}(\pi_t) + p_{\pi_t,k} - (k + t - 2)c - l_k) \quad \forall k \in K \; \forall t = 1,..,T \quad (3)$$

$$u_{k,t}(\pi_t) = s_{k,t}(\pi_t) - e_{k,t-1}(\pi_{t-1}) \quad \forall k \in K \; \forall t = 1,..,T \quad (4)$$

$$s_{k,t}(\pi_t) = max(e_{k,t-1}(\pi_{t-1}), e_{k-1,t}(\pi_t), (k + t - 2)c) \quad \forall k \in K \; \forall t = 1,..,T \quad (5)$$

$$e_{k,t-1}(\pi_{t-1}) = s_{k,t-1}(\pi_{t-1}) + p_{\pi_{t-1},k} - w_{k,t-1}(\pi_{t-1}) \quad \forall k \in K \; \forall t = 2,..,T \quad (6)$$

$$e_{k-1,t}(\pi_t) = s_{k-1,t}(\pi_t) + p_{\pi_t,k-1} - w_{k-1,t}(\pi_t) \quad \forall k \in K - \{1\} \; \forall t = 1,..,T \quad (7)$$

$$e_{k,t}(\pi_t) = min(s_{k,t}(\pi_t) + p_{\pi_t,k}, (k + t - 2)c + l_k) \quad \forall k \in K \; \forall t = 1,..,T \quad (8)$$

$$\lfloor \lambda_i t \rfloor \leq X_{i,t} \leq \lceil \lambda_i t \rceil, \; X_{i,T} = d_i \quad \forall i \in I \; \forall t = 1,..,T \quad (9)$$

Las expresiones (1) y (2) determinan, respectivamente, la sobrecarga W y el tiempo inerte U, generados por la secuencia $\pi(T)$. Las igualdades (3) permiten determinar las sobrecargas parciales de cada estación k y de cada periodo t, sin libre interrupción de las operaciones, provocando la interrupción forzosa cuando se alcanza el límite superior de la ventana temporal: $(k + t - 2)c + l_k$. Por su parte, las igualdades (4) definen el tiempo inerte parcial que se genera en cada estación y en cada periodo de fabricación en función de $\pi(T)$. Las igualdades (5), por un lado, y (6), (7) y (8), por otro, determinan los instantes mínimos de inicio, $s_{k,t}$, y de finalización, $e_{k,t}$, de las $|K| \times D$ operaciones. Finalmente, las condiciones (9) imponen la satisfacción del plan de demanda representado por $\vec{d}$ y obligan a preservar el mix de producción en todos

los periodos; para formular estas últimas restricciones, empleamos las variables $X_{i,t}$ que simbolizan el número de unidades de producto de tipo $i \in I$ contenidas en las secuencias parciales: $\pi(t) \equiv (\pi_1, \ldots, \pi_t) \subseteq \pi(T)$ $(\forall\, t = 1, .., T)$.

3 Algoritmo GRASP

GRASP es una metaheurística multiarranque [11,12] provista de dos fases: (1) Fase constructiva de una solución inicial, para la que es usual emplear un procedimiento Greedy no determinista; y (2) Fase de mejora de la solución, cuyo propósito es alcanzar un óptimo local en un vecindario concreto. La aplicación consecutiva de ambas fases recibe el nombre de iteración. Tras ejecutar un número prefijado de iteraciones, GRASP ofrece una solución final que es la mejor de todas las iteraciones.

3.1 Fase constructiva: Procedimiento Greedy

Se construye una secuencia de productos $\pi(T) = (\pi_1, \ldots, \pi_T)$, asignando de manera progresiva en cada etapa t $(t = 1, \ldots, T)$ un tipo de producto presente en la lista de candidatos a ocupar la posición t de la secuencia – sea $CL(t)$ dicha lista. Por tanto, llegados a la etapa t se añade a la secuencia $\pi(t-1) = (\pi_1, \ldots, \pi_{t-1})$, ya consolidada, un tipo de producto $i \in CL(t)$. Inicialmente, para que un producto tipo $i \in I$ forme parte de la lista $CL(t)$ debe cumplir al menos dos condiciones:

(c.1) El número de unidades $X_{i,t-1}$ de tipo $i \in I$ contenidas en la secuencia $\pi(t-1)$ debe ser menor que su demanda dentro del plan de producción: $X_{i,t-1} < d_i$.

