V Congreso de Ingeniería de Organización
Valladolid-Burgos, 4-5 Septiembre 2003

Índice

Determinación de la secuencia óptima para introducir productos en una línea de producción con transportadores. *

Manuel Mateo Doll[1], Svend Vanderveken [2]
[1] Departamento de Organización de Empresas (ETSEIB, Avda Diagonal, 647,7º, 08028 Barcelona, Universidad Politécnica de Cataluña, manel.mateo@upc.es)
[2] Faculté Polytechnique de Mons (B 7000 Mons, Belgium, svd_vdvk@hotmail.com)

RESUMEN

Los sistemas productivos con grúas o transportadores que realizan funciones de traslado de unidades de producto deben ser programados eficientemente. En este caso se aborda la producción de un conjunto de unidades, de productos diferentes, cada una de las cuales debe visitar una secuencia particular de baños. Se ha diseñado una heurística que ofrece soluciones factibles, aunque aborda el problema parcialmente formando ciclos de producción que contienen unidades de productos diferentes. Para reducir el tiempo total de producción según el algoritmo inicialmente propuesto, también se busca la mezcla de operaciones pertenecientes a ciclos consecutivos de grupos de piezas.

***Palabras clave**: programación, Hoist Scheduling Problem, heurística, ciclo.*

1. Introducción.

En ciertas líneas de producción, especialmente en la fabricación de placas de circuitos impresos, se usan las grúas para el transporte de productos entre estaciones de trabajo. Los ejemplos más extendidos son líneas de tratamiento químico, compuestas de una estación de entrada y de una estación de salida, una serie de tanques que contienen disoluciones químicas y grúas que transportan productos de un tanque a otro. En dichas líneas se aplica a los productos una cierta secuencia de tratamientos químicos o procesos de recubrimiento.

Las grúas realizan operaciones como extraer productos de los tanques, depositarlos en dichos tanques, transportarlos entre tanques, moverse en vacío entre dos de los tanques o estar paradas. Una grúa sólo puede transportar un producto a la vez.

Los productos pueden llegar de manera aleatoria, o no, a la estación de carga. Posteriormente, deben recorrer de manera secuencial algunos o todos los tanques de la línea (se permite saltar estaciones, según el tipo de producto final que se quiera obtener). Finalmente, los productos abandonan el sistema por la estación de descarga. Los tiempos de carga y descarga, así como los de viaje de grúa entre pares de tanques, son datos constantes.

La mayoría de las variantes para el problema de programación de grúa (en inglés, *Hoist Scheduling Problem*, HSP) han de respetar, como mínimo, dos condiciones específicas:

* Este trabajo se deriva del proyecto de investigación financiado por el MCYT con referencia DPI2001-2169, titulado “Secuenciación de unidades con restricciones múltiples”.

- Una vez extraído un producto de un tanque, inmediatamente se transporta al siguiente. Esto significa que no se permiten stocks entre estaciones.
- Cada producto cuenta para cada operación a realizar con una ventana temporal específica, o sea, por una duración mínima y una duración máxima. Si la permanencia de un producto en una estación no alcanza este valor mínimo o excede al máximo, el producto se considera defectuoso.

El *Hoist Scheduling Problem* (tratado en Yih [1] por ejemplo) consiste en determinar el uso óptimo de las grúas (cuándo y dónde recoger cada producto y a dónde llevarlo). El criterio de optimización es generalmente que la tasa de producción total sea máxima. Las restricciones específicas del problema implican que, una vez el producto entra en la línea, se tiene que asegurar que podrá salir en condiciones. Considerando todo esto, se lleva a cabo la programación de los movimientos de grúa. Se ha demostrado que el problema es NP-completo incluso en el caso de una grúa y un único tipo de producto en el sistema [2].

2. Descripción de la situación planteada.

El sistema productivo está constituido por una secuencia de baños ($1 \le j \le M$). *P* tipos diferentes de productos se fabrican en dicha línea de producción. Cada producto *i* requiere visitar una secuencia diferente de estaciones o baños, entre unos tiempos mínimo ($a_{i,tk}$) y máximo ($b_{i,tk}$), si se refieren al tanque *tk*. Se conoce el número de unidades n_i de cada producto *i* ($1 \le i \le P$).

