Combinación de reglas bajo un esquema de búsqueda dispersa para secuenciar modelos mixtos en líneas de montaje

Jaime Cano Belmán [a] Roger Z. Ríos Mercado [b] Joaquín Bautista Valhondo [c]

Resumen— **Se presenta una hiperheurística para el problema de secuenciación de modelos mixtos en líneas de montaje. El criterio de secuenciación es la minimización de la sobrecarga producida en las estaciones de la línea de montaje debido a las diferentes cantidades de trabajo necesarias para procesar diferentes versiones de los productos. Secuencias largas de productos ricos en contenido de trabajo pueden producir sobrecarga cuando en las estaciones no se tiene la capacidad de terminar los trabajos en el espacio temporal destinado para ello. El nivel superior de la hiperheurística esta basado en la metodología de búsqueda dispersa (SS), mientras que el nivel inferior trabaja con combinación de reglas de prioridad. Se proponen dos esquemas generales de la hiperheurística y se prueban diferentes niveles de mejoramiento. Los resultados experimentales muestran que mejoras locales en las secuencias del conjunto de referencia final o pequeñas mejoras en las secuencias obtenidas por combinación de reglas producen buenas soluciones.**

Palabras clave— **secuenciación, reglas de prioridad, búsqueda dispersa, justo a tiempo, hiperheurística**

I. Introducción

En los sistemas de gestión de producción tipo arrastre (pull), concretamente, dentro del marco de la filosofía de producción justo a tiempo (JIT por sus siglas en inglés), la secuenciación de operaciones se convierte en el aspecto fundamental del sistema [1]. Las líneas de montaje de productos mixtos (MMAL) permiten manufacturar productos con muchas y pequeñas variantes, evitando inventarios y tiempos de preparación importantes. Este tipo de líneas se usa por ejemplo en el montaje de automóviles. La secuenciación de los productos es una decisión importante para el uso eficiente de las líneas de montaje. La situación se puede caracterizar como un problema de ordenación con objetivos representables mediante alguna medida de eficiencia. En la literatura se pueden encontrar diferentes objetivos de secuenciación. El objetivo depende de las políticas administrativas de la compañía y de las restricciones impuestas por el sistema ([2], [3]).

Uno de los criterios que menciona Monden [4] como objetivo de secuenciación en un entorno JIT es nivelar la carga de trabajo en cada proceso en las líneas de trabajo de productos mixtos. En este trabajo el criterio de secuenciación es la minimización de la sobrecarga. La sobrecarga puede entenderse como el trabajo que no puede ser completado dentro de una estación cerrada debido a que el tiempo de proceso de la unidad en curso es más grande que el tiempo disponible [5]. Al haber diferentes modelos de un mismo producto que deben ser ensamblados en la misma línea, algunos tendrán un tiempo de proceso mayor que el tiempo de ciclo en las estaciones de la línea de montaje. Si se introducen consecutivamente productos con tiempos de proceso mayores al tiempo de ciclo, llegará un momento en que no se tenga tiempo suficiente en la estación para terminar el trabajo sobre alguno de estos productos y el trabajo quedará incompleto cuando el producto salga de la estación.

Este trabajo que excede la capacidad o disponibilidad de trabajo en las estaciones recibe entre otros nombres, el de sobrecarga (*work overload* [5], [6]), trabajo perdido o pendiente (*remaining work*, [7]) o trabajo utilitario (*utility work*, [8]). Cuando la filosofía de gestión de la empresa obliga a terminar el trabajo dentro de las estaciones sin recibir ayuda extra se usa el término: paro de línea (*conveyor stoppage*, [9]), ya que los trabajadores tienen la posibilidad de detener el avance de la línea para poder terminar el trabajo sobre los productos dentro de la estación. En el presente trabajo se adopta el concepto de sobrecarga usado en [5] y [6].

Para resolver este problema NP-duro [5] se han propuesto procedimientos exactos basados en ramificación y acotamiento [10] o programación dinámica [5]. Sin embargo, los procedimientos exactos son eficientes sólo para intancias pequeñas o que consideran una única estación o productos opcionales. Se han propuestos procedimientos heurísticos prospectivos [11], voraces [5] y procedimientos basados en reglas de espacios [12] para versiones diferentes del problema. En la literatura pueden encontrarse procedimientos que consideran múltiples objetivos [3], y enfoques metaheurísticos [13], híbridos multiobjetivo [14] y basados en enumeración implícita parcial (*beam search*) [15]. Una revisión reciente de la literatura relacionada con el problema de secuenciación puede encontrarse en Boysen, Fliedner y Scholl [16].

En el presente trabajo se propone el uso de reglas de prioridad dentro de un esquema mayor. Las reglas de prioridad son un elemento común en procedimientos heurísticos para problemas de secuenciación. Un

[a] Universidad Autónoma de Nuevo León. E-mail: jaime@yalma.fime.uanl.mx
[b] Universidad Autónoma de Nuevo León. E-mail: roger@yalma.fime.uanl.mx
[c] Universitat Politècnica de Catalunya. E-mail: catedra.nissanmotoriberica@nobel.upc.edu

algoritmo heurístico ofrece mejores soluciones cuanto mayor son los aspectos que combina la regla. Las reglas de prioridad se usan para establecer una lista ordenada de productos candidatos a ser secuenciados en cada etapa de decisión.