(c.2) La producción del producto i hasta el periodo t $(X_{i,t} = X_{i,t-1} + 1)$ debe satisfacer las restricciones de preservación del mix de producción: $\lfloor \lambda_i t \rfloor \leq X_{i,t} \leq \lceil \lambda_i t \rceil$.

En ocasiones, la lista $CL(t)$ puede quedar vacía al imponer a la vez las condiciones (c.1) y (c.2). En tal caso, y manteniendo la condición (c.1), relajaremos por pasos la condición (c.2): primero imponiendo sólo la limitación superior de preservación, segundo imponiendo sólo la limitación inferior y tercero suprimiendo ambos límites.

Posteriormente, se evalúan los tipos de producto candidatos $i \in CL(t)$ con el fin de establecer un orden prioritario de selección en cada etapa. En este trabajo, emplearemos dos índices de prioridad jerarquizados para establecer dicha ordenación.

El primer índice, asociado al primer objetivo o meta a alcanzar, se refiere a la sobrecarga de trabajo generada por la secuencia $\pi_i(t) \equiv \pi(t-1) \cup \{i\}$, que resulta al añadir el producto $i \in CL(t)$ a la secuencia consolidada en la etapa $t-1$. Esto es:

$$f_i^{(t)} = W(\pi_i(t)) = W(\pi(t-1)) + \sum_{k=1}^{|K|} b_k w_{k,t}(i)$$
$$(\forall i \in CL(t) \wedge \forall t = 1, .., T) \qquad (10)$$

donde $w_{k,t}(i)$ simboliza la sobrecarga parcial de trabajo que soporta un procesador de la estación $k \in K$ cuando la t-ésima unidad de producto que se fabrica es del tipo i. Esta sobrecarga, sin libre interrupción de operaciones, se calcula así:

$$w_{k,t}(i) = max\big(0, s_{k,t}(i) + p_{i,k} - (k + t - 2)c - l_k\big) \qquad (11)$$

siendo $s_{k,t}(i)$ el instante de inicio de la operación en la estación k de una unidad de producto i que ocupa la t-ésima posición en la secuencia. Dicho instante depende del inicio del t-ésimo ciclo de fabricación en la estación k y de los instantes en que se dan por concluidas las operaciones en curso en las estaciones k y $k-1$. Estos instantes, sin libre interrupción de operaciones y adoptando el convenio $s_{1,1}(i) = 0\ \forall i \in I$, se determinan mediante las expresiones recurrentes siguientes:

$$s_{k,t}(i) = max\big(e_{k,t-1}(\pi_{t-1}), e_{k-1,t}(i), (k+t-2)c\big) \tag{12}$$

$$e_{k,t-1}(\pi_{t-1}) = s_{k,t-1}(\pi_{t-1}) + p_{\pi_{t-1},k} - w_{k,t-1}(\pi_{t-1}) \tag{13}$$

$$e_{k-1,t}(i) = s_{k-1,t}(i) + p_{i,k-1} - w_{k-1,t}(i) \tag{14}$$

El segundo índice (subordinado al primero) atiende a conseguir secuencias de productos que generen el mínimo tiempo inerte en las estaciones de trabajo. Esto es:

$$g_i^{(t)} = U\big(\pi_i(t)\big) = U\big(\pi(t-1)\big) + \textstyle\sum_{k=1}^{|K|} b_k u_{k,t}(i)$$
$$(\forall i \in CL(t) \wedge \forall t = 1,..,T) \tag{15}$$

siendo $u_{k,t}(i)$ el tiempo inerte del que dispone un procesador de la estación k, entre el instante de finalización $e_{k,t-1}(\pi_{t-1})$ de la unidad de la posición $t-1$ de la secuencia, y el instante de inicio $s_{k,t}(i)$ de la t-ésima unidad, siendo ésta del tipo i. Esto es:

$$u_{k,t}(i) = s_{k,t}(i) - e_{k,t-1}(\pi_{t-1}) \tag{16}$$

El par de índices $\big(f_i^{(t)}, g_i^{(t)}\big)$ permite ordenar, siempre en sentido no decreciente, los elementos de la lista $CL(t)$ convirtiéndola en la lista ordenada $\overline{CL}(t)$. La ordenación atiende a la aplicación en jerarquía de dos criterios, por tanto, el criterio de tiempo inerte sólo será efectivo cuando se produzca empate en sobrecarga.