El objetivo de la heurística propuesta es determinar una secuencia factible de movimientos de la grúa, con carga, tal que el tiempo para procesar todas las unidades en la línea sea el mínimo posible y se cumplan las restricciones de ventanas temporales. Posteriormente, la secuencia puede modificarse aplicando un algoritmo de *branch and bound* para alcanzar un menor tiempo total de proceso.

3. Notación.

La notación para el caso de diferentes tipos de piezas, cada uno de los cuales puede pasar por una secuencia de tanques distinta, es la siguiente:

P	número de tipos diferentes de productos
i	índice de tipos de producto (i=1,...,P)
n_i	número de unidades del producto *i* a producir
$N=\sum_{(i=1;P)} n_i$	número total de unidades a producir
M	número de tanques de la línea de producción
j	índice para la unidad *j*-ésima del producto *i* ($j=1,...,n_i$)
tk	índice de tanques de línea ($tk \in \{1,...,M\}$)
g_i	número de operaciones del producto *i*
$a_{i,tk}$; $b_{i,tk}$	tiempos mínimo y máximo de remojo del producto *i* en el tanque *tk* (i=1,...,P; $tk \in \{1,...,M\}$)
$f_{tk,tk'}$	tiempos de viaje de la grúa con y sin carga entre los baños *tk* y *tk'* ($tk,tk' \in [0,M+1]$)
$t_{i,j,tk}$	tiempo de inicio del movimiento de grúa en el tanque *tk*, de la *j*-ésima unidad del producto *i* ($i=1,...,P$; $j=1,...,n_i$; $tk=1,...,g_i$)

$m_{i,tk}$	*tk*-ésimo tanque visitado por una unidad del producto tipo *i*(i=1,...,P; tk=1,...,g_i)
P_c	número de unidades totales en un ciclo (de varios tipos de pieza)
U_c	número de productos en un ciclo o subciclo ($U_c \leq P_c$)
$tc_{i,tk}$	tiempo de inicio en un ciclo del movimiento sobre el tanque *tk* del producto *i*
uc_l	producto *l*-ésimo de la secuencia de productos en un ciclo

4. Los ciclos como base del procedimiento propuesto.

Para reducir el tiempo de cálculo, la heurística se basa en una programación mediante ciclos. Un ciclo es una secuencia de movimientos de grúa para procesar un conjunto de unidades desde la estación de carga hasta la de descarga. El número de unidades tratadas por ciclo será, en principio, igual a la cantidad de tipos de producto aún por secuenciar. La idea es calcular el ciclo y repetirlo tantas veces como convenga, lo cual implica que el estado de la línea (posición de las unidades en curso y de la grúa) debe ser el mismo al empezar cada ciclo.

Se utiliza una aproximación similar al algoritmo de Yih[1] para programación de operaciones: cada producto se programa lo más temprano posible. Como esto genera conflictos de tanque (cuando más de un producto coincide en la misma estación a la vez) y conflictos de grúa (si la grúa no dispone de suficiente tiempo para realizar movimientos consecutivos), se modifica la programación del último producto insertado para hacerla compatible con la de los demás. Los conflictos de tanque se resuelven retrasando el instante de entrada del último producto añadido. Los conflictos de grúa se resuelven usando la tolerancia (o ventana temporal) de uno de los productos involucrados en la infactibilidad. Si ninguna de las tolerancias conlleva una solución factible, se opta por retrasar el instante de entrada y así queda resuelto el conflicto.

5. Algoritmo para la obtención de un ciclo.

Sea P el número de tipos de productos diferentes y sea P_c el número de productos de un ciclo.

A grandes rasgos, el algoritmo presentado en [3] consiste en los siguientes pasos:

1. Calcular un ciclo con P_c tipos de productos diferentes (para el primer ciclo P_c=P):
   - Escoger el par de productos más productivos. Para determinar el más interesante, se utiliza la tasa de producción (1) de la secuencia.
   - Insertar el producto con máxima tasa de producción (1) al final del ciclo construido, hasta terminar todos los productos pendientes.

2. Repetir el ciclo tantas veces como se pueda (mientras quede por secuenciar una unidad de cada tipo involucrado en el ciclo en curso).

3. SI queda como mínimo un tipo de producto:
   SI sólo queda un único tipo de producto en la estación de carga:
   - Insertar cada unidad de ese tipo una a una.
   - Cada vez que se añade una unidad, resolver conflictos de tanque y grúa.

