Procedimientos de secuenciación en líneas de montaje minimizando la sobrecarga

Joaquín Bautista[1], Jaime Cano[2]

Resumen—**Se consideran una variante del problema de secuenciación de unidades en líneas de montaje en contexto JIT. El criterio es la minimización de la sobrecarga. Se proponen algunas heurísticas que buscan el orden en que se introducirán los diferentes variantes de un mismo producto en una estación. Se toma de la literatura un procedimiento para varias estaciones para probar las heurísticas *greedy* propuestas.**

Palabras clave—**Líneas de montaje, reglas de prioridad, JIT.**

I. INTRODUCCIÓN

Las líneas de montaje de productos mixtos (como las de las ensambladoras de automóviles) son un medio que permiten producir grandes cantidades de un producto en diferentes versiones, sin necesidad de hacer grandes esfuerzos en cambios de modelos y permiten además reaccionar rápidamente a los cambios en las demandas de los clientes. Esta distinción entre versiones no sólo repercute en la apariencia de la variante, sino también en las cargas de trabajo generadas en las estaciones de la línea dedicadas a la incorporación de opciones especiales o valores de los atributos. Así, una vez que la línea ha sido equilibrada [1] y se ha obtenido un tiempo de ciclo, es necesario determinar el orden en que los diferentes modelos deben de entrar en la línea de montaje de acuerdo a cierto criterio. Dado que el tiempo de ciclo es un promedio ponderado de tiempos de procesamiento de todas las tareas necesarias para montar un producto, existirán tareas con alto y bajo contenido de trabajo, las cuales deben controlarse para evitar sobrecargas y tiempos muertos en la línea, es decir, deben encontrarse secuencias de unidades que regularicen las cargas de trabajo en todas las estaciones. En otras palabras, dado que cada tiempo de ciclo entra una nueva unidad en la línea o estación, si se introducen sucesivamente productos con tiempo de proceso mayor al ciclo, puede producirse un retraso en la terminación de alguno de estos productos. Lo cual puede producir paros de línea o trabajos perdidos o incompletos.

En este trabajo se asume que las estaciones son cerradas por ambos lados y el transportador que mueve los productos a través de las estaciones lo hace a velocidad constante. Básicamente, existen dos enfoques para este problema, uno de ellos lo toma [2] al minimizar el número de unidades ricas en contenido de trabajo que exceden a una cantidad k permitida en segmentos de secuencia de tamaño l, el otro enfoque, el cual adoptamos también, es el propuesto por Yano y Rachamadugu en [3], donde proponen un modelo de programación lineal para el problema, teniendo como objetivo maximizar el trabajo completado (equivalente a la minimización del trabajo perdido o sobrecarga). Los autores se centran en ejemplares con dos tipos de productos (básicos y con opción). Proponen un procedimiento que busca un esquema o patrón compuesto por ambos tipos de productos, de manera tal que la posición del trabajador dentro de la estación es la misma antes y después de haber procesado dicho esquema (el origen/inicio de la estación). Proponen también una heurística *greedy* para varias estaciones que usa el procedimiento de una sola estación para predecir cual producto puede ocasionar menor sobrecarga en cada etapa. Para ejemplares con K estaciones, la complejidad computacional de este procedimiento es de $O(KN)$. Bolat y Yano [4] extienden el trabajo anterior proponiendo tres métodos de resolución: (1) el primero determina un esquema *regenerativo* (la situación del operario al inicio y final del esquema es la misma) que se replica, para construir la solución; (2) el segundo es un algoritmo *greedy* que procura, paso a paso, evitar la trabajo perdido (sobrecarga) inevitable, combinando unidades; y (3) el tercer algoritmo, también *greedy*, intenta reducir el tiempo ocioso al evitar mientras es posible, asignar unidades con poca carga de trabajo. Tsai [5] extiende los trabajos anteriores al tener en consideración los tiempos de desplazamiento del operario, y establece dos objetivos: minimizar el desplazamiento máximo del trabajador a partir del origen de la estación y minimizar el trabajo no completado. El procedimiento propuesto tiene una complejidad computacional $O(logN)$ y ofrece la solución óptima en determinadas condiciones. Para el caso de más de una estación, se emplean los resultados del procedimiento anterior como cotas.

