Algoritmo híbrido GRASP-LP para secuenciar modelos mixtos en una línea de montaje con sobrecarga, libre interrupción y regularidad en la producción

Joaquín Bautista [1], Rocío Alfaro-Pozo [1]

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

Resumen. Se presenta un algoritmo híbrido GRASP asistido por Programación Lineal (LP) para resolver un problema de secuenciación de productos mixtos en una línea de montaje. El objeto del problema es obtener una secuencia de fabricación de modelos que genere la mínima sobrecarga con libre interrupción de las operaciones, preservando el mix de producción. El híbrido GRASP-LP implementado se compara con otros procedimientos, mediante instancias de un caso de estudio de la planta de fabricación de motores de Nissan en Barcelona.

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

1 Preliminares

Las líneas de montaje de modelos mixtos son aquellas que pueden fabricar versiones distintas de un producto, sin necesidad de cambios físicos en las estaciones de trabajo y con unos tiempos de preparación despreciables ante unidades consecutivas distintas. En este tipo de sistema productivo, distinguimos dos categorías de problemas:

p1. Equilibrado: Asignar eficientemente un conjunto de tareas de ensamblado de un producto a un conjunto de estaciones de trabajo dispuestas en serie, respetando una serie de restricciones temporales, espaciales y de riesgo [1].
p2. Secuenciación: Definir el orden de fabricación de los modelos en función de uno o más criterios, un plan de demanda y el tiempo disponible para ejecutarlo [2].

Los objetivos de los problemas de secuenciación responden a diversas preocupaciones de índole productivo [3]; entre ellos encontramos:

o1. Maximizar el número de unidades completadas en la línea, reduciendo el tiempo inerte de los operarios, las esperas innecesarias y las pérdidas de producción ocasionadas por excesos de carga de trabajo en las estaciones [4].
o2. Minimizar el número de restricciones violadas por la secuencia-solución, ante la presencia de unos condicionantes de carácter tecnológico y ergonómico que afectan a algunos componentes especiales del ensamblado [5].
o3. Maximizar la constancia de las tasas de fabricación de productos y de las tasas de consumo de componentes con el propósito de reducir al mínimo los niveles máximos de stock de estos últimos [6].

El presente trabajo (extensión de [7]) trata del MMSP-W (*Mixed Model Sequencing Problem with Workload Minimisation*) con libre interrupción de las operaciones (p2-o1). El problema consiste en establecer una biyección entre los elementos de un conjunto T de ciclos de fabricación (enumeramos: t $(t = 1, .., T)$) y los de un conjunto Ψ de productos (T elementos). Los elementos del conjunto Ψ se pueden agrupar en clases excluyentes ψ_i que cumplen: $\Psi = \bigcup_{i \in I} \psi_i$ y $\psi_i \cap \psi_{i'} = \emptyset, \forall \{i, i'\} \in I$; donde I es el conjunto de tipos de productos (enumeramos: i $(i = 1, .., n)$).

Cada tipo de producto $i \in I$ requiere para su compleción un tiempo de proceso $p_{i,k}$ $(i \in I, k \in K)$, medido a actividad normal (factor de actividad: $\alpha^N = 1$), en cada estación de trabajo k del conjunto de estaciones K que constituye la línea (enumeramos k $(k = 1, .., m)$). Obviamente, las diferencias entre las clases ψ_i (4x4, furgones, etc.) hace que los valores de los tiempos $p_{i,k}$ sean heterogéneos, mientras que el tiempo concedido a los procesadores (operarios y robots) para realizar una operación en cualquier estación y para cualquier unidad de producto es siempre el mismo; dicho tiempo recibe el nombre de tiempo de ciclo, lo notaremos c y se medirá también a actividad normal. Esta discrepancia entre el tiempo de ciclo y los tiempos de proceso provoca que los procesadores se puedan encontrar ante dos estados:

s1. Estado de espera con tiempo inerte: Parada entre el instante de finalización de una operación y el de inicio de la siguiente por no estar disponible el producto.
s2. Estado de bloqueo por sobrecarga: Los procesadores no disponen de tiempo suficiente para completar la operación.