   SINO:
   - Actualizar P_c (P_c disminuye hasta el número de tipos de producto pendientes).
   - Actualizar la lista de tipos de producto, suprimiendo los finalizados.

- Volver al paso 1.

SINO (no queda ningún tipo de producto en la estación de carga):

- FIN.

Ejemplo

Sean cuatro productos diferentes a fabricar (p_1, p_2, p_3, p_4), de los cuales se deben producir un total de 4, 2, 3 y 2 unidades, respectivamente.

De entre las diversas parejas de productos, supongamos que la más productiva sea (p_4, p_2). A continuación, se examinaría las secuencias (p_4, p_2, p_1) y (p_4, p_2, p_3). Supongamos que ésta última resulta más productiva, con lo cual la posible secuencia con cuatro unidades, una de cada producto sería: (p_4, p_2, p_3, p_1). A final del paso 2, se dispone de la secuencia parcial con dos unidades de cada tipo: (p_4, p_2, p_3, p_1, p_4, p_2, p_3, p_1).

Se recalcula una secuencia con los productos restantes p_1 y p_3. Por ejemplo, se obtiene la secuencia (p_1, p_3), la cual se añade al final de la secuencia previa. Entonces quedan unidades de un solo producto, p_1. La secuencia final para que las unidades se introduzcan en la línea es (p_4, p_2, p_3, p_1, p_4, p_2, p_3, p_1, p_1, p_3, p_1). Ahora faltaría por determinar el orden de los movimientos de grúa y su temporización.

6. Descripción de algunos elementos del algoritmo.

6.1. Evaluación de secuencias o subsecuencias de piezas.

En el paso 1 del algoritmo deben compararse entre sí secuencias o subsecuencias de piezas: parejas de productos, tríos, cuartetos, etc. Para ello, se define la tasa de producción de un conjunto de U_c unidades como:

$$\tau = \frac{\sum_{u=1}^{U_C} \sum_{tk=1}^{g_i} a_{u,tk}}{\max_{\substack{u=1,\ldots,U_C \\ u_i \le n_i}} \left(t_{i,j,g_i} + f_{g_i,M+1}\right) - \min_{i=1,\ldots,P} \left(t_{i,1,0}\right)} \qquad (1)$$

En el numerador se suman las duraciones mínimas necesarias de todas las operaciones, o sea la duración útil, con lo que cualquier tiempo suplementario es una pérdida de tiempo. El denominador calcula el tiempo total para ejecutar todas las operaciones del programa. La secuencia más productiva es aquélla que tenga el máximo valor τ. Dicha definición puede servir también para comparar secuencias parciales, sin el mismo número de unidades.

6.2. Secuencia del primer par de productos.

Dado que los cálculos del resto del ciclo sólo son actualizaciones (añadiendo un producto al final de la secuencia), se trata por separado el procedimiento de generación del primer par de productos. Con P_c tipos diferentes de productos, se ha de examinar P_c*(P_c-1) posibles parejas.

Se empieza programando el primer producto del par seleccionado. Se considera que la grúa espera en cada tanque justo la duración mínima. Esto se realiza para los P_c productos solos. A continuación, se añade el segundo producto del par, justo después del primero. Se necesita determinar el instante mínimo de entrada y actualizar todos los tiempos de movimiento para disponer de un programa con dos productos sin conflictos (ni de grúa ni de tanque). Al final de esta primera etapa, se dispone de uc_i y $t_{i,1,k}$, para i=1,2 y $0 \le k \le g_i$.

6.3. Adición de una unidad en la última posición de un ciclo de productos.

Un ciclo se compone siempre de una unidad de cada uno de los tipos de productos restantes en la estación de carga. A principio del algoritmo, P_c se inicializa como P. Sea L el conjunto de P_c tipos de producto restantes (para el primer ciclo: L={1,...,P_c }).