En el presente texto nos centraremos en las aportaciones de Yano y Rachamadugu [3], de Bolat y Yano [2,4] y Tsai [5], extendiéndonos sobre ellas. Nuestras propuestas contemplan procedimientos para varios productos y varias estaciones. Se

[1] UPC, Av. Diagonal 647, 08028. Barcelona, España. E-mail: joaquin.bautista@upc.edu
[2] UPC, Jordi Girona 1-3, 08034. Barcelona, España. E-mail: jaime.cano-belman@upc.edu

experimenta también con mejora local y se propone una hiper-heurística que manipula reglas de prioridad. Se realizan experiencias computacionales para evaluar los procedimientos.

II. MEDICIÓN DE LA SOBRECARGA

Yano y Rachamadugu [3] propone una forma general para medir el trabajo perdido considerando una sola estación. Los autores miden el trabajo perdido en unidades de tiempo. Como unidad de tiempo se tiene el tiempo de ciclo de fabricación c (tiempo entre llegadas de dos unidades consecutivas en la estación). Para describir la medición de la sobrecarga, asúmase que se tiene una estación. Sea L la longitud de la estación en unidades de tiempo, p_i el tiempo de proceso de la tarea sobre el producto i (i=1,..,I), s_t el instante de inicio de la tarea en la posición t (s_1=0; s_t=max(t-1,f_{i-1})), f_i el instante de finalización de la tarea en la posición t (f_t=min(s_t+p_i, t-1+L)), y wo_t el trabajo perdido ocasionado por la tarea en la posición t de la secuencia. Dada una secuencia de longitud T (t=1,…,T), y considerando sólo una estación, la sobrecarga obtenida en la posición t de la secuencia, se puede expresar así: $wo_t=[p_i+s_t-(t-1+L)]^+$. En definitiva, la sobrecarga total es $z=\sum_t wo_t$. Otras formulaciones de programación matemática del problema pueden encontrarse, entre otras, en [3] y [6]. Aunque el pareciera un problema de asignación es difícil de tratar debido a su falta propiedades estructurales. Las restricciones sobre las posiciones de los operarios y las variables para medir la sobrecarga (que son continuas si se asume c como unidad de tiempo) dificultan la resolución del problema. La mayor dificultad es el gran número de soluciones posibles y el esfuerzo computacional para evaluarlas. El problema es NP-*hard* [3], [5] o [6].

III. PROCEDIMIENTOS CONSTRUCTIVOS

Se proponen cuatro procedimientos inspirados en trabajos encontrados en la literatura [2], [3] y [4]. Las propuestas asumen múltiples productos y una sola estación. Dichos procedimientos se usan como predictores de sobrecarga en otro procedimiento para estaciones múltiples tomado de [3].

A. Procedimiento YR

El procedimiento original de Yano y Rachamadugu (YR), [3] considera una estación y dos tipos de productos: básicos (que notaremos B) y opcionales o especiales (que notaremos A). El procedimiento se basa en la repetición de una subsecuencia estable formados por m_a unidades A y m_b unidades B. Dado que el tiempo de proceso de una unidad especial p_a es mayor al tiempo de ciclo c, existe una cantidad máxima X de unidades A que pueden secuenciarse de forma consecutiva sin producir sobrecarga. Así, X es el máximo entero que satisface $X \le (L-c)/(p_a-c)$, y es también el máximo valor que puede tomar m_a.

De esta manera se tiene un esquema regenerativo que puede repetirse mientras exista producción pendiente para completar uno más. Para un valor dado de m_a la máxima utilización se alcanza cuando m_b toma el valor más pequeño que satisface $m_a \cdot p_a + m_b \cdot p_b = m_a + m_b$, donde $m_a \le X$. Así, asumiendo que existen valores enteros para m_a y m_b, la máxima utilización se alcanza resolviendo el modelo no lineal (1).

$$\begin{aligned} Max \quad & (p_a \cdot m_a + p_b \cdot m_b)/(m_a + m_b) \\ s.a. \quad & m_a \le X \\ & p_a \cdot m_a + p_b \cdot m_b \le m_a + m_b \\ & m_a, m_b \ge 0, enteros \end{aligned} \tag{1}$$

Sea n_i la cantidad a fabricar del producto i ($T=\sum n_i$), y nc el máximo número de ciclos que se pueden componer con m_a unidades A y m_b unidades B. La secuencia se construye de acuerdo a los pasos siguientes: 1) asignar los nc ciclos, 2) asignar $x_a=\min(n_a-nc\cdot m_a, m_a)$ productos A, 3) asignar $x_b= n_b - nc\cdot m_b$ productos B, y 4) $n_a-nc\cdot m_a-x_a$ productos A, si procede.