Tras dicha ordenación, la lista $\overline{CL}(t)$ se reduce a través del factor de admisión Λ (tanto por ciento de productos-tipo que pasan a sorteo entre los mejores candidatos), obteniéndose la lista restringida $\overline{RCL}(t, \Lambda)$, idéntica a $\overline{CL}(t)$ cuando $\Lambda = 100\%$.

Detallamos en la Tabla 1 la fase constructiva de GRASP para construir una secuencia de productos $\pi(T)$ con mínima sobrecarga con mínimo tiempo inerte y con máxima preservación del mix de producción respetando la jerarquía de criterios.

La secuencia de tareas $\pi(T)$, resultante de la fase constructiva de GRASP, puede violar, en una o varias posiciones, la condición de preservación del mix de producción cuando, en la etapa t del Paso-1 de la ejecución del algoritmo, la primera lista tentativa $CL(t)$ queda vacía y se abre el paso a tipos de producto con demanda pendiente que no satisfacen la condición (c.2). En tal caso, acto seguido se resuelve un problema de máxima satisfacción de restricciones ($\lfloor \lambda_i t \rfloor \leq X_{i,t} \leq \lceil \lambda_i t \rceil\ , \forall i \forall t$) empleando un procedimiento de intercambio de tipos de productos que transforma la secuencia original $\pi(T)$ en la secuencia $\hat{\pi}(T)$ que sí satisface las restricciones de preservación del mix de producción (9) en todas las posiciones y para todos los tipos de producto.

Tabla 1. Esquema de la fase constructiva de la secuencia con orientación hacia mínima sobrecarga, mínimo tiempo inerte y preservación del mix de producción.

0. *Inicialización:*

   Leer: $\Lambda, I, K, D, c, \left(d_i, p_{i,k}, l_k\right)\ \forall i \in I\ \forall k \in K$
   Hacer: $T = D, t = 0, \pi(t) = \{\emptyset\}, (X_i = 0, \lambda_i = d_i/D)\ \forall i \in I$

1. *Creación del conjunto de tipos de producto candidatos:*

   $t \leftarrow t + 1$
   Sea $CL(t) = \{i \in I: (X_i < d_i) \wedge (\lfloor \lambda_i t \rfloor \leq X_i + 1 \leq \lceil \lambda_i t \rceil)\}$
   - Si $CL(t) = \{\emptyset\} \Rightarrow CL(t) = \{i \in I: (X_i < d_i) \wedge (X_i + 1 \leq \lceil \lambda_i t \rceil)\}$
   - Si $CL(t) = \{\emptyset\} \Rightarrow CL(t) = \{i \in I: (X_i < d_i) \wedge (X_i + 1 \geq \lfloor \lambda_i t \rfloor)\}$
   - Si $CL(t) = \{\emptyset\} \Rightarrow CL(t) = \{i \in I: X_i < d_i\}$

2. *Valoración de tipos de producto candidatos:*

   $\forall i \in CL(t)$, utilizando: (10) - (16), determinar:

   $f_i^{(t)} = W\left(\pi_i(t)\right) = W\left(\pi(t-1)\right) + \sum_{k=1}^{|K|} b_k w_{k,t}(i)$

   $g_i^{(t)} = U\left(\pi_i(t)\right) = U\left(\pi(t-1)\right) + \sum_{k=1}^{|K|} b_k u_{k,t}(i)$

3. *Ordenación de tipos de producto candidatos:*

   Sea: $\overline{CL}(t) = \left(i_1, \dots, i_{|RCL(t)|}\right)$ la lista ordenada de productos candidatos.