El esquema mayor dentro del cual se usan las reglas de prioridad esta basado en la metodología de búsqueda dispersa [17]. El esquema típico de esta metodología poblacional considera un método generador de diversidad, un método de mejora, un método para actualizar el conjunto de referencia, un método generador de subconjuntos y un método de combinación. Como se explica más adelante, el procedimiento propuesto aquí difiere del original en algunos aspectos. El principal de ellos radica en que el conjunto de referencia contiene cadenas de reglas de prioridad en lugar de soluciones al problema.

Enfoques hiperheurísticos con búsqueda dispersa como esquema mayor ya han sido aplicados a probemas de balanceo de líneas [18], [19] con buenos resultados. Otras propuestas hiperheurísticas pueden encontrarse en [20] para problemas de horarios o en [21] para problemas de empaquetamiento.

Este trabajo está organizado de la siguiente manera: en la Sección II se describen las características del sistema productivo y se da una formulación del problema, la Sección III describe los detalles del procedimiento constructivo por combinación de reglas, mientras que en la Sección IV se describe el esquema general de búsqueda dispersa aquí propuesto. Los resultados computacionales se muestran en la Sección V y las conclusiones del trabajo en la Sección VI.

II. Descripción y formulación del problema

El problema se estudia asumiendo las siguientes características y políticas del sistema productivo: velocidad constante del transportador, estaciones cerradas, tiempos de proceso deterministas, introducción de productos en la línea a intervalos de tiempo constantes, dos operadores trabajan simultáneamente en cada estación y realizan su trabajo tan pronto como sea posible. Al tiempo que transcurre entre dos lanzamientos consecutivos se le denomina tiempo de ciclo (c). Existen cargas de trabajo expresadas en unidades de tiempo de proceso p_{ik} para cada producto i ($i = 1, \ldots, I$) en cada estación k de la línea ($k = 1, \ldots, K$). La longitud de las estaciones L_k también es expresada en unidades de tiempo. Se incurre en sobrecarga w cuando no es posible terminar el trabajo sobre algún producto dentro de los límites de alguna estación. Una solución factible para el problema está dada por una secuencia de longitud T que contiene las unidades demandadas n_i de cada producto i, de tal manera que $T = \sum_{i=1}^{I} n_i$. Una solución óptima será aquella que minimice la suma de las sobrecargas producida en todas las estaciones.

Sea Π la colección de permutaciones de T objetos de los cuales n_i son de tipo i, el problema consiste en encontrar una permutación $\pi = (\pi_1, \pi_2, \ldots, \pi_T)$ que minimiza el valor asociado de sobrecarga (1) y satisface las restricciones de dezplazamiento (2).

$$\underset{\pi \in \Pi}{\text{Minimizar}}\, w(\pi) = \sum_{k=1}^{K} \sum_{t=1}^{T} w_{k,\pi_t} \qquad (1)$$

Sujeto a:

$$\pi \in F = \{\pi : s_{k,\pi_t} + p_{k,\pi_t} - w_{k,\pi_t} - c \leq s_{k,\pi_{t+1}},$$
$$s_{k,\pi_t} + p_{k,\pi_t} - w_{k,\pi_t} \leq L_k\}$$
$$k = 1, \ldots, K; t = 2, \ldots, T \qquad (2)$$

Donde las variables s_{k,π_t} and w_{k,π_t} representan la posición del trabajador en la estación k al iniciar el trabajo sobre el producto en la posición π_t de la permutación, y la sobrecarga producida en la estación k después de haber terminado el trabajo sobre el producto en la posición π_t, respectivamente. Dichas variables se pueden obtener recursivamente según las expresiones (3) y (4). En general se asume que $s_{k,\pi_1} = 0$.

$$s_{k,\pi_t}(\pi) = \text{máx}\{\text{mín}\{s_{k,\pi_{t-1}} + p_{k,\pi_{t-1}}, L_k\} - c, 0\}\}$$
$$t = 2, ..., T \qquad (3)$$

$$w_{k,\pi_t}(\pi) = \text{máx}\{s_{k,\pi_t} + p_{k,\pi_t} - L_k, 0\}$$
$$t = 1, ..., T \qquad (4)$$

III. Construcción de soluciones con reglas prioridad

En un intento por aprovechar las ventajas que pueden obtenerse del conocimiento del problema mediante el uso de reglas de prioridad dentro de un esquema mayor, se propone una hiperheurística. Una hiperheurística puede verse como una heurística que selecciona otras heurísticas para resolver un problema. Las estrategias de combinación de reglas se basan en que la información sobre el interés de las posibles alternativas se guardan de distinta manera por las diferentes reglas y de que esa información puede aprovecharse mediante mecanismos de combinación.

En el nivel bajo, la hiperheurística se apoya en un procedimiento constructivo por combinación de reglas (PCCR). Dada una cadena de reglas $C = (r^1, r^2, \ldots, r^T)$ ($r \in R$), el PCCR obtiene una secuencia de productos (solución) equivalente $S = (s^1, s^2, \ldots, s^T)$. En cada etapa t, el procedimiento selecciona el producto i que mejor satisface la cadena r y lo asigna en la posición actual de la solución. El pseudocódigo del procedimiento se muestra en la Figura 1.