B. Extensión del procedimiento YR

La extensión del procedimiento de Yano y Rachamadugu (YRx) considera una sola estación y diferentes productos, opciones o atributos, distinguibles por su tiempo de proceso. Aunque un producto no demande ninguna opción extra en una estación se asume que requiere una cantidad mínima de trabajo sobre él.

De manera similar al procedimiento original, YRx busca esquemas compuestos por m_i unidades de tipo i (i=1,…I), que se repiten mientras se tienen unidades pendientes suficientes para completar un esquema más. Si no hay productos suficientes se busca un nuevo esquema. Sea I en número total de modelos diferentes, n_i la demanda inicial del modelo i, p_i el tiempo de procesamiento del producto de clase i, $\mathcal{A}$ el conjunto de modelos que requieren tiempo de proceso mayor al tiempo de ciclo, $\mathcal{B}$ el conjunto de productos que demandan tiempo de proceso en la estación menor al ciclo, y x_i la cantidad máxima de unidades de tipo i que pueden asignarse de forma consecutiva sin ocasionar sobrecarga ($i \in \mathcal{A}$) u ocio ($i \in \mathcal{B}$). La cantidad de cada modelo que conforma una secuencia se encuentra con el siguiente modelo:

$$\begin{aligned} Max \quad & \sum_{i=1}^{I} p_i \cdot m_i \Big/ \sum_{i=1}^{I} m_i \\ s.a. \quad & m_i \le \min\{x_i, d_i\} \\ & \sum_{i=1}^{I} p_i \cdot m_i \le \sum_{i=1}^{I} m_i \\ & m_i \ge 0, entero \end{aligned} \tag{2}$$

En el modelo (2) $x_i \le L-c/p_i-c$, $\forall i \in \mathcal{A}$, y $x_i \le L-c/c-p_i$, $\forall i \in \mathcal{B}$. d_i es la producción pendiente del modelo i después de repetir convenientemente la subsecuencia. Cuando $t=0$, $d_i=n_i$. Para acomodar los m_i productos dentro del ciclo, se consideran alternadamente unidades del conjunto $\mathcal{A}$ y luego del $\mathcal{B}$. El orden de incorporación de las unidades en la secuencia se hace de acuerdo al orden decreciente del índice $r_i=m_i \cdot |c-p_i|$ de cada producto. Así, del conjunto $\mathcal{A}$, se selecciona el producto con el mayor índice (sea j) y se asignan las m_j unidades. Después del grupo $\mathcal{B}$, se selecciona el producto con mayor r_i (sea l) y se asignan las m_l unidades del producto l. Y así sucesivamente. Si no se encuentra solución al modelo (2), se asigna en la etapa correspondiente de la secuencia una serie de productos del conjunto $\mathcal{A}$ de acuerdo al valor decreciente del índice r_i siempre y cuando no se incurra en sobrecarga. Posteriormente se asignan en la etapa correspondiente de la secuencia, unidades del conjunto $\mathcal{B}$, de acuerdo al valor decreciente del índice r_i, permitiendo, para regenerar completamente, incurrir en ocio sólo una vez. Para ser resuelto, el modelo (2) es transformado en un modelo lineal.

C. *Procedimientos greedy*

Se proponen tres diferentes variantes de un procedimiento que intenta favorecer el movimiento de los trabajadores en las estaciones hacia los límites superiores e inferiores alternativamente (*up-down*) como se ilustra en la figura 1. Las flechas representan tiempos de proceso, mientras que las líneas punteadas indican desplazamiento del trabajador hacia la siguiente unidad en la línea. Si una de las flechas largas se asignara después de la cuarta unidad en la figura 1, se produciría sobrecarga o trabajo perdido, por lo que se requiere la fase *down* para evitarla y para intentar regenerar el sistema.

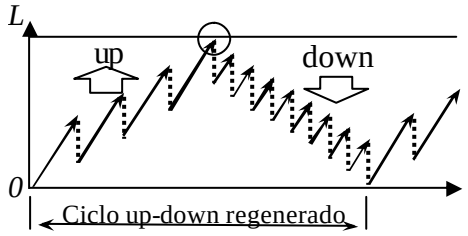

Fig. 1. Diagrama de desplazamiento up-down.