   - Se cumple: $pos\left(i, \overline{CL}(t)\right) < pos\left(i', \overline{CL}(t)\right)\ \forall \{i, i'\} \subseteq \overline{CL}(t)$, si se satisface la condición:

   $$\left[\left(f_i^{(t)} < f_{i'}^{(t)}\right)\right] \vee \left[\left(f_i^{(t)} = f_{i'}^{(t)}\right) \wedge \left(g_i^{(t)} < g_{i'}^{(t)}\right)\right]$$

4. *Selección del tipo de producto en la lista restringida* $\overline{RCL}(t, \Lambda) \subseteq \overline{CL}(t)$:

   - Sea: $pos^* = -int(-\Lambda \cdot |\overline{CL}(t)| \cdot RND)$ la posición seleccionada. Entonces, seleccionar el tipo de producto i^* que ocupa dicha posición:

   $$i^* = i_{pos^*} \in \overline{RCL}(t, \Lambda) = \left(i_1, \dots, i_{|\overline{RCL}(t,\Lambda)|}\right)\ con\ \overline{RCL}(t, \Lambda) \subseteq \overline{CL}(t)$$

5. *Actualización:*

   $X_{i^*} \leftarrow X_{i^*} + 1\ ;\ \ \pi(t) \equiv \pi(t-1) \cup \{i^*\}$

6. *Test de finalización:*

   Si $t < T$ Ir a Paso 1
   Si no, Finalizar

3.2 Fase de mejora local

Atendiendo a los dos objetivos en jerarquía (mínima sobrecarga de trabajo en las estaciones y mínimo tiempo inerte de los operadores), se parte de la secuencia $\hat{\pi}(T)$ y comienza una fase de mejora local en la que se ejecutan consecutiva y repetitivamente 4 algoritmos de descenso, sobre 4 vecindarios, hasta que ninguno de ellos mejora la mejor solución conseguida durante la iteración. Entre dos secuencias que satisfacen las restricciones de regularidad en producción, para todo producto y posición, diremos

que una es preferible a otra cuando su sobrecarga total tenga menor valor, y, en caso de empate, cuando presente menor tiempo inerte. Los algoritmos de descenso son:

(i) *Intercambio en avance*: Para toda posición t de la secuencia en curso $\hat{\pi}(T)$, se detecta el tipo de producto que ocupa dicha posición y se busca la siguiente posición más próxima $t'(t' > t)$ ocupada por un producto del mismo tipo ($\hat{\pi}_t = \hat{\pi}_{t'}$); si no existe, se hace $t' = T + 1$. Después, se realiza el intercambio tentativo entre $\hat{\pi}_t$ y los elementos contenidos en el rango $[t + 1, t' - 1]$ de la secuencia. Se consolida el primer intercambio que reduce la sobrecarga total $W(\hat{\pi}(T))$ o, *ex aequo*, el que reduce el tiempo inerte total $U(\hat{\pi}(T))$, siempre que con el intercambio se respeten las restricciones de preservación del mix de producción para todo producto y posición del rango $[t + 1, t' - 1]$. Esto se repite mientras haya mejora.

(ii) *Intercambio en retroceso*: Mientras haya mejora, para toda posición t de la secuencia $\hat{\pi}(T)$, se detecta el tipo de producto que ocupa la posición t y se busca la anterior posición más cercana $t'(t' < t)$ ocupada por el mismo tipo de producto ($\hat{\pi}_t = \hat{\pi}_{t'}$); si no existe, se hace $t' = 0$. Tras ello, se realiza el intercambio tentativo entre $\hat{\pi}_t$ y los elementos del rango $[t' + 1, t - 1]$ de la secuencia. Sin violar en el rango $[t' + 1, t - 1]$ las restricciones de preservación del mix de producción, se consolida el primer intercambio que reduce la sobrecarga total $W(\hat{\pi}(T))$ o el que reduce el tiempo inerte total $U(\hat{\pi}(T))$, en caso de empate en sobrecarga.