En este trabajo se usan 20 reglas de prioridad ($|R| = 20$). Las reglas consideran aspectos como tiempos de procesamiento, la demanda pendiente, la diferencia absoluta entre el tiempo de proceso de cada producto en cada estación y el tiempo de ciclo, los desplazamientos de los trabajadores, la estación cuello de botella, la sobrecarga, el tiempo improductivo

PCCR (C, n_i)
0 $S = \emptyset$, $d_i = n_i$;
1 **para** ($t = 1$ hasta T) **hacer**
2 $s^t \leftarrow i^* \in r^t$;
3 actualizar d_{i^*};
4 **fin para**
5 **devolver**(S);

Fig. 1. PCCR

y la regularidad en la producción y en la sobrecarga. Los detalles de las reglas se muestran en el Apéndice. Una vez que se obtiene una solución S su correspondiente valor de sobrecarga se puede evaluar según las expresiones (3) y (4).

IV. Hiperheurística

El nivel superior de la hiperheurística que aquí se propone (HH) se basa en la búsqueda dispersa (SS). Dicha metología opera sobre el llamado conjunto de referencia (RS) en el que se almacenan buenas soluciones encontradas a lo largo de la búsqueda. Buenas soluciones se refiere a soluciones de alta calidad y a soluciones diversas. Típicamente SS funciona como sigue: se generan soluciones diversas mediante un método generador de diversidad. Estas soluciones pueden ser mejoradas mediante el método de mejora. Mediante un método de combinación se obtienen nuevas soluciones combinando las existentes en el RS a las que también se les puede aplicar un procedimiento de mejora. Las parejas a combinar se seleccionan mediante el método generador de subconjuntos. En cada etapa del proceso se actualiza el RS con el método de actualización, guardando así las mejores soluciones.

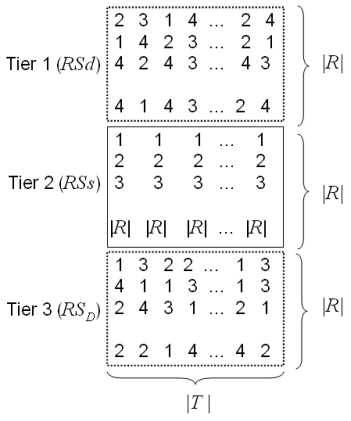

Fig. 2. Conjunto de referencia de 3 partes

En HH el RS difiere de uno normal en que el nuestro contiene cadenas de reglas de prioridad. Diferenciamos dos tipos de RS: $RS1$ y $RS2$. El $RS1$ esta compuesto por dos partes (*tiers*): un subconjunto dinámico (RS_d) y uno estático (RS_s). RS_d es actualizado en cada iteración. Al iniciar el procedimiento las reglas del RS_d son determinadas de manera aleatoria. RS_s permanece fijo a lo largo de la búsqueda. Cada cadena j del subconjunto RS_s contiene unicamente la regla r ($j = r$). Con ello se asegura que todas las reglas se siguen tomando en cuenta a lo largo del proceso aunque no produzcan buenas soluciones, ya que pueden ayudar a diversificar la búsqueda. Ambos subconjuntos del RS guardan tantas cadenas de prioridad como $|R|$. El $RS2$ está conformado además por un tercer subconjunto que almacena cadenas diversas (RS_D). Si se usa el $RS1$ es necesario aplicar una fase de diversificación explícita, mientras que si se usa el $RS2$ esta fase ya está implícita en el método de combinación. En HH no se usa un método generador de diversidad tal cual como se concibe en la metodología de SS. Por lo anterior, diferenciamos dos procedimientos HH1 y HH2 que corresponden al $RS1$ y $RS2$ respectivamente. La Figura 2 muestra un RS con tres subconjuntos. El esquema $HH1$ correspondiente al $RS1$ se muestra en al Figura 3, y el esquema $HH2$ del $RS2$ se muestra en al Figura 4.

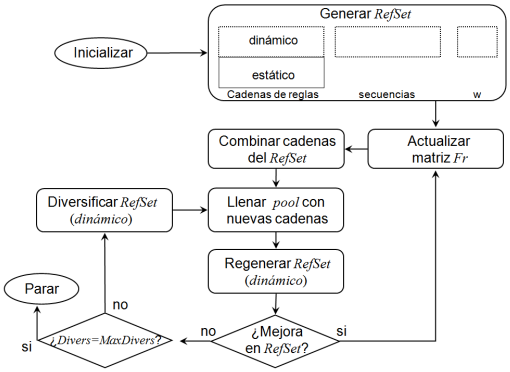

Fig. 3. Esquema HH1 para RS1

En la fase inicial del procedimiento se crea el RS con cadenas de reglas de prioridad. En HH el conjunto de referencia inicial se crea seleccionando aleatoriamente las reglas para conformarlo. Una vez creado el RS se calcula la matriz de frecuencias Fr, y se actualiza en cada iteración del procedimiento. La matriz de frecuencias Fr es un elemento importante, ya que contiene información sobre la frecuencia de aparición de las reglas de prioridad en las cadenas del RS.