En cada etapa (*up* y *down*) se asigna el producto del grupo correspondiente ($\mathcal{A}$ o $\mathcal{B}$) con el mayor valor del índice dinámico $r_i=d_i \cdot |c-p_i|$. Asumiendo la posición inicial del trabajador en el límite inferior de la estación $s_1=0$, se sigue el siguiente procedimiento para completa la secuencia:

```
Mientras hay producción pendiente
   Aux=true;
  Mientras wi=0 y i Î A
   seq(t)¬ i|ri=max(ri);Aux=false;
  Mientras wi=0 y i Î B
   seq(t)¬ i|ri=max(ri);Aux=false;
  Si(Aux)
   seq(t)¬ i|ri=max(ri) y min{|wi|};
Fin
```

Algoritmo 1. up-down para varios productos y una estación

Donde $w_i(t)=max[s_t+p_i-L,0]$, $\forall\ i \in \mathcal{A}$, y $w_i(t)=min[s_t+p_i-c,0]$ $\forall\ i \in \mathcal{B}$.

Siguiendo el esquema del algoritmo *up-down*, se proponen tres procedimientos: Ud, UdC y UdR. El primero de ellos (Ud) permite a los trabajadores incurrir en tiempo inactivo para alcanzar la regeneración completa del sistema (que el trabajador vuelva al inicio de la estación), UdC no permite producir ni sobrecarga ni ocio mientras ello sea posible, UdR permite incurrir en cierta cantidad de ocio o en sobrecarga según la posición de la secuencia. Dicha cantidad permitida se determina por $lbw/T*t$. lbw es una cota inferior de sobrecarga, que para el caso de estaciones múltiples se determina como se indica en (3).

$$lbw = \sum_{k=1}^{K} \left[\sum_{i=1}^{I} n_i \cdot p_{ik} - \left(c \cdot (T-1) + L_k \right) \right]^{+} \qquad (3)$$

Donde $[x]^+$ es el máximo entre $(0,x)$. UdR intenta distribuir o regularizar la sobrecarga obligada a lo largo de la secuencia.

D. *Procedimiento para estaciones múltiples*

Se toma un procedimiento para estaciones múltiples de [3]. Este procedimiento se basa en procedimientos para una estación para encontrar cual es el producto con mejor predicción de sobrecarga. La secuencia se construye de forma progresiva de manera que en el instante t, el producto con producción pendiente y el mejor predictor de sobrecarga sea seleccionado y asignado en la *t-esima* etapa de la secuencia.

Sea p_{ik} el tiempo de proceso de la tarea sobre el producto i (i=1,…,I) en la estación k (k=1,…,K), wp_{ik} la predicción de sobrecarga ocasionada en la estación k, asumiendo que se ha asignado un producto de tipo i, $w_i(k,t)$ la sobrecarga ocasionada en la estación k en la etapa t de la secuencia si se asignase un producto de tipo i, s_{kt} el instante de inicio del trabajo en la estación k en el periodo t (o posición inicial del trabajador en la estación k en el periodo t), y $sc_i(t)$ es la predicción total de sobrecarga producida por el producto i en la etapa t. El procedimiento para estaciones múltiples se describe a continuación:

```
Para cada etapa de la secuencia
  Para cada producto
    suponer seq(t)¬ i;[d_i-1];
    calcular predictor sc_i(t);
    reestablecer d_i;
  Siguiente producto
  seq(t)¬ i*|sc_i*(t)=min{sc_i(t)};
Siguiente etapa
```

Algoritmo 2. procedimiento múltiples estaciones.

Donde wp_{ik} se obtiene con los procedimientos para una sola estación. La sobrecarga obtenida en el periodo t debido a la asignación del producto i se determina con (5) y (6).

$$sc_i(t) = \sum_{k=1}^{K} \left(wp_{ik} + w_i(k,t) \right) \cdot sc_k \quad (4)$$

$$w_i(k,t) = [s_{kt}+p_{ik} - ((t-1)^c)-L_k]^+ \quad \forall i \in \mathcal{A} \quad (5)$$

$$w_i(k,t) = [(t^c)-s_{kt}-p_{ik}]^+ \quad \forall i \in \mathcal{B} \quad (6)$$

IV. MEJORA LOCAL

Para mejorar las soluciones de los procedimientos constructivos se ha aplicado mejor local. Se usan dos tipos de vecindarios: intercambio de elementos e inserciones de segmentos.