Escoger el par de productos más productivos, según $\tau(1)$.
PARA np=3 hasta P_c:
  Inicializar $\tau^*=0$.
  PARA i=1 hasta P_c:
    SI $i \notin UC$; $i \in L$:
      - Añadir el producto tipo *i* en la np-ésima posición de ciclo (programar tan pronto como sea posible).
      - Resolver conflictos de tanque.
      - Resolver conflictos de grúa.
      - Calcular τ para el nuevo programa parcial.
      - SI $\tau > \tau^*$, asignar producto *i* a la última posición: $uc_{np}=i$; $\tau^*=\tau$.
  SIGUIENTE i
  Suprimir de la lista de tipos pendientes el producto uc_{np}.
SIGUIENTE np

Conocidos los valores de uc_i y $t_{i,1,k}$, para i=1,2 y $0 \le k \le g_i$, se añade el producto tipo *i* en la posición *np* del ciclo. El procedimiento para insertarlo se inspira en el algoritmo de Yih [1]:

1. Decidir el instante mínimo de entrada del nuevo producto *i* ($t_{i,1,0}$). Si el primer tanque de *i* es utilizado por alguno de los anteriores *np-1* productos, el producto *i* debe entrar una vez ese tanque quede libre.
2. Calcular el resto de valores $t_{i,1,tk}$ ($0 \le tk \le g_i$) sin considerar la presencia de otros productos: se supone que la grúa está plenamente dedicada al producto *i* y que éste permanece el tiempo mínimo ($a_{i,tk}$) en cada tanque.

6.4. Resolución de conflictos de tanque.

Un conflicto de tanque se produce cuando el tanque necesario para una operación del producto *i* está ocupado por alguno de los productos previos. Para resolverlo, es conveniente:

1. Comprobar en cuáles de los g_i tanques requeridos por el producto *i* existe un conflicto con otra unidad.
2. Para cada uno de los tanques, si existe un conflicto, calcular el mínimo retraso (*MR*): diferencia de tiempo entre los instantes en que el producto *i* deja el tanque *tk* según el programa $t_{i,1,tk}$ y en que sería preferible que lo hiciera para evitar el conflicto.

3. Quedarse con el máximo de dichos valores de *MR*.
4. Retrasar el instante de entrada del producto *i* en un tiempo *MR*, y repetir lo mismo para todas las operaciones de grúa requeridas por el producto *i*.
5. Al retrasar directamente el valor *MR* la entrada del producto *i*, se puede asegurar que se corrigen los conflictos hallados en el paso 1, aunque se pueden crear otros conflictos de tanque. Entonces, debe ejecutarse de nuevo los pasos 1 a 5, hasta que no quede conflicto alguno.

Aunque la grúa quede libre inmediatamente después un movimiento, se puede crear un conflicto de grúa por no considerar el tiempo necesario de desplazamiento del producto *i* entre los tanques *tk* y *tk+1* ($0 \le tk \le g_i$). Como respuesta, se plantean dos soluciones:

- "Solución con poco retraso" en el instante de entrada (*MR*): sólo resuelve conflictos de tanque y deja que la segunda parte del algoritmo trate los conflictos de grúa.
- "Solución con gran retraso" en el instante de entrada (*MR'*): intenta evitar o reducir algunos de los consecuentes conflictos de grúa.

Ambas versiones, *MR* y *MR'*, han sido implementadas. Los conflictos de grúa se solucionan usando tolerancias o retrasando el instante de entrada (si las tolerancias no sirven).

6.5. Resolución de conflictos de grúa.

Un conflicto de grúa sucede cuando la grúa no tiene suficiente tiempo para atender dos operaciones de transporte consecutivas. Supongamos que el primer movimiento de grúa (A) concierne al producto *i* y que el segundo (B) concierne al producto *i'* (figura 1).

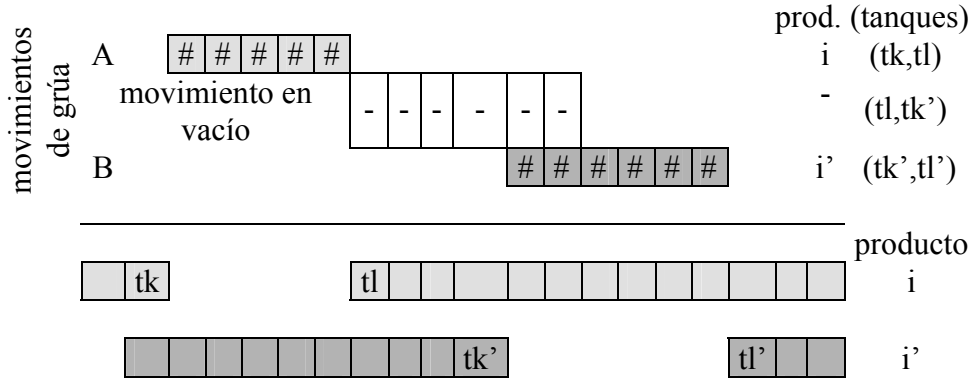

Figura 1: Conflicto de grúa entre los movimientos A y B.