El siguiente paso es la combinación de las cadenas del RS. La combinación se realiza considerando la información contenida en la matriz de frecuencias Fr. Al conjunto de cadenas generadas se le denomina *pool*. La regeneración del conjunto de referencia es el siguiente paso. En la regeneración se selecciona un conjunto de cadenas *buenas*, considerando las cadenas existentes en el RS y en el *pool*. Ese grupo de cadenas buenas se introducen en el RS, lo cual mejorará su estado hasta que después de varias iteraciones no mejore más. Entonces se procede a la diversificación, la cual también se realiza en función de la matriz de frecuencias. En el esquema de HH2

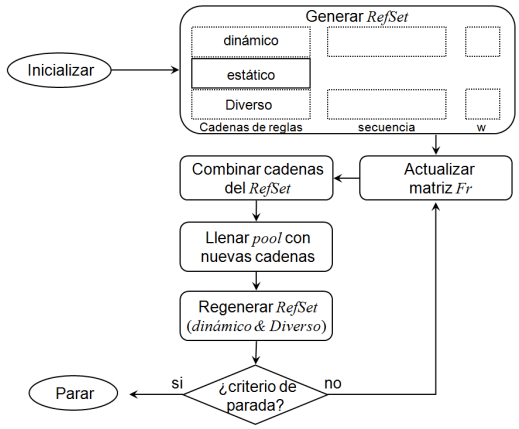

Fig. 4. Esquema HH2 para RS2

(Figura 4) no se considera una fase de diversificación explícita ya que el RS contiene cadenas diversas. Los detalles de cada paso se presentan en las secciones siguientes.

A. Método de combinación

Con el método de combinación de cadenas podemos producir otras nuevas que mantienen características de las anteriores. El método de combinación se basa en la matriz de frecuencias $Fr(r,t)$. Esta matriz de frecuencias de dimensiones $(|R|,T)$ contiene el número de veces que la regla r ha aparecido en la posición t de las cadenas de reglas contenidas en el RS. Por ejemplo, $Fr(3,8)=15$ indica que la regla $r=3$ ha aparecido 15 veces en la posición $t=8$ de las cadenas del RS.

$$C_k^t = \begin{cases} C_p^t & \text{si } C_p^t = C_q^t \\ C_p^t & \text{si } Fr(C_p^t,t) \geq Fr(C_q^t,t) \\ C_q^t & \text{de otro modo} \end{cases} \quad (5)$$

Así, dadas dos cadenas $C_q = (r_q^1, r_q^2, \ldots, r_q^T)$ y $C_p = (r_p^1, r_p^2, \ldots, r_p^T)$, la regla que contendrá la nueva cadena C_k en la posición t está determinada según la expresión (5).

Dado que el RS contiene cadenas que han producido buenas soluciones y que Fr se actualiza en cada iteración según la información del RS, el método de combinación debería guiar la búsqueda a mejores soluciones. Usando la misma idea, cuando se usa el $RS1$, se pueden aprovechar las reglas que han aparecido con menor frecuencia para diversificar la búsqueda. Todas las cadenas obtenidas con el método de combinación debe ser sometidas al $PCCR$ para obtener la secuencia de productos equivalente y su correspondiente valor de sobrecarga, todo lo cual se guarda en el llamado *pool*.

B. Método de mejora

Se aplica un método de mejora por reinserción de segmentos [22]. Dada una solución S, su vecindario $N(S)$ es el conjunto de todas las soluciones alcanzables desde S realizando un movimiento *move*. Este *move* consiste en seleccionar un segmento de *seg* elementos consecutivos de S, extraerlo de su posición original e insertarlo en una posición diferente (Figura 5).

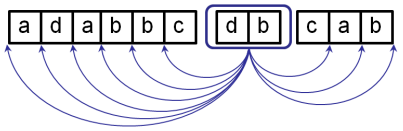

Fig. 5. Mejora por reinsercion de segmentos

Dado T y un segmento de longitud *seg*, todos los posibles movimientos son $|N(S)| = (T-seg)^2$. Cada vez que un vecindario es explorado, se selecciona un tamaño de segmento $1 \leq seg \leq 15$. Para instancias pequeñas con $T \leq 15$, el valor máximo permitido es de 8. La selección del tamaño de *seg* se realiza probabilísticamente, donde aquellos tamaños de segmento que han producido mayor número de mejoras, tienen mayor probabilidad de ser seleccionados. En HH el procedimiento de mejora puede ser aplicado en las secuencias del RS o del *pool*. En función del grado de calidad que se desea puede variarse el grado de profundidad de mejora, principalmente si se aplica la mejora en las secuencias del *pool*. En HH la combinación de dos cadenas de reglas (del RS) produce una nueva cadena. Dado $|R| = 20$ reglas y el esquema HH1, el tamaño del $pool = (2 \cdot |R|)!/2! \cdot 38! = 780$ cadenas. Después de evaluar las cadenas del *pool*, se consideran aquellas cadenas con mejores valores de sobrecarga para actualizar el RS_d. A este proceso se le llama actualización o regeneración del RS.