E. Intercambios

Se hacen pruebas con intercambio de dos y de tres elementos de la secuencia. El intercambio de dos elementos (2S) produce una solución vecina. Sin embargo, el intercambio de tres elementos puede producir hasta 5 nuevas soluciones (figura 2). De esas cinco posibilidades, sólo en dos de ellas todos los tres elementos cambian de posición en la secuencia: (b,c,a) y (c,a,b).

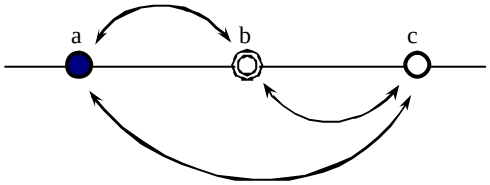

Fig. 2. Intercambio de 3 elementos.

Por ello, se diferencian dos vecindarios en el intercambio de tres elementos: 3S(a) y 3S(b). El primero toma en cuenta sólo las dos soluciones en las que cambian de lugar los tres elementos. El segundo, realiza las cinco alternativas. En la experiencia computacional realizada para medir la bondad de la mejora local, también se ha probado la aplicación de las cinco posibilidades después de haber llegado a un optimo local con 2S, lo cual notaremos 2-3S(b).

F. Inserción

Dado un tamaño dado de segmento *or* (1=*or*=*T*), la extracción de dicho segmento y su inserción en un posición diferente de la secuencia produce una solución vecina. En la experiencia computacional se hacen pruebas para 2=*or*=10.

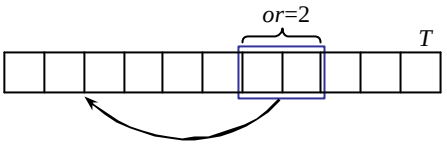

Fig. 3. Inserción de segmentos.

Como puede intuirse, el tamaño del vecindario con inserción de segmentos (*T*-*or*) es mucho menor que el tamaño del vecindario por intercambio de elementos. Las pruebas experimentales para la mejora local realizan hasta $\sqrt{T}$ iteraciones sin mejora.

V. REGLAS DE PRIORIDAD

En esta sección se describe una hiper-heurística (heurística de heurísticas). Aunque hay diferencias sustanciales, el procedimiento está inspirado en Scatter Search (SS). En lugar de usar soluciones para generar otras nuevas, la propuesta usa cadenas con reglas de prioridad para generar otras nuevas. El valor objetivo de una cadena de reglas le corresponde el valor solución de la secuencia equivalente a la cadena de reglas dada. Dicha secuencia solución se obtiene mediante el procedimiento constructivo de combinación de reglas (PCCR).

G. PCCR

PCCR es un procedimiento constructivo *greedy* basado en la combinación de reglas de prioridad. La asignación de cierto producto en una etapa de la secuencia depende de la regla de prioridad en dicha etapa. Un conjunto de reglas $R=\{r_1,r_2,r_3,\ldots,r_R\}$ determina el orden de los productos en la secuencia. La cadena (secuencia de reglas de prioridad) tiene la misma longitud que una secuencia solución (*T*). Así, la regla en la posición t de la cadena, determina de entre un conjunto disponible de productos, el que mejor satisface dicha regla.

Dada una cadena de tamaño *T*, la determinación del producto que mejor satisface la regla r en la posición t de la cadena, redetermina de la manera siguiente:

- Para cada producto candidato, calcular el calor de la aplicación de la regla *r*,
- Seleccionar el producto i con el mejor valor para la cadena r,
- Asignar i en la posición t de la secuencia,
- Actualizar la demanda de i.