(iii) *Inserción en avance*: Para toda posición t de la secuencia $\hat{\pi}(T)$ en curso, se detecta el tipo de producto que ocupa dicha posición y se busca la posición más próxima posterior $t'(t' > t)$ con un producto del mismo tipo ($\hat{\pi}_t = \hat{\pi}_{t'}$); si no existe, se hace $t' = T + 1$. Seguidamente, se realiza la inserción tentativa del producto $\hat{\pi}_t$ en el rango de posiciones $[t + 1, t' - 1]$ de la secuencia. Se consolida la primera inserción que reduce la sobrecarga total $W(\hat{\pi}(T))$ o, en caso de empate, la que reduce el tiempo inerte total $U(\hat{\pi}(T))$, siempre que con la inserción de $\hat{\pi}_t$ se respeten las restricciones de preservación del mix de producción para todo producto y posición del rango $[t + 1, t' - 1]$. Esto se repite mientras haya mejora.

(iv) *Inserción en retroceso*: Para toda posición t de la secuencia $\hat{\pi}(T)$ en curso, se detecta el tipo de producto que ocupa la posición t y se busca la posición más próxima anterior $t'(t' < t)$ con un producto del mismo tipo ($\hat{\pi}_t = \hat{\pi}_{t'}$); si no existe, se hace $t' = 0$. Se inserta tentativamente el producto $\hat{\pi}_t$ en el rango de posiciones $[t' + 1, t - 1]$ de la secuencia. Se consolida la primera inserción que reduce la sobrecarga total $W(\hat{\pi}(T))$ o (*ex aequo*) la que reduce el tiempo inerte total $U(\hat{\pi}(T))$. En toda inserción consolidada de $\hat{\pi}_t$ se respetan las restricciones de preservación del mix de producción para todo producto y toda posición del rango $[t + 1, t' - 1]$. El procedimiento se repite mientras haya mejora.

4 Experiencia computacional. Caso de Estudio

La experiencia computacional está orientada a analizar el comportamiento de GRASP frente a otros dos procedimientos en cuanto a calidad de soluciones y tiempos de CPU; éstos son: (1) BDP (*Bounded Dynamic Programming*) y (2) MILP (*Mixed Integer Linear Programming*). Para el análisis, tomamos un caso de estudio de la planta

de Nissan en Barcelona, basado en una línea de ensamblado de 9 tipos de motores agrupados en 3 familias (4x4, furgonetas y camiones) en la que trabajan 42 operarios cuando el tiempo de ciclo es del orden de 3 minutos.

Las características del caso que nos ocupa son:

- Número de estaciones de trabajo: $|K| \equiv m = 21$.
- Número de tipos de producto: $|I| = 9$ ($i = 1,..,9$).
- Tiempo de ciclo: $c = 175\ s.$, y ventana temporal: $l_k = 195\ s.\ (\forall k = 1,..,21)$.
- Número de procesadores homogéneos (con 2 operarios): $b_k = 1\ (\forall k = 1,..,21)$.
- Tiempos de proceso $p_{i,k}$ ($\forall i \in I, \forall k \in K$) según producto y estación, comprendidos entre $89\ s.$ y $185\ s.$ a actividad normal (ver [3]: Table 5).
- Número de planes de demanda: $|\mathrm{E}| = 23$ ($\varepsilon = 1,..,23$). Todos ellos con idéntica demanda diaria (ver [3]: Table 6, Block I, NISSAN-9ENG).
- Demanda diaria: $T \equiv D_\varepsilon = 270\ unidades\ (\forall \varepsilon = 1,..,23)$.

Las características de los procedimientos empleados son:

- BDP: Algoritmo BDP-MMSPW (ver [3]): (1) código compilado y ejecutado en iMac (Intel Core 2 Duo 2.33 GHz, 3 GB de RAM); (2) máximo número de transiciones desde cada vértice igual al número de tipos de producto $|I| = 9$; (3) anchos de ventana $H = (1,10,50,100,250,500,750,1000)$ para los 23 planes de demanda (184 ejecuciones del algoritmo); (4) solución inicial Z_0 para H_n igual a la mejor solución obtenida con H_{n-1}, excepto para $H_1 = 1$, donde $Z_0 \to \infty$; (5) tiempo de CPU medio empleado para cada plan de demanda igual a 6416 *s.*; y (6) sin forzar preservación del mix de producción y sin libre interrupción de las operaciones.
- MILP: Modelos $3 \cup 4\_pmr$ y $4 \cup 3\_pmr$ (ver [10]): (1) implementación para solver Gurobi v4.5.0 y ejecución en iMac (Intel Core i7 2.93 GHz, 8 GB de RAM); (2) tiempo de CPU máximo concedido a cada modelo por plan de demanda igual 7200 *s.* (46 ejecuciones del solver); y (3) con restricciones de preservación del mix de producción y con libre interrupción de las operaciones.
- GRASP: (1) código compilado y ejecutado en iMac (Intel Core i7 2.93 Ghz, 8 GB de RAM); (2) máximo 10 iteraciones por plan de demanda; (3) factor de admisión $\Lambda = (25\%, 50\%, 100\%)$ (690 soluciones en 69 ejecuciones); (4) tiempo de CPU medio empleado por plan de demanda igual a 425 *s.*; y (5) con restricciones de preservación del mix de producción y sin libre interrupción de las operaciones.