C. Actualización del conjunto de referencia

Una vez que se han realizado todas las combinaciones de cadenas por pares y se tienen todas las cadenas descendientes se realiza la actualización o regeneración del RS, esto es, se consideran las cadenas de reglas del *pool* con mejores valores de sobrecarga para ser incorporadas en la parte operativa del conjunto de referencia (RS_d). Específicamente, las $|R|/2$ peores cadenas del RS son sustituidas por las $|R|/2$ mejores del *pool*. Estas mejores cadenas pueden ser el resultado de haber aplicado el método de mejora sobre ellas. Siempre se tiene almacenada la mejor solución encontrada hasta el momento. En el proceso de regeneración del RS se evita duplicar las cadenas de reglas, por lo tanto, dentro del RS existirán cadenas con valor de sobrecarga diferentes. Una vez terminada la fase de regeneracion del RS, se evalúa el estado del mismo. Para ello, en cada iteración se guardan los valores de sobrecarga de la mejor y de la peor cadena del RS. El estado del RS está en función de dichos valores. Se considera que el estado del RS ha mejorado si al menos en una de las cadenas de referencia ha habido una mejora en el valor de la

sobrecarga. En este punto del proceso se actualiza la matriz de frecuencias Fr. Cuando se usa el esquema HH2, también debe actualizarse el subconjunto RS_D del RS. El proceso inicia borrando las cadenas en RS_D. Iterativamente, para cada cadena en el *pool* diverso, se calcula su distancia $d(C, RS_d, RS_D)$. Dicha distancia es la llamada distancia de Hamming. La cadena candidata $C \in pool$ se compara con las cadenas del RS_d y las existentes en RS_D. La cadena $C \in pool$ con mayor medida de distancia es introducida al RS_D.

D. Fase de diversificación

Dentro del esquema HH1, dado que el RS no cuenta con subconjunto de diversidad, se aplica explícitamente un proceso de diversificación con la intención de mover la busqueda a otra zona del espacio. Así, cuando la combinación de cadenas deja de producir mejoras en el estado del conjunto de referencia se pasa a la fase de diversificación. Para ello se aprovecha la información que contiene la matriz de frecuencias $Fr(r,t)$.

$$C_k^t = \begin{cases} C_p^t & \text{si } C_p^t = C_q^t \\ C_p^t & \text{si } Fr(C_p^t, t) < Fr(C_q^t, t) \\ C_q^t & \text{de otro modo} \end{cases} \quad (6)$$

La diversificación se realiza en dos fases. La primera de ellas consiste en generar cadenas diversas. Ello se logra combinando las cadenas del RS, pero esta vez no se buscan las reglas con mayor frecuencia en Fr, sino aquellas con menor frecuencia como se muestra en (6). En la segunda fase, se seleccionan de forma iterativa las cadenas diversificadas del *pool* que son más diferentes respecto a las cadenas contenidas ya en el RS. El grado de diferenciación entre dos cadenas se mide con el número de no coincidencias que existe entre ambas cadenas (distancia de Hamming). Existe una coincidencia si en la misma posición t de ambas cadenas se encuentra la misma regla r. En la diversificación se buscan aquellas cadenas con el mayor número de no coincidencias. Cada cadena candidata en el *pool* es comparada con cada una de las cadenas existentes en ese instante en el RS. El número total de no coincidencias de una cadena candidata respecto al RS será la suma de las no coincidencias de la cadena candidata con la primera cadena del RS, mas las no coincidencias con la segunda cadena del RS, y así sucesivamente. Después de obtener esta medida de distancia para todas las cadenas del *pool* diverso, se introduce al RS aquella candidata con la mayor distancia respecto al RS. El proceso se repite hasta llenar el RS con cadenas diversas.

V. Experiencia computacional

Para probar la bondad de HH se realizaron pruebas computacionales en una PC con procesador pentium 4 a 2.4 GHz, 512 MB RAM en sistema operativo windows XP profesional 2002. Las pruebas se realizaron sobre una bateria de 100 instancias diseñadas y generadas en la literatura [13]. Todas las instancias tienen un tiempo de ciclo $c = 90$ unidades de tiempo. Los tiempos de proceso fueron generados de la siguiente manera: para cada producto i se genera aleatoriamente T_i entre el intervalo $[0{,}75 \cdot c \cdot K,]$, luego cada p_{ik} se genera aleatoriamente dentro del intervalo $[0{,}5 \cdot T_m/K, \text{mín}\{l_k, 1{,}5 \cdot T_m/K\}]$. Las instancias son generadas con diferentes valores de I (5,10 y 20), de K (5,10 y 20) y longitud de estación L_k (110, 150 y [88,132]) en unidades de tiempo. Además las instancias tienen diferentes longitudes de sequencia T. Los valores de T son: 10, 20, 30, 50, 80,110, 140, 170, 200, 250, 300, 350 y 400. Los valores de las demandas n_i se generan aleatoriamente entre los intervalos $[0{,}5 \cdot T/I, 1{,}5 \cdot T/I]$.