H. Hiper-heurística

De forma similar a SS [7], la hiper-heurística propuesta (HH) crea nuevos elementos combinando los existentes. HH también opera sobre un conjunto

de referencia (*RefSet*), pero, en lugar de contener soluciones, contiene cadenas de reglas. En seguida se muestra el pseudo-código de HH:

```
Inicio
  Crear RefSet estático y dinámico;
  Iniciar matriz de frecuencias Fr;
Mientras Divs < MaxDivs
  Combinar cadenas del RefSet;
  Regenerar RefSet;
  Si RefSet no mejora
    Diversificar RefSet;
Fin mientras
```

Algoritmo 3. pseudo-código HH

Combinando esas cadenas se crean otras nuevas. En HH, el *RefSet* contiene tantas cadenas de reglas como reglas se consideran (*R*).El *RefSet* está compuesto por una parte estática (*RSs*) y una dinámica (*RSd*). El tamaño de cada sección del *RefSet* es de tamaño es igual a *R*. *RSs* se denomina estático porque no se modifica durante el proceso de búsqueda. En el *RSs*, la cadena de reglas 1 contiene sólo la regla 1, la cadena de reglas 2 contiene solo la regla 2, y así sucesivamente. De esta manera el proceso siempre toma en cuenta todas las reglas que considera el proceso para generar nuevas cadenas.

$$cr_r(r,r,r,...,r) \rightarrow RSs = \{cr_r : r \in R\} \quad (7)$$

El *RSd* cambia en cada iteración del proceso. Las cadenas de reglas iniciales se generan de forma aleatoria. El *RSd* contiene un conjunto *elite* de cadenas de reglas. *RSd* se actualiza en cada iteración del proceso tomando en cuenta las nuevas cadenas con mejores valores de sobrecarga que se han generado en la fase de combinación.

Como se dijo, el valor solución de una cadena de reglas se obtiene aplicando el PCCR. En este trabajo se han aplicado 20 reglas de prioridad. Si durante el proceso PCCR hay empate entre dos o más productos, se toman dichos productos y se aplican las reglas en orden numérico hasta desempatar.

Las reglas de prioridad usadas en este trabajo consideran diferentes criterios. Las reglas 1-4 usan priorizan de acuerdo los tiempos de proceso de los productos en las estaciones. Las reglas 5 y 6 seleccionan el producto con mayor y menor demanda pendiente, respectivamente. Las reglas 7, 8, 14 y 15 diferencian los productos relacionando su producción pendiente con la diferencia entre el tiempo de proceso y tiempo de ciclo. El desplazamiento de los trabajadores se considera en las reglas 9 y 10. La estación cuello de botella se contempla en las reglas 11 y 12. Las reglas 16 y 18 usan la sobrecarga para priorizar. Las reglas 17 y 19 usan el tiempo muerto. La regla 13 prioriza según la regularidad de la sobrecarga, mientras que la regla 20 prioriza regularizando el tiempo muerto.

En cada iteración del proceso, la matriz de frecuencia $Fr(r,t)$ es actualizada. Dicha matriz contiene el número de veces que la regla r aparece en la posición t de las cadenas de reglas del *RSd*. El proceso de combinación de cadenas del *Refset* se apoya en *Fr*, al igual que la Diversificación. Por cada par de cadenas ascendentes se obtiene una descendente.

La regla que ocupará la posición t de la cadena descendente se determina de acuerdo la frecuencia que la regla r tiene en la etapa t de la matriz *Fr*.

$$cr(t) = \begin{cases} cp(t) & \text{si } cp(t) = cq(t) \\ cq(t) & \text{si } Fr(cp(t),t) \geq Fr(cq(t),t) \\ cp(t) & \text{de otro modo} \end{cases}$$

Una vez que se han combinado todas las cadenas de reglas del *RefSet* se usa el PCCR para obtener el valor objetivo de las cadenas descendentes. EL *RefSet* es *regenerado* tomando en cuenta aquellas cadenas descendentes con mejores valores de sobrecarga. Se han usado tres estrategias de regeneración:

- El *RefSet* se actualiza con las mejores cadenas ascendentes y descendentes.
- Se actualiza todo el *RefSet* con las *R* mejores cadenas descendentes.
- Las peores αR cadenas del *RefSet* son sustituidas por las mejores αR descendentes.

En el proceso de regeneración, no hay duplicación de cadenas.