1. Utilización de la tolerancia del movimiento B

Se procurará retrasar el movimiento de grúa B sin mover el movimiento A. Se comprueba si el retraso impuesto por A es menor o igual que la tolerancia del producto *i'* en el tanque *tk'* donde empieza la operación B. Si es cierto, se retrasa B y las siguientes operaciones de grúa del producto *i'*. No obstante, dicha tolerancia no se puede usar en dos situaciones:

(a) Si el tanque *tk'* es la estación de carga (tk=0), la solución sería retrasar la entrada

del producto al sistema productivo.
(b) Si el tanque destino del movimiento A es idéntico al tanque inicio del B (tl=tk').

2. Utilización de la tolerancia del movimiento A

Si no se puede usar la tolerancia de B, se busca realizar B antes que A. Esto significa ejecutar B según lo previsto y más tarde A, con el retraso impuesto por el cambio. Se producen dos excepciones similares a las detectadas al usar la tolerancia de B:

(a) Si el tanque *tk* es la estación de carga, se retrasa la entrada del producto.
(b) Si el tanque destino del movimiento B es el tanque inicio del A (tl'=tk).

3. Retraso de la entrada de un producto i

Si ninguna de ambas tolerancias puede ser útil, se retrasa el instante de entrada del producto *i* para resolver el conflicto.

6.6. Análisis de la relación entre ciclos de piezas repetidos.

Un ciclo se repite tantas veces como el número de unidades del producto con menor cantidad de unidades pendientes de secuenciar. La primera versión de la heurística simplemente coloca unos ciclos después de los otros sin modificarlos ("ciclos individuales"). Se ha realizado una segunda versión en que se trata de mezclar ciclos antes que la mera repetición. Por supuesto, la mezcla de ciclos permite reducir el tiempo total de proceso de la programación final [3].

La versión de los "ciclos mezclados" busca modificar el final de un ciclo para intentar colocar operaciones del inicio del siguiente más temprano que en la versión inicial. El algoritmo es similar al que añade un producto al final de la secuencia (se añade una secuencia al final de otra). La gran diferencia es la imposibilidad de utilizar tolerancias para resolver conflictos de grúa. Esto ocurre porque ambos ciclos son el mismo, y las modificaciones que se hagan a uno se deben realizar sobre el otro también.

7. Experiencia computacional.

Los ejemplares para probar la validez de los anteriores algoritmos quedan definidos por el número de tanques en la línea (M), la velocidad de movimiento de la grúa (para movimientos con carga o sin carga: $f_{tk,tk'}$, $e_{tk,tk'}$) y el número de tipos de producto (P). Para cada tipo de producto, se necesita conocer el número de unidades a producir (n_i), los tanques a visitar (g_i) y los tiempos mínimo ($a_{i,tk}$) y máximo ($b_{i,tk}$) de permanencia en cada una de las estaciones. Se ha trabajado con líneas de producción entre 5 y 10 estaciones y de 4 a 8 tipos de productos. Esto supone un total de 30 clases de problemas diferentes. Para cada clase, se ha creado 9 instancias diferentes, con lo cual finalmente se dispone de 270 ejemplares a resolver.

El número de estaciones a visitar por cada tipo de producto se basa en la tabla 1, redondeados aleatoriamente a los enteros más próximos. Ésta indica que el 20% de los tipos de producto en problemas de clase B han de visitar *M* tanques, 40% lo hacen en *M-1*, 20% en *M-2* tanques y el resto, en *M-3* tanques. De cada *pack* de 9 problemas, 3 pertenecen a la clase A, 3 a la clase B, y 3 de la clase C.

Número de tanques g_i	**Clases de problemas**		
	Clase A	**Clase B**	**Clase C**
M	20%	20%	40%
M –1	20%	40%	20%
M – 2	40%	20%	20%
M – 3	20%	20%	20%
	100%	**100%**	**100%**

Tabla 1: Clasificación de problemas según el número de operaciones por pieza.