Dado que sólo conocemos óptimos confirmados para algunas instancias y a falta de buenas cotas inferiores conocidas aún para instancias pequeñas, en la experiencia computacional comparamos los resultados con las mejores valores obtenidos. Debe mencionarse también que se sabe a priori que los resultados para algunas instancias puede tomar valor $w = 0$. Para salvar los detalles anteriores se usa la siguiente medida de calidad dev. La desviación dev de una solución S para la instancia h respecto a la mejor solución obtenida S_{best} se obtiene según (7).

$$dev_h = |S_{best} - S_h|/|\epsilon + S_h| * 100\,\% \quad (7)$$

Con la finalidad de detectar algunas estrategias en la aplicación del procedimiento de mejora, se distinguen diferentes variantes que aplican para HH1 y HH2. Se consideran 4 niveles de mejora: NoI, $IRSf$, $IP1N$ y $IP10\,\%$. En NoI no se aplica mejora en ningún punto de HH. $IRSf$ toma el resultado de NoI y aplica mejora local sobre las secuencias del RS_d final. En $IP1N$ se aplica el procedimiento de mejora sobre las secuencias del *pool*. En este caso se explora únicamente el vecindario inmediato de las secuencias del *pool*. Dado un segmento de tamaño seg hay $(T - seg)^2$ $moves$ posibles. En la variante $IP10\,\%$ también se aplica mejora local sobre las secuencias del *pool*. La profundidad de la mejora se establece en $10\,\%$. Este $10\,\%$ de mejora se determina respecto a la cota lbw. Si el valor de sobrecarga original es w_0, la mejora termina cuando la cantidad mejorada iguala o supera el valor $(w_0 - lbw)/10$, donde lbw es la diferencia entre la cantidad de trabajo (en tiempo) necesario para producir toda la demanda n_i y el tiempo disponible para ello, lo cual se expresa en (8).

$$lbw = \sum_{k=1}^{K} \left[\sum_{i=1}^{I} n_i p_{ik} - (c(T-1) + L_k) \right]^+ \quad (8)$$

En HH1, se establece una cantidad máxima de $Nomejoramaxcombs = 3$ iteraciones sin mejora en

el estado del RS para aplicar el proceso de diversificación. El máximo de diversificaciones sin mejora $MaxDivs = 4$. Se estableció un tiempo maximo de busqueda de 3600 segundos en el bucle iterativo de ambos HH1 y HH2. La mejora local en el RS final se aplica después de este criterio de paro dentro del bucle. Para HH2 el número máximo de diversificaciones sin mejora se estableció en $MaxDivs = 4$. Con miras a tener una referencia, se consideran las soluciones encontradas con el software de optimización CPLEX9.0.

TABLA I
MEDIDAS DE DESEMPEÑO PARA HH1

		Grado de mejora en HH1			
proc.	CPLEX	NoI	IRSf	IP1N	IP10 %
Av.*dev*(%)	21.04	18.71	1.94	0.09	1.77
opt*	18	16	23	25	25
Av.Iters	-	50.96	50.96	6.24	6.37
Av.Divs	-	4.95	4.95	0.39	0.48
Av.CPU(seg)	3600	269	504	12545	16275

TABLA II
MEDIDAS DE DESEMPEÑO PARA HH2

		Grado de mejora en HH2			
proc.	CPLEX	NoI	IRSf	IP1N	IP10 %
AV. *dev*(%)	21.04	20.59	5.20	0.52	0.32
opt*	18	11	24	23	25
Av.Iters	-	10.8	10.8	2.7	1.3
AV.CPU(seg)	3600	510	555	10275	3596

Las Tablas I y II muestran las medidas de desempeño para $HH1$ y $HH2$, respectivamente, como la desviación promedio AV.*dev*(%) respecto al mejor valor encontrado, el número de óptimos alcanzados $opt*$ por cada variante del procedimiento, el promedio de iteraciones hechas por procedimiento, el promedio de fases de diversificación (sólo en Tabla I), y el promedio de tiempo de CPU en segundos requerido por cada versión de HH (Av.CPU(seg)). En $HH1$ el número de óptimos aumenta de 16 hasta 25, cuando se aplica procedimiento de mejora en alguna parte del procedimiento. Aunque $HH2$ encuentra menos óptimos cuando se usa sin mejora, al aplicarla también se pueden encontrar hasta 25 óptimos. Debe notarse que el número de iteraciones promedio para $HH2$ es una quinta parte si se compara con las de $HH1$. Esto se atribuye al tamaño del RS y al esfuerzo adicional en la fase de combinación y en la mejora de las soluciones del *pool*.

Con CPLEX9.0 y despúes de una hora de búsqueda, se encontraron los valores óptimos para 18 de las 100 instancias de la batería. Para siete instancias más se encontró con alguno de los procedimientos aquí propuestos, soluciones cuyo valor equilave a la cota inferior encontrada al aplicar el método de ramificación y acotamiento de CPLEX9.0. Para estas siete instancias, en promedio, CPLEX está 29.95 % por arriba del mejor valor de sobrecarga encontrado con HH.

En general, con HH se encuentran soluciones óptimas para los dos grupos de instancias pequeñas (T =(10, 20, 30), L =(110, [88,132])), y para el grupo de instancias con estaciones largas (L =150, P =10, M =10, T =(140, 200, 250, 300, 350)). El procedimiento que ofrece el mejor compromiso entre calidad y tiempo de CPU es $IRSf$ el cual aplica mejora local únicamente a las secuencias del RS final.