Cuando la regeneración deja de producir mejoras en el estado del *RefSet* se realiza la Diversificación. El proceso de Diversificación se realiza en dos fases: 1) Creación de cadenas de reglas diversificadas, y 2) Selección de las nuevas cadenas con mayor grado de diferenciación. La fase 1 se realiza con ayuda de la matriz de frecuencia *Fr*, pero el criterio de decisión ya no es el de tomar la regla con mayor frecuencia en *Fr*, sino aquella con menor frecuencia en *Fr*. Dados dos cadenas ascendentes, se obtiene una cadena descendente diversificada. La idea detrás del proceso es intuitiva: crear cadenas con reglas de prioridad que antes no se contemplaban en el *RefSet*, para intentar moverse a otro lugar en el espacio de búsqueda. En el segundo paso de la diversificación se toman las cadenas diversificadas con mayor grado de diferenciación respecto a las cadenas existentes en cada momento en el *RefSet*. El grado de diferenciación entre dos cadenas se mide con el número de coincidencias. Se da una coincidencia si en la posición t de las dos cadenas que se comparan existe la misma regla de prioridad r.

VI. EXPERIENCIA COMPUTACIONAL

Se realiza una experiencia computacional para probar la calidad de los procedimientos propuestos. Se usa la batería de 100 problemas diseñada en [6]. Todas las instancias asumen c=90 unidades de

tiempo. No se consideran costos por incurrir en sobrecarga o tiempo muerto en las estaciones. Para medir la calidad de las soluciones se usa el índice global utilizado por el autor de las instancias. A falta de buenas cotas inferiores de sobrecarga para comparar los procedimientos, en [6] se usa wo^*_h como la mejor solución encontrada para la instancia h (h=1,…,100). Dado que el valor de wo^*_h puede ser cero, se usa la desviación relativa global *rel.wo*.

$$rel.wo = \left(\sum_{h=1}^{100} wo_h - \sum_{h=1}^{100} wo_h^*\right) \Big/ \sum_{h=1}^{100} wo_h^* \cdot 100\% \qquad (8)$$

En la experiencia computacional se usan tres índices de desviación relativa global. El índice original (8) equivale a *rel.wo*2. En *rel.wo*1, el valor de wo^*_h equivale a la cota inferior de sobrecarga *lbw*. En rel.wo3, wo^*_h es la mejor solución encontrada considernado además la soluciones obtenidas con el software CPLEX 7.5 durante 15 minutos de búsqueda. Las pruebas se han realizado en una PC Pentium 4 CPU 2.4Ghz, 512 MB RAM en sistema operativo Windows XP profesional. En el procedimiento YRx también se uso CPLEX 7.5.

TABLA I
RESULTADOS GLOBALES POR PROCEDIMIENTO

Index	Ud	UdC	UdR	YRx
*rel.wo*1	45.64	41.82	40.97	69.25
*rel.wo*2	5.49	2.73	2.11	27.90
*rel.wo*3	6.00	3.22	2.59	28.51
#best	19	42	44	2
Cpu	3.99	4.96	4.56	4276

La tabla I contiene los tres índices de calidad para los cuatro procedimientos constructivos descritos en la sección III. Los mejores valores de los índices se obtienen con el procedimiento UdR, el cual reparte la sobrecarga determinada por lbw, a lo largo de la secuencia. Por otra parte, nuestra propuesta YRx obtiene los peores resultados. Cpu indica el tiempo promedio requerido (segundos) por cada procedimiento para obtener la solución. #best indica el número de veces que el procedimiento obtuvo mejor resultado que los demás.

Como ya se mencionó, se aplicó procedimientos de mejora local sobre los resultados obtenidos con los procedimientos constructivos. La tabla II contiene los valores para el índice *rel.wo*1 después de haber encontrado un óptimo local de cuerdo con cada tipo de vecindario (BL) y con cada procedimiento constructivo inicial.

Debido a que el tamaño del vecindario es mucho menor que en intercambio, en la inserción de segmentos el esfuerzo computacional también es mucho menor. La mejora local con intercambio fue limitada a 3600 segundos, excepto para el caso 2-3S(b), en el cual se permitió un máximo de 3600 segundos en la fase 2S y 1800 segundos en la fase 3S(b). En ninguna instancia de la batería se requirió más de medio minuto para encontrar un óptimo local con el vecindario por inserción de segmentos. En general, el índice *rel.wo*1 se reduce a la mitad con la mejora local. La mejor local también es mejor que la búsqueda hecha con CPLEX.