El número de unidades de cada tipo se escoge de manera aleatoria entre 5 y 15, para que el total sea $40 \leq N \leq 120$. La duración mínima toma un valor aleatorio en el intervalo $a_{i,tk}=[20,80]$. Para cada *subpack* de 3 problemas, uno de ellos tiene duración máxima $b_{i,tk}$ en el intervalo $[a_{i,tk}+20;3\cdot a_{i,tk}]$, otro en $[a_{i,tk}+20;4\cdot a_{i,tk}]$, y el tercero en $[a_{i,tk}+20;3\cdot a_{i,tk}]$,

Los tiempos computacionales, ofrecidos en segundos, se han obtenido en un PentiumIII 400 Mhz con 128 Mb RAM. En la tabla 2 se recogen los valores medios para los tiempos de cálculo y τ de cada valor de *M* y de *P*, sea mediante ciclos individuales o ciclos mezclados:

	CICLOS INDIVIDUALES				CICLOS MEZCLADOS			
	Gran retraso (gr)		Poco retraso (pr)		Gran retraso (gr)		Poco retraso (pr)	
	τ	**t(s)**	**τ**	**t(s)**	**τ**	**t(s)**	**τ**	**t(s)**
M=5	0,83	0,33	0,82	0,24	0,84	0,34	0,83	0,35
M=6	0,80	0,58	0,79	0,22	0,80	0,59	0,79	0,62
M=7	0,77	0,80	0,76	0,61	0,77	0,85	0,79	0,62
M=8	0,75	1,13	0,74	0,81	0,75	1,14	0,75	1,22
M=9	0,73	1,30	0,73	0,95	0,73	1,30	0,73	1,34
M=10	0,74	1,57	0,73	1,24	0,74	1,57	0,74	1,66
P=4	0,75	0,22	0,74	0,14	0,75	0,23	0,75	0,23
P=5	0,83	0,48	0,82	0,35	0,83	0,49	0,83	0,52
P=6	0,79	0,77	0,78	0,56	0,79	0,79	0,78	0,78
P=7	0,77	1,28	0,76	0,94	0,77	1,29	0,77	1,28
P=8	0,78	1,92	0,77	1,50	0,78	1,92	0,78	1,93

Tabla 2: Tasa de producción τ y tiempo medio de cálculo (s), según nº de tanques M y tipos de productos P.

La ocupación de la línea es generalmente mejor con "gran retraso" que con "poco retraso", pero esto dista de ser una generalización. Sin embargo, si ambos tiempos de cálculo no son muy elevados, puede ser bueno probar ambas posibilidades cada vez. Los "ciclos mezclados" requieren como mínimo el tiempo de cálculo de "ciclos individuales", pues la mezcla de ciclos es una operación suplementaria. Los mejores resultados casi siempre se obtienen con "poco retraso" y "ciclos mezclados", pero con mayor tiempo de cálculo. Finalmente, se ha calculado los valores medios según la pertenencia de las instancias a cada clase (A, B o C), pero no se observa gran diferencia entre ellas.

4. Conclusiones.

En la experiencia computacional se ha trabajado con 270 instancias diferentes [3], clasificadas para probar las variantes de heurísticas propuestas. De ella, se desprende que las mejores programaciones son casi siempre con "ciclos mezclados", pero comporta mayor tiempo de cálculo. No se observan diferencias significativas entre los diferentes grupos de problemas: simplemente los resultados parecen un tanto mejores con los problemas más "duros".

La heurística, de hecho, encuentra rápidamente una primera solución factible del problema, con lo cual cumple su misión. Además, se considera que se ha mejorado el algoritmo de Yih [1]. No obstante, al tratar en los ciclos producto a producto se obtiene una secuencia parcial de unidades sin contemplar el conjunto de todas las secuencias posibles.

Referencias

[1] Yih, Y. (1994). An algorithm for hoist scheduling problems, *International Journal of Production Research*, vol. 32, 3, pp. 501-516.

[2] Lei L.; Wang, T.-J. (1989). A proof: the cyclic hoist scheduling problem is NP-complete, *Working paper #89-0016*, Rutgers University.

[3] Vanderveken, S.; Mateo, M. (2002). "Hoist Scheduling Problem. Determination of the optimal sequence of products' introduction into a production line. Heuristic", *Documento Interno de Trabajo D.I.T. 2002/07*, Departamento de Organización de Empresas. Universitat Politècnica de Catalunya.