Con el fin de aliviar el estado s2 en una estación genérica k $(k = 1, .., m)$, se puede conceder ocasionalmente a cada procesador un tiempo mayor al ciclo c, que llamamos ventana temporal l_k $(l_k > c)$, para completar su unidad de producto. Es evidente que tal concesión reducirá el tiempo disponible del operario para trabajar sobre su siguiente unidad y, por supuesto, mermará el tiempo disponible en la estación siguiente $(k + 1)$ para trabajar sobre la unidad retenida tras su liberación. Si el empleo de la ventana temporal no es suficiente para completar todo el trabajo requerido sobre la unidad, ésta se liberará sin completar hacia la siguiente estación, dando lugar a una ineficiencia que llamamos sobrecarga de trabajo o caída de producción.

La interrupción de una operación sobre una unidad incompleta se puede hacer de dos maneras:

i1. Interrupción forzada: Se produce cuando el operario alcanza el límite de la ventana temporal l_k en su estación sin que aquel haya completado el tiempo de proceso que le corresponde.
i2. Interrupción libre: Se produce cuando la unidad es liberada, sin que se haya completado la operación, antes de que el operario alcance el límite de la ventana temporal; obviamente, si se alcanza dicho límite, la operación se interrumpe también.

En cualquier caso, el propósito final del MMSP-W es obtener una secuencia de modelos que minimice la sobrecarga total de trabajo en la línea o, alternativa y equivalentemente, que maximice el trabajo total completado (Teorema 1 en [3]).

Por su parte, el objetivo (o3) ofrece a las secuencias de fabricación propiedades deseables en entornos *Just-In-Time* del sector Automoción [6]. Por ello, hemos incorporado al MMSP-W genuino restricciones que propician la regularidad en la producción, esto es: preservación del mix de producción en la secuencia de fabricación [8,9].

En este trabajo nos centramos en la variante del MMSP-W que combina los objetivos o1 y o3 y, a diferencia de [7], permitimos la interrupción libre de las operaciones i2. El objetivo o1 queda representado por funciones objetivo (sobrecarga W y tiempo inerte U), el objetivo o3 por restricciones sobre el mix de producción (*pmr*), y la condición i2 por inecuaciones. Llamaremos MMSP-W/pmr/free a esta versión del problema.

En este marco, hemos resuelto un caso inspirado en la planta de motores de Nissan BCN y, para ello, hemos implementado un algoritmo híbrido GRASP (*Greedy Randomized Adaptive Search Procedure*) asistido por LP (*Linear Programming*). Los resultados del híbrido GRASP-LP los comparamos con los obtenidos en los trabajos estado del arte del problema: Programación Dinámica Acotada asistida por programación lineal (BDP-2) [10] y con Programación Lineal Entera Mixta (MILP) [9].

El texto que sigue se estructura así: formulamos el MMSP-W/pmr/free en la sección 2; la sección 3 la dedicamos al algoritmo GRASP implementado y al programa lineal que asiste a GRASP para mejorar las soluciones que éste ofrece; en la sección 4 describimos el caso de estudio y mostramos los resultados ofrecidos por los tres procedimientos; finalmente, recogemos las conclusiones del trabajo en la sección 5.

2 MMSP-W con preservación del mix de producción y libre interrupción de operaciones

Dados:

a. Los conjuntos Tipo de producto ($I: i = 1,..,|I|$) y Estaciones ($K: k = 1,..,m$).
b. El tiempo de ciclo c, las ventanas temporales l_k ($k \in K$), el número de procesadores asignados a cada estación b_k ($k \in K$), y los tiempos de proceso $p_{i,k}$ ($i \in I \wedge k \in K$) de las operaciones, medidos a actividad normal.
c. El plan de demanda $\vec{d} = (d_1, \dots, d_n)$, donde d_i es el número de unidades del modelo $i \in I$; y el vector mix de producción $\vec{\lambda} = (\lambda_1, \dots, \lambda_n)$, donde 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 MMSP-W/pmr/free básico lo formulamos así:

$$W\big(\pi(T)\big) = \sum_{t=1}^{T} \sum_{k=1}^{m} b_k w_{k,t}(\pi_t) \tag{1}$$

$$U\big(\pi(T)\big) = \sum_{t=1}^{T} \sum_{k=1}^{m} b_k u_{k,t}(\pi_t) \tag{2}$$

$$0 \le w_{k,t}(\pi_t) \le max\Big(0, s_{k,t}(\pi_t) + p_{\pi_t,k} - e_{k,t}(\pi_t)\Big) \quad \forall k \in K\ \forall t \in \mathrm{T} \tag{3}$$