VI. Conclusiones

Se estudia el problema de secuenciación productos mixtos en líneas de montaje, con el criterio de minimización de sobrecarga. Se consideran estaciones cerradas, velocidad contante del transportador, tiempos de proceso deterministas, introducción de productos en la línea a intervalos de tiempo constantes, los(las) operadores(operadoras) realizan su trabajo tan pronto como sea posible, los tiempos de preparación y de dezplazamiento de los trabajadores es despreciable. Se propone una hiperheurística HH basada en la metodología de búsqueda dispersa como nivel superior. En el nivel inferior de HH se propone el uso de procedimientos constructivos por combinación de reglas. Se describen dos esquemas generales de HH y se prueban diferentes niveles de profundidad en el método de mejora. La mejora local si bien ayuda a obtener buenos resultados, es costosa, principalmente cuando se aplica a las soluciones del *pool*, donde puede perderse la ventaja de una heurística sobre un procedimiento exacto. El procedimiento de mejora es mejor aprovechado cuando se aplica sobre cadenas del conjunto élite.

Agradecimientos

Este trabajo se realizó durante una estancia postdoctoral del primer autor en la Universidad Autónoma de Nuevo León, México, en el marco de apoyos a estancias postdoctorales del Consejo Nacional de Ciencia y Tecnología (CONACyT), México. Agradecemos también los apoyos dados a través de PROTHIUS-II, proyecto DPI2007-63026 del Gobierno de España, y por la Cátedra Nissan UPC.

Referencias

[1] J. Bautista Valhondo, *Procedimientos heurísticos y exactos para la secuenciación en sistemas productivos de unidades homogéneas (contexto JIT)*, Disertación doctoral, Universitat Politècnica de Catalunya, Barcelona, España, Abril 1993.

[2] K. Okamura and 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*, vol. 17, no. 3, pp. 233–247, 1979.

[3] S. Kotani, T. Ito, and K. Ohno, "Sequencing problem for a mixed-model assembly line in the Toyota production system," *International Journal of Production Research*, vol. 42, no. 23, pp. 4955–4974, 2004.

[4] Y. Monden, *Toyota Production System*, Norcross, GA, 1983.

[5] C. A. Yano and R. Rachamadugu, "Sequencing to minimize work overload in assembly lines with product options," *Management Science*, vol. 37, no. 5, pp. 572–586, 1991.

[6] J. Bautista and J. Cano, "Minimizing work overload in mixed-model assembly lines," *International Journal of Production Economics*, vol. 112, no. 1, pp. 177–191, 2008.
[7] A. Bolat, "Efficient methods for sequencing minimum job sets on mixed model assembly lines," *Naval Research Logistics*, vol. 44, no. 5, pp. 419–437, 1997.
[8] L.H. Tsai, "Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage," *Management Science*, vol. 41, no. 3, pp. 485–495, 1995.
[9] Z. Xiaobo and K. Ohno, "Algorithms for sequencing mixed models on assembly line in a JIT production system," *Computers & Industrial Engineering*, vol. 31, no. 1, pp. 47–56, 1997.
[10] A. Bolat, "A mathematical model for sequencing mixed models with due dates," *International Journal of Production Research*, vol. 41, no. 5, pp. 897–918, 2003.
[11] C. A. Yano and A. Bolat, "Survey, developement, and application of algorithms for sequencing paced assembly lines," *Journal of Manufacturing and Operations Management*, vol. 2, no. 3, pp. 172–198, 1989.
[12] A. Bolat and C. A. Yano, "Scheduling algorithms to minimize utility work at a single station on paced assembly line," *Production Planning and Control*, vol. 3, no. 4, pp. 393–405, 1992.
[13] A. Scholl, R. Klein, and W. Domschke, "Pattern based vocabulary building for effectively sequencing mixed-model assembly lines," *Journal of Heuristics*, vol. 4, no. 4, pp. 359–381, 1998.
[14] A. Rahimi-Vahed and A. H. Mirzaei, "A hybrid multi-objective shuffled frog-leaping algorithms for a mixed-model assembly line sequencing problem," *Computers & Industrial Engineering*, vol. 53, no. 4, pp. 642–666, 2007.
[15] E. Erel, Y. Gocgunz, and I. Sabuncuoğlu, "Mixed-model assembly line sequencing using beam search," *International Journal of Production Research*, vol. 45, no. 22, pp. 5265–5284, 2007.
[16] N. Boysen, M. Fliedner, and A. Scholl, "Sequencing mixed-model assembly lines: Survey, classification and model critique," *European Journal of Operational Research*, vol. 192, no. 2, pp. 349–373, 2009.
[17] M. Laguna and R. Martí, *Scatter Search*, Kluwer Academic Publishers, Boston, 2003.
[18] J. Bautista, R. Suárez, M. Mateo, and R. Companys, "Local search heuristics for the assembly line balancing problem with incompatibilities between tasks," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, April 24-28 2000, pp. 2404–2409.
[19] J. Bautista, E. Fernández, J. L. González Velarde, and M. Laguna, "Hiperheuristica para un problema de equilibrado de lineas de montaje usando scatter search," in *Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, Granada, España, Septiembre 2005, pp. 839–845.
[20] E. Burke and S. Petrovic, "Recent research directions in automated timetabling," *European Journal of Operational Research*, vol. 140, no. 2, pp. 266–280, 2002.
[21] P. Ross, S. Schulenburg, J. Marin-Blazquez, and E. Hart, "Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002*, Morgan Kauffmann, Ed., New York, July 9-13 1997, pp. 942–948.
[22] J. Cano Belmán, *Modelos y algoritmos de secuenciación en líneas de ensamble de productos mixtos*, Disertación doctoral, Universitat Politècnica de Catalunya, Barcelona, España, Diciembre 2006.