La Tabla 2 recoge los mejores resultados conseguidos por BDP (ver Table 7 en [3]) , MILP (ver Table 3 en [8]) y GRASP, para la sobrecarga W en los 23 planes de demanda $\varepsilon \in \mathrm{E}$. En ella se muestran también el algoritmo *Ganador* en cada plan de demanda y las ganancias unitarias de GRASP frente a BDP (ΔGvB), GRASP frente a MILP (ΔGvM) y BDP frente a MILP (ΔBvM), que se determinan así:

$$\Delta \mathcal{P}v\mathcal{P}'(\varepsilon) = \frac{W_{\mathcal{P}'}(\varepsilon) - W_{\mathcal{P}}(\varepsilon)}{min\big(W_{\mathcal{P}'}(\varepsilon), W_{\mathcal{P}}(\varepsilon)\big)}$$

$$\forall \varepsilon \in \mathrm{E},\ \forall \mathcal{P} \in \{GRASP, BDP\}, \forall \mathcal{P}' \in \{BDP, MILP\} \qquad (17)$$

Table 2. Para cada plan $\varepsilon \in$ E, Sobrecarga W según procedimiento ($W_{BDP}, W_{MILP,}W_{GRASP}$), Ganancia unitaria entre pares de procedimientos ($\Delta GvB, \Delta GvM, \Delta BvM$) y Algoritmo *Ganador*.

$\varepsilon \in$ E	W_{BDP}	W_{MILP}	W_{GRASP}	ΔGvB	ΔGvM	ΔBvM	$Ganador$
1	166	186	142	0.17	0.31	0.12	GRASP
2	464	383	404	0.15	-0.05	-0.21	MILP
3	432	423	436	-0.01	-0.03	-0.02	MILP
4	440	307	535	-0.22	-0.74	-0.43	MILP
5	897	661	868	0.03	-0.31	-0.36	MILP
6	663	478	748	-0.13	-0.56	-0.39	MILP
7	823	731	790	0.04	-0.08	-0.13	MILP
8	129	160	96	0.34	0.67	0.24	GRASP
9	1149	751	1235	-0.07	-0.64	-0.53	MILP
10	1249	1208	1246	0.00	-0.03	-0.03	MILP
11	50	122	124	-1.48	-0.02	1.44	BDP
12	369	287	284	0.30	0.01	-0.29	GRASP
13	379	336	399	-0.05	-0.19	-0.13	MILP
14	578	423	543	0.06	-0.28	-0.37	MILP
15	553	442	461	0.20	-0.04	-0.25	MILP
16	223	251	255	-0.14	-0.02	0.13	BDP
17	640	488	556	0.15	-0.14	-0.31	MILP
18	962	619	1067	-0.11	-0.72	-0.55	MILP
19	980	945	971	0.01	-0.03	-0.04	MILP
20	104	150	234	-1.25	-0.56	0.44	BDP
21	854	561	943	-0.10	-0.68	-0.52	MILP
22	1104	984	1084	0.02	-0.10	-0.12	MILP
23	107	121	107	0.00	0.13	0.13	BDP/GRASP

A partir del análisis de la Tabla 2 podemos afirmar:

- El procedimiento Ganador es MILP con 16 mejores soluciones sobre 23. En este aspecto, GRASP es el segundo mejor procedimiento, consiguiendo la mejor solución en 3 ocasiones (planes 1, 8 y 12), mientras que BDP queda en última posición con 2 mejores soluciones (planes 11 y 20). BDP y GRASP empatan en el plan 23.
- GRASP vence a BDP en 12 ocasiones de 23 y empata en una de ellas, no obstante, la ganancia unitaria media de GRASP sobre BDP es del 12%, cuando GRASP es el ganador, mientras que la de BDP sobre GRASP es del 36%, cuando vence BDP. La ganancia unitaria media global de BDP sobre GRASP es del orden del 9%.
- GRASP gana a MILP en 4 ocasiones (planes 1, 8, 12 y 23). La ganancia unitaria media global de MILP sobre GRASP es del orden del 18%, mientras que, cuando GRASP vence a MILP o viceversa, las ganancias giran ambas alrededor del 28%.
- BDP vence a MILP 6 veces de 23 (planes 1, 8, 11, 16, 20 y 23). Las ganancias unitarias medias parciales, cuando BDP vence a MILP y viceversa, son iguales al 42% y al 28%. MILP vence globalmente a BDP con una ganancia del 9%.
- BDP, MILP y GRASP necesitaron en promedio 6416 *s*., 14400 *s.* y 425 *s.*, respectivamente, para confirmar sus mejores soluciones de cada plan de demanda $\varepsilon \in E$.

5 Conclusiones

A pesar de la desventaja con la que juega GRASP - sin libre interrupción de operaciones frente a MILP y con preservación del mix de producción frente a BDP-, el algoritmo implementado se muestra competitivo venciendo a BDP y ocupando la segunda posición como *Ganador*. En cuanto a tiempos de CPU, GRASP es el procedimiento más rápido en promedio de los tres: 12 veces más rápido que BDP (teniendo en cuenta que Intel Core i7 es 1.26 veces más rápido que Intel Core 2 Duo, empleando este último un solo procesador) y casi 34 veces más rápido que MILP.

Agradecimientos. Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad (Gobierno de España) con el proyecto FHI-SELM2 (TIN2014-57497-P).

6 Referencias

1. Battaïa, O., Dolgui, A.: A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics 142(2), pp. 259-277 (2013).
2. Boysen, N., Fliedner, M., Scholl, A.: Sequencing mixed-model assembly lines: Survey, classification and model critique. European Journal of Operational Research 192(2), pp. 349–373 (2009).
3. Bautista, J., Cano, A.: Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimization and interruption rules. European Journal of Operational Research 210, pp. 495–513 (2011).
4. Yano, C.A., Rachamadugu, R.: Sequencing to Minimize Work Overload in Assembly Lines with Product Options, Management Science 37(5), pp. 572-586 (1991).
5. Bautista, J., Pereira, J., Adenso-Díaz, B.: A GRASP approach for the extended car sequencing problema. Journal of Scheduling 11(1), pp. 3-16 (2008).
6. Bautista, J., Cano, A., Alfaro, R., Batalla, C.: Impact of the Production Mix Preservation on the ORV Problem. In Bielza, C. et al. (Eds.): CAEPIA 2013, LNAI 8109, pp. 250–259. Springer-Verlag Berlin Heidelberg (2013).
7. Scholl, A., Klein, R., Domschke, W.: Pattern based vocabulary building for effectively sequencing mixed-model assembly lines. Journal of Heuristics 4(4), pp. 359-381 (1998).
8. Omar, M., Sarker, R., Othman, W.A.M.: A just-in-time three-level integrated manufacturing system for linearly time-varying demand process. Applied Mathematical Modelling 37(3), pp. 1275–1281 (2013).
9. Fullerton, R.R., Kennedy, F.A., Widener, S.K.: Lean manufacturing and firm performance: The incremental contribution of lean management accounting practices. Journal of Operations Management 32(7-8), pp. 414-428 (2014).
10. Bautista, J., Cano, A., Alfaro, R.: Modeling and solving a variant of the mixed-model sequencing problem with work overload minimisation and regularity constraints. An application in Nissan's Barcelona Plant. Expert Systems with Applications 39, pp. 11001–11010 (2012).
11. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), pp. 109-133 (1995).
12. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures: Advances, hybridizations, and applications: In: Handbook of Metaheuristics, pp. 283-319. Springer US (2010).