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

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

$$e_{k,t}(\pi_t) = s_{k,t}(\pi_t) + p_{\pi_t,k} - w_{k,t}(\pi_t)\ \forall k \in K\ \forall t \in \mathrm{T} \tag{6}$$

$$e_{k,t}(\pi_t) \le (k+t-2)c + l_k\ \forall k \in K\ \forall t \in \mathrm{T} \tag{7}$$

$$e_{k,0}(\pi_0) = e_{0,t}(\pi_t) = 0 \quad \forall k \in K\ \forall t \in \mathrm{T} \tag{8}$$

$$\lfloor \lambda_i t \rfloor \le X_{i,t} \le \lceil \lambda_i t \rceil,\ X_{i,T} = d_i\ \forall i \in I\ \forall t \in \mathrm{T} \tag{9}$$

El problema consiste en hallar una secuencia $\pi(T) = (\pi_1, \dots, \pi_T)$ de productos con las siguientes propiedades: (i) mínima sobrecarga W, (ii) mínimo tiempo inerte U, (iii) que cumpla el plan de demanda $\vec{d}$, (iv) que satisfaga las restricciones de preservación del mix de producción, y (v) con interrupción libre de las operaciones.

En la formulación, las definiciones (1) y (2) determinan, respectivamente, la sobrecarga W y el tiempo inerte U, generados por la secuencia $\pi(T)$. Las inecuaciones (3) acotan la sobrecarga parcial en toda estación k y todo ciclo t, permitiendo la libre interrupción de cualquier operación desde su instante de inicio hasta su instante de compleción o el que fija la ventana temporal: $(k + t - 2)c + l_k$. Las igualdades (4) definen el tiempo inerte parcial en cada estación y ciclo en función de $\pi(T)$. Las igualdades (5) determinan los instantes mínimos de inicio, $s_{k,t}$, mientras que (6), (7) y (8) determinan los instantes mínimos de finalización, $e_{k,t}$, de las $m \times D$ operaciones. Finalmente, las condiciones (9) obligan a preservar el mix de producción en todo ciclo y a cumplir el plan de demanda $\vec{d}$.

Para formular la preservación del mix, empleamos aquí las variables $X_{i,t}$, que simbolizan el número de unidades de tipo $i \in I$ contenidas en las secuencias parciales: $\pi(t) \equiv (\pi_1, \dots, \pi_t) \subseteq \pi(T)$ $(\forall\, t = 1, .., T)$.

3 Algoritmo híbrido GRASP-LP

GRASP es una metaheurística multiarranque [11] provista de dos fases: fase constructiva de una solución inicial, para la que se emplea un procedimiento greedy no determinista, y una fase de mejora de la solución, cuyo propósito es alcanzar un óptimo local en un vecindario concreto. Por su parte, la programación lineal es una técnica clásica de optimización que permite modelar y resolver, mediante algoritmos exactos, problemas con función objetivo lineal sujeta a restricciones lineales [12].

La naturaleza MMSP-W/pmr/free propicia la aplicación de ambas técnicas de resolución: GRASP se encargará del carácter combinatorio del problema, hallando la mejor secuencia $\pi(T) = (\pi_1, \dots, \pi_T)$ con interrupciones forzadas, y LP se encargará de la optimización con variables continuas, minimizando las funciones (1) y (2). Tras un número prefijado de iteraciones (fase constructiva más fase mejora), GRASP obtiene la mejor secuencia con interrupciones forzadas, la cual se emplea como dato de entrada de un programa lineal que minimiza la sobrecarga y el tiempo inerte globales.

3.1 Fase 1: Construcción de una secuencia

Se construye una secuencia de modelos $\pi(T) = (\pi_1, \dots, \pi_T)$, asignando progresivamente en cada etapa t $(t = 1, \dots, T)$ un producto de la lista de candidatos a ocupar la posición t de la secuencia – llamamos $CL(t)$ a dicha lista. Por tanto, cuando se alcanza la etapa t, se añade a la secuencia $\pi(t-1) = (\pi_1, \dots, \pi_{t-1})$, ya consolidada, un producto $i \in CL(t)$ (ver esquema en Tabla 1).