Apéndice

Las 20 reglas de prioridad usadas en HH se describen en esta sección. Dentro el PCCR, un producto i es candidato a cierta regla r sólo si $d_i > 0$. Sea:

I conjunto de productos;
i producto i, $i \in I$;
K conjunto de estaciones;
k estacione k, $k \in K$;
n_i demanda original del producto i, $i \in I$;
t posición o etapa de la secuencia, $(t = 1, \ldots, T)$, $T = \sum_{i=1}^{I} n_i$;
d_i demanda pendiente del producto i (if $t = 0$, $d_i = n_i$), $i \in I$, $t = 1, \ldots, T$;
p_{ik} tiempo de proceso para el producto i en la estación k, $i \in I$, $k \in K$;
r_{ik} indice dinámico para el producto i en la estación k, $r_{ik} = d_i * \Delta_{ik}$, $i \in I$, $k \in K$;
Δ_{ik} desplazamiento del trabajador en la estación k debida al producto i, $\Delta_{ik} = |p_{ik} - c|$, $i \in I$, $k \in K$;
L_k longitud de la estación k, $k \in K$;
u_i tasa producción ideal para el producto i, $i \in I$, $u_i = n_i/T$;
s_k posición inicial del trabador en la estación k, $k \in K$;
w_t sobrecarga acumulada hasta la etapa $t-1$, $w_t = \sum_1^t w_{tk}$, $k \in K$;
$\bar{p}$ tiempo de proceso promedio $\bar{p} = \sum p_{ik}/(|K| \cdot |I|)$, $i \in I$, $k \in K$;
k^b estación cuello de botella;
w_i sobrecarga producida por el producto i en la etapa actual t, $w_i = \sum_{k=1}^{K} w_{ti}$, $i \in I$, $k \in K$;
o_i tiempo de ocio en la etapa actual t producido por el producto i, $o_i = \sum_{k=1}^{K} o_{ti}$, $i \in I$, $k \in K$.

Las reglas se enumeran a continuación:

1 Producto i con mayor tiempo de proceso, $i^* \in \arg\max\{p_{ik}\}$.
2 Producto i con menor tiempo de proceso, $i^* \in \arg\min\{p_{ik}\}$.
3 Producto i con tiempo de proceso más cercano a $\bar{p}$, $i^* \in \arg\min\{|p_{ik} - \bar{p}|\}$.
4 Producto i con tiempo de proceso mas cercano a c, $i^* \in \arg\min\{\Delta_{ik}\}$.
5 Producto i con mayor demanda pendiente , $i^* \in \arg\max\{d_i\}$.
6 Producto i con menor demanda pendiente, $i^* \in \arg\min\{d_i\}$.
7 Producto i con mayor índice dinámico $i^* \in \arg\max\{r_i\}$.
8 Producto i con menor índice dinámico r_{ik}, $i^* \in \arg\min\{r_i\}$.
9 Producto i que ocasiona mayor desplazamiento de todos los trabajadores, $i^* \in \arg\max\{\sum_{k=1}^{K} \Delta_{ik}\}$.
10 Producto i que ocasiona el menor desplazamiento de todos los trabajadores, $i^* \in \arg\min\{\sum_{k=1}^{K} \Delta_{ik}\}$.
11 Producto i con mayor tiempo de proceso en estación cuello de botella, $i^* \in \arg\max\{p_{ik^b}\}$.
12 Producto i con menor tiempo de proceso en estación cuello de botella, $i^* \in \arg\min\{p_{ik^b}\}$.
13 Producto i que de asignarse se acerca más al ideal de producción u_i, $i^* \in \arg\min\{|(n_i - d_i) - t \cdot u_i|\}$.
14 Producto i con la mayor suma del índi-

ce dinámico para todas las estaciones, $i^* \in \arg\max \sum_{k=1}^{K} r_{ik}$.
15 Producto i con la menor suma del índice dinámico para todas las estaciones, $i^* \in \arg\min \sum_{k=1}^{K} r_{ik}$.
16 Producto i que de asignarse, produce mayor sobrecarga en la etapa actual, $i^* \in \arg\max\{w_i\}$.
17 Producto i que de asignarse, produce el mayor tiempo muerto en al etapa actual, $i^* \in \arg\max\{o_i\}$.
18 Producto i que de asignarse, produce la menor sobrecarga en la etapa actual, $i^* \in \arg\min\{w_i\}$.
19 Producto i que de asignarse, produce el menor tiempo muerto en la etapa actual, $i^* \in \arg\min\{o_i\}$.
20 bajo el concepto de regularidad, producto i con sobrecarga mas cercana al ideal de sobrecarga (assuming $lbw > 0$), $i^* \in \arg\min\{(lbw/T)\cdot t - w_t - w_i\}$; donde w_t es el valor acumulado.