TABLA II
DESVIACIÓN RELATIVA GLOBAL 1, PARA BL

*rel.wo*1 (%)	Procedimiento Constructivo Inicial			
	Ud	UdC	UdR	YRx
BL	45.64	41.82	40.97	69.25
2S	24.32	24.30	24.53	24.54
3S(a)	35.16	33.03	34.84	55.39
3S(b)	32.99	31.86	30.55	44.15
2-3S(b)	23.96	23.88	24.09	24.06
2Ins	23.88	23.64	23.80	24.38
3Ins	23.51	23.35	23.34	24.22
4Ins	23.34	23.20	23.19	24.25
5Ins	23.15	23.00	23.00	24.02
6Ins	23.01	22.95	22.99	24.03
7Ins	22.98	22.87	22.77	23.91
8Ins	22.82	22.71	22.63	23.34
9Ins	22.72	22.61	22.60	23.35
10Ins	22.56	22.65	22.44	23.29

Una tercera experiencia computacional se realizó para HH, la cual, es independiente de las soluciones obtenidas con los procedimientos descritos en la sección III y IV. Como se muestra en la tabla III, la experiencia contempla tres procedimientos de regeneración y hasta un máximo de seis diversificaciones (columna izquierda).

TABLA III
RESULTADOS PARA HH

*rel.wo*1 (%)	Regeneration type		
Max Diversif.	Reg1	Reg2	Reg3
D3	42.81	42.79	42.54
D4	42.72	42.74	42.45
D5	42.64	42.64	42.47
D6	42.53	42.55	42.33

Los índices globales de HH no muestran una diferencia marcada entre cada tipo de regeneración. El número de diversificaciones en el proceso HH tampoco muestra signos de mejora sustancial. Las soluciones obtenidas con HH son similares en calidad a las obtenidas con los procedimientos constructivos o con CPLEX (en 15 minutos de búsqueda). En promedio, HH termina el proceso de búsqueda a los 156 segundos de iniciado el proceso.

VII. CONCLUSIONES

Se estudia una variante del problema de secuenciación de productos en líneas de ensamble. Se atiende al enfoque en el cual un producto

demanda un componente o tributo en cada estación de la línea, lo cual requiere un tiempo de procesamiento para su aplicación. El objetivo de los procedimientos propuestos es la minimización de la sobrecarga que se puede obtener debido al tiempo concedido en las estaciones y a las cargas de trabajo de una secuencia dada. Como en [3] y [6], se asume que ambos lados de la estación son cerrados y que el tiempo que requiere el trabajador para ir de una unidad a la siguiente es insignificante. Se han recopilado algunos procedimientos de la literatura y se proponen otros de tipo *greedy*. Los procedimientos propuestos consideran múltiples estaciones y múltiples productos.

La desviación relativa global con relación a la cota inferior de sobrecarga utilizada (lbw) indica resultados muy satisfactorios si se considera que la cota no es de calidad. Los procedimientos constructivos requieren, en promedio, solo unos segundos de tiempo de computo. Aun para las instancias más grandes de la batería, los tiempos son muy aceptables. Los mejores índices se obtienen con el procedimiento que reparte la sobrecarga a lo largo de la secuencia.

La hiper-heurística propuesta también obtiene buenos resultados, comparables con los obtenidos con los procedimientos constructivos. En función los parámetros de las instancias, HH mejora los resultados de los constructivos. En más de la mitad de las instancias, HH iguala o mejora el resultado obtenido con el software CPLEX en el tiempo de búsqueda estipulado.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el MEC del Gobierno Español a través del Proyecto DPI2004-03475. Agradecemos el apoyo dado por Nissan Spanish Industrial Operations, la Cátedra Nissan UPC y la CONACYT de México.

REFERENCIAS

[1] Monden Y. Toyota production System. Institute of Industrial Engineers Press, Norcross, GA. 1983.

[2] Bolat A., Yano C. A surrogate objective for utility work in paced assembly lines. Production planning and control, vol 3, nº4, 406-412, 1992b.

[3] Yano C., Rachamadugu R. Sequencing to minimize work overload in assembly lines with product options. Management science, vol 37, nº5, 572-568, 1991.

[4] Bolat A., Yano C. Scheduling algorithms to minimize utility work at a single station on paced assembly line. Production planning and control, vol 3, nº4, 393-405,1992a.

[5] Tsai L. Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage. Management science, vol 37, nº5,572-586, 1995.

[6] Scholl A., Klein R., Domschke W. Pattern based vocabulary building for effectively sequencing mixed-model assembly line. Journal of heuristics, 4,357-381, 1998.

[7] Laguna M., Martí R. Scatter search, methodology and implementations in C. Kluwer Academic Publishers, USA, 2003.