Para que el producto $i \in I$ entre en la lista $CL(t)$ debe cumplir 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 modelo i hasta el periodo t ($X_{i,t} = X_{i,t-1} + 1$) debe satisfacer las restricciones de preservación del mix de producción: $\lfloor \lambda_i t \rfloor \leq X_{i,t} \leq \lceil \lambda_i t \rceil$. Esto equivale a decir que la n-ésima unidad de tipo $i \in I$ (notaremos n_i) debe lanzarse en un ciclo t_{n_i} del intervalo $[t_{min}(n_i), t_{max}(n_i)]$, cumpliéndose por tanto: $t_{min}(n_i) \leq t_{n_i} \leq t_{max}(n_i)$, para todo $n_i = 1, .., d_i$.

Si la lista $CL(t)$ queda vacía al imponer a la vez las condiciones (c.1) y (c.2), se mantiene la condición (c.1) y se relaja la condición (c.2).

Posteriormente, se ordenan los productos candidatos $i \in CL(t)$ en la etapa t. Dicha ordenación responde a dos índices de prioridad jerarquizados.

El primer índice se refiere a la sobrecarga de trabajo generada por la secuencia $\pi_i(t) \equiv \pi(t-1) \cup \{i\}$, que resulta al añadir $i \in CL(t)$ a $\pi(t-1)$. Esto es:

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

En la expresión (10), $w_{k,t}(i)$ simboliza la sobrecarga parcial de trabajo soportada por un procesador de la estación $k \in K$ cuando la t-ésima unidad de producto es de tipo i. Esta sobrecarga, con interrupciones forzadas, se calcula según (11).

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

En la expresión (11), $s_{k,t}(i)$ es el instante de inicio de la operación en la estación k cuando un modelo tipo i 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 $k-1$. Adoptando la regla de interrupciones forzadas y haciendo $s_{1,1}(i) = 0 \; \forall i \in I$, los instantes de inicio y de finalización de las operaciones se determinan así:

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

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

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

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:

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

En la expresión (15), $u_{k,t}(i)$ es el tiempo inerte del que dispone un procesador de la estación k, entre los instantes $e_{k,t-1}(\pi_{t-1})$ y $s_{k,t}(i)$. Consecuentemente:

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

Con los índices $f_i^{(t)}$ y $g_i^{(t)}$ ordenamos crecientemente los elementos de la lista $CL(t)$ convirtiéndola en la lista $\overline{CL}(t)$. Como dicha ordenación se aplica jerárquicamente, el tiempo inerte $\left(g_i^{(t)}\right)$ sólo influye cuando hay un empate en la sobrecarga $\left(f_i^{(t)}\right)$.

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

El esquema de la fase constructiva GRASP implementada se recoge en la Tabla 1.

Tabla 1. Fase constructiva GRASP para una secuencia con interrupción forzada de operaciones, con sobrecarga y tiempo inerte mínimos y con 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: $T = D, t = 0, \pi(t) = \{\emptyset\}, (n_i = 0, \lambda_i = d_i/D)\ \forall i \in I$
1. *Creación del conjunto de tipos de producto candidatos:*
   $t \leftarrow t + 1$
   Sea $CL(t) = \left\{i \in I: (n_i < d_i) \wedge (t_{min}(n_i + 1) \leq t_{n_i+1} \leq t_{max}(n_i + 1))\right\}$
   - Si $CL(t) = \{\emptyset\} \Rightarrow CL(t) = \{i \in I: n_i < d_i\}$
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:*
   $n_{i^*} \leftarrow n_{i^*} + 1\ ;\ \pi(t) \equiv \pi(t-1) \cup \{i^*\}$
6. *Test de finalización:*
   Si $t < T$ Ir a Paso 1
   Si no, Finalizar

La secuencia de tareas $\pi(T)$, resultante de la fase constructiva GRASP, puede violar la condición de preservación del mix de producción cuando, en el Paso-1 de alguna etapa de ejecución del algoritmo, la lista $CL(t)$ queda vacía; entonces se abre paso a todos los productos con demanda pendiente. Si esto sucede, se activa un procedimiento de intercambio para resolver un problema de máxima satisfacción de restricciones $\left[t_{min}(n_i) \leq t_{n_i} \leq t_{max}(n_i), \forall n_i = 1, ., d_i: i \in I\right]$ transformando la secuencia original $\pi(T)$ en la secuencia $\hat{\pi}(T)$ que sí satisface las restricciones de preservación.

3.2 Fase 2: Mejora de la solución mediante búsqueda local

Similarmente a [7], se parte de la secuencia $\hat{\pi}(T)$, que satisface las condiciones (9), dando comienzo a la 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 preserven el mix de producción se prefiere la que tenga menor sobrecarga total $W\left(\pi(T)\right)$ y, en caso de empate, la que presente menor tiempo inerte $U\left(\pi(T)\right)$. Los

algoritmos de descenso se basan en el intercambio y en la inserción de productos (ver [7]) y están orientados a la exploración de ciclos de la secuencia tanto en sentido creciente como decreciente. Dichos procedimientos son: (i) intercambio en avance, (ii) intercambio en retroceso, (iii) Inserción en avance, y (iv) inserción en retroceso. El resultado de esta segunda fase es la secuencia $\pi^*(T) = (\pi_1^*, \dots, \pi_T^*)$.

3.3 Fase 3: Minimización de la sobrecarga mediante Programación Lineal

Tras la fase de mejora de GRASP, la secuencia $\pi^*(T) = (\pi_1^*, \dots, \pi_T^*)$, con menor sobrecarga, se toma como dato del programa lineal LP-W cuyo objetivo es reducir al mínimo la sobrecarga global permitiendo la libre interrupción de las operaciones.

Sea LP-W: $$\min W = \sum_{t=1}^{T} \sum_{k=1}^{|K|} b_k w_{k,t} \quad (17)$$

Sujeto a:

$$v_{k,t} + w_{k,t} = p_{\pi_t^*,k} \qquad \forall k = 1,.,m;\ \forall t = 1,.,T \quad (18)$$

$$s_{k,t} \geq (k + t - 2)c \qquad \forall k = 1,.,m;\ \forall t = 1,.,T \quad (19)$$

$$s_{k,t} \geq s_{k,t-1} + v_{k,t-1} \qquad \forall k = 1,.,m;\ \forall t = 2,.,T \quad (20)$$

$$s_{k,t} \geq s_{k-1,t} + v_{k-1,t} \qquad \forall k = 2,.,m;\ \forall t = 1,.,T \quad (21)$$

$$s_{k,t} + v_{k,t} \leq (k + t - 2)c + l_k \qquad \forall k = 1,.,m;\ \forall t = 1,.,T \quad (22)$$

$$v_{k,t}, w_{k,t} \geq 0 \qquad \forall k = 1,.,m;\ \forall t = 1,.,T \quad (23)$$

donde $s_{k,t}$, $v_{k,t}$ y $w_{k,t}$ son variables reales que representan el instante de inicio, el trabajo completado y la sobrecarga de trabajo de la t-ésima operación en la estación k, respectivamente.

Con la utilización de LP-W, el procedimiento híbrido GRASP-LP se encuentra en igualdad de condiciones ante la resolución del problema para competir con otros procedimientos de la literatura: MILP [9] y BDP-2 [10].

4 Experiencia computacional. Caso de Estudio

La experiencia computacional que proponemos está enfocada a analizar el comportamiento de GRASP-LP frente a otros dos procedimientos en cuanto a calidad de soluciones y tiempos de CPU; dichos procedimientos son: (1) BDP (*Bounded Dynamic Programming*) y (2) MILP (*Mixed Integer Linear Programming*). Igual que en [7], el análisis se realiza con un caso de estudio de la planta de Nissan en Barcelona: una línea de ensamblado de 9 tipos de motores agrupados en 3 familias (4x4, furgonetas y camiones) en la que trabajan 42 operarios con un tiempo de ciclo de 175 *s*.

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]: Tabla 5).
- Número de planes de demanda: $|\mathrm{E}| = 23$ ($\varepsilon = 1,..,23$). Todos ellos con idéntica demanda diaria (ver [3]: Tabla 6, Block I, NISSAN-9ENG).
- Demanda diaria: $T \equiv D_\varepsilon = 270\ unidades$ ($\forall \varepsilon = 1,..,23$).

Los códigos compilados de los procedimientos han sido ejecutados en un iMac (Intel Core i7 2.93 GHz, 8 GB de RAM). Las características de los tres procedimientos son:

- BDP-2: Algoritmo BDP con preservación del mix de producción en dos versiones (2/1 y 2/2) según pseudo-dominancias de vértices (ver [10]): (i) máximo número de transiciones desde cada vértice igual al número de tipos de producto $|I| = 9$; (ii) anchos de ventana $H = (1, 36, 81, 126)$ para los 23 planes de demanda (184 ejecuciones del algoritmo contemplando las 2 versiones); (iii) solución inicial Z_0 para H_n igual a la mejor solución obtenida con H_{n-1}, excepto para $H_1 = 1$, donde $Z_0 \to \infty$; (iv) tiempo de CPU medio empleado por plan de demanda igual a 5026.6 *s*.; y (v) preservación del mix de producción y libre interrupción de las operaciones con la asistencia la programación lineal (solver Gurobi).
- MILP: Modelo $4 \cup 3\_pmr$ (ver [9]): (i) implementación para solver Gurobi v4.5.0; (ii) tiempo de CPU máximo concedido a cada modelo por plan de demanda igual a 7200 *s*. (23 ejecuciones), empleando un tiempo medio de 6605.1; y (iii) con restricciones de preservación del mix de producción y con libre interrupción de las operaciones.
- GRASP-LP: (i) máximo número de iteraciones por plan de demanda igual a 10; (ii) tres valores del factor de admisión $\Lambda = (25\%, 50\%, 100\%)$ (690 soluciones en 69 ejecuciones); (iii) tiempo de CPU medio por plan empleado por las dos fases de GRASP igual a 425.3 *s*.; (iv) con restricciones de preservación del mix de producción; y (v) libre interrupción de las operaciones mediante el programa lineal LP-W tras obtener las mejores secuencias por GRASP [7,13], empleando un tiempo de CPU medio igual a 1.25 s. con solver CPLEX v11.0 ejecutado en iMac (Intel Core 2 Duo 2.33 GHz, 3 GB de RAM) usando un solo procesador.

La Tabla 2 recoge los mejores resultados conseguidos por BDP-2 (ver Tabla 2.20 en [10]), MILP (ver Table 3 en [9] columna $4 \cup 3\_pmr$) y GRASP-LP (este trabajo), para la sobrecarga W en los 23 planes de demanda $\varepsilon \in$ E. En ella se muestran también el algoritmo *Ganador* en cada plan de demanda y las ganancias unitarias de GRASP-LP frente a BDP-2 (ΔGvB), GRASP-LP frente a MILP (ΔGvM) y BDP-2 frente a MILP (ΔBvM), que se determinan así:

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

$$\forall \varepsilon \in \mathrm{E},\ \forall \mathcal{P} \in \{GRASP\_LP, BDP\_2\}, \forall \mathcal{P}' \in \{BDP\_2, MILP\} \qquad (24)$$

A partir del análisis de la Tabla 2 podemos afirmar:

- El procedimiento ganador en número de soluciones es BDP-2 con 12 mejores soluciones sobre 23; el segundo mejor procedimiento es GRASP-LP, que consigue la mejor solución en 7 ocasiones (planes 1, 7, 8, 10, 12, 17 y 23), mientras que MILP queda en última posición con 5 mejores soluciones (planes 3, 10, 19, 21 y

22). MILP y GRASP empatan en el plan 10, y MILP confirma como óptimas las soluciones de los planes 10 y 19.

- GRASP-LP vence a BDP-2 en 10 ocasiones de 23. La ganancia unitaria media de BDP-2 sobre GRASP-LP es del 15%, cuando BDP-2 es el ganador, mientras que la de GRASP-LP sobre BDP-2 es del 11%, cuando vence GRASP-LP. En media global, la ganancia unitaria de BDP-2 sobre GRASP-LP es sólo del 4%.
- GRASP-LP gana a MILP en 12 planes y empata en el plan 10. La ganancia unitaria media global de GRASP-LP sobre MILP es del orden del 6%. En detalle, GRASP-LP vence a MILP con una ganancia unitaria media parcial del 22%, y MILP vence a GRASP-LP parcialmente con ganancia del orden del 12%.
- BDP-2 vence a MILP 16 veces de 23. Las ganancias unitarias medias parciales, cuando BDP-2 vence a MILP y viceversa, son iguales al 13% y al 1%. BDP-2 vence globalmente a MILP con una ganancia del 9%.
- BDP-2, MILP y GRASP-LP necesitaron en promedio 5026.6 *s*, 6605.1 *s* y 426.6 *s*, respectivamente, para confirmar su mejor solución en cada plan de demanda.

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

$\varepsilon \in E$	W_{BDP_2}	W_{MILP}	W_{GRASP_LP}	ΔGvB	ΔGvM	ΔBvM	W_{BMG}	$Ganador$
1	166	186	**98**	0.69	0.90	0.12	98	GRASP-LP
2	**318**	383	342	-0.08	0.12	0.20	318	BDP-2
3	444	**423**	430	0.03	-0.02	-0.05	423	MILP
4	**305**	307	419	-0.37	-0.36	0.01	305	BDP-2
5	**633**	661	662	-0.05	-0.00	0.04	633	BDP-2
6	**428**	478	525	-0.23	-0.10	0.12	428	BDP-2
7	740	731	**728**	0.02	0.00	-0.01	728	GRASP-LP
8	112	160	**92**	0.22	0.74	0.43	92	GRASP-LP
9	**739**	751	911	-0.23	-0.21	0.02	739	BDP-2
10	1209	**1208**	**1208**	0.00	0.00	-0.00	1208	G-LP/MILP
11	**92**	122	96	-0.04	0.27	0.33	92	BDP-2
12	293	287	**268**	0.09	0.07	-0.02	268	GRASP-LP
13	**277**	336	294	-0.06	0.14	0.21	277	BDP-2
14	**381**	423	397	-0.04	0.07	0.11	381	BDP-2
15	**422**	442	429	-0.02	0.03	0.05	422	BDP-2
16	**216**	251	227	-0.05	0.11	0.16	216	BDP-2
17	466	488	**464**	0.00	0.05	0.05	464	GRASP-LP
18	**610**	619	698	-0.14	-0.13	0.01	610	BDP-2
19	949	**945**	948	0.00	-0.00	-0.00	945	MILP
20	**129**	150	169	-0.31	-0.13	0.16	129	BDP-2
21	565	**561**	725	-0.28	-0.29	-0.01	561	MILP
22	991	**984**	987	0.00	-0.00	-0.01	984	MILP
23	111	121	**107**	0.04	0.13	0.09	107	GRASP-LP
Media	-	-	-	-0.04	0.06	0.09	-	-

5 Conclusiones

BDP-2 queda como procedimiento estado del arte del problema MMSP-W/pmr/free, seguido por GRASP-LP a sólo el 4% en ganancia unitaria global, y por MILP al 9% y 6% de los anteriores. En tiempos de CPU, GRASP-LP es el más eficiente: 11.8 y 15.5 veces más rápido que BDP-2 y MILP, respectivamente, siendo importante la rapidez en nuestro caso, ya que 1 segundo de parada de línea supone un coste de 2.29 €. Como trabajos futuros, proponemos extender el método propuesto a algoritmos basados en Beam Search y Colonias de Hormigas.

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. Bautista, J., Batalla-García, C., Alfaro-Pozo, R.: Models for assembly line balancing by temporal, spatial and ergonomic risk attributes. European Journal of Operational Research 251(3), pp. 814-829, (2016).
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 minimisation and interruption rules. European Journal of Operational Research 210(3), pp. 495-513 (2011).
4. Cano-Belmán, J., Ríos-Mercado, R.Z., Bautista, J.: A scatter search based hyper-heuristic for sequencing a mixed-model assembly line. Journal of Heuristics 16(6), pp 749-770 (2010)
5. Bautista, J., Pereira, J., Adenso-Díaz, B.: A GRASP approach for the extended car sequencing problem. Journal of Scheduling 11, pp. 3-16 (2008).
6. Monden, Y.: Toyota Production System · An Integrated Approach to Just-In-Time. Springer US (1994).
7. Bautista, J., Alfaro-Pozo, R., Batalla-García, C.: GRASP for sequencing mixed models in an assembly line with work overload, useless time and production regularity. Progress in Artificial Intelligence 5(1), pp. 27-33 (2016).
8. Bautista, J., Cano, A., Alfaro, R., Batalla, C.: Impact of the Production Mix Preservation on the ORV Problem. Advances in Artificial Intelligence, Volume 8109 of the series Lecture Notes in Computer Science, pp. 250-259. Springer Berlin Heidelberg (2013).
9. 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(12), pp. 11001–11010 (2012).
10. Cano, A.: Modelado y resolución de problemas de secuenciación en contexto JIT/DS mediante BDP. Tesis doctoral UPC: http://hdl.handle.net/10803/275986 (2014)
11. 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).
12. Vanderbei, R.J.: Linear Programming · Foundations and Extensions. International Series in Operations Research & Management Science, Volume 114. Springer US (2008).
13. Bautista, J., Alfaro, R.: Mejores soluciones del conjunto de instancias Nissan-9Eng.I para el problema MMSP-W/pmr/free. Report de Recerca UPC: http://hdl.handle.net/2117/88164 (2016).