Hiperheurística para un problema de equilibrado de líneas de montaje usando Scatter Search

Joaquín Bautista[1], Elena Fernández[1], José Luis González Velarde[2], Manuel Laguna[3]

[1] Universitat Politècnica de Catalunya, Avda. Diagonal 647, 7ª Planta 08028 Barcelona
[2] Instituto Tecnológico de Monterrey, Garza Sada Sur 2501, 64849, Monterrey, N.L:, Mx.
[3] Leeds School of Business, University of Colorado, Boulder, CO 80309-0419
{joaquin.bautista, e.fernandez}@upc.edu, gonzalez.velarde@itesm.mx, manuel.laguna@colorado.edu

Resumen

El presente trabajo se centra en la aplicación de un procedimiento basado en *Scatter Search* (SS) para resolver un problema de equilibrado de líneas de montaje. Tras una introducción a los denominados *Assembly Line Balancing Problems* (ALBPs) se propone un modelo básico para su variante simple (SALBP). Tras ello, se presentan las heurísticas greedy (basadas en reglas de prioridad) empleadas para resolver SALBP, se plantea su hibridación y se propone un procedimiento de combinación de soluciones, representadas por secuencias de reglas heurísticas de prioridad en la asignación de tareas, bajo un esquema SS. Finalmente, se realiza una experiencia computacional, con instancias de referencia, para probar el procedimiento, y se establecen las conclusiones del trabajo.

1. Introducción

Las líneas de producción y de montaje se caracterizan por estar constituidas por un conjunto de puestos de trabajo denominados estaciones. Las estaciones (m, indexadas en un conjunto K) se disponen en serie y en paralelo y a través de ellas fluye la obra en curso de un producto (motores, carrocerías, bastidores, etc.) gracias a un sistema de transporte encargado, por una parte, de aportar, los materiales requeridos al flujo principal y, por otra, mover las unidades del producto de una estación a la siguiente. Las unidades del producto pueden ser idénticas o, más usual actualmente, presentar distintas variantes.

La fabricación de una unidad de producto se suele dividir en un conjunto V de tareas n indexadas, a su vez, en un conjunto J (el cual depende de la lista de materiales y de las especificaciones de montaje suministradas por ingeniería de producción). Al subconjuntos de tareas asignadas a una estación concreta k ($1 \le k \le m$) se denomina *carga* de la estación k y la representaremos mediante S_k ($S_k \subseteq V$) y se impone como condición que cada tarea debe asignarse a una única estación. Una tarea concreta j ($1 \le j \le n$) requiere para su ejecución un tiempo de operación $t_j > 0$ que depende de las tecnologías de fabricación y de los recursos empleados.

Adicionalmente, la naturaleza del producto y, de nuevo, las tecnologías de fabricación implican que para cada tarea j exista un conjunto de tareas precedentes inmediatas, P_j, que deben concluirse antes de iniciarse la tarea j. Normalmente, estas restricciones suelen representarse mediante un grafo acíclico cuyos vértices se asocian a las tareas y cada arco dirigido (i,j) representa que la tarea i debe finalizarse antes de que se inicie la tarea j, por tanto, si $i \in S_h$ y $j \in S_k$, entonces debe cumplirse $h \le k$.

El tiempo de carga de una estación k, $t(S_k)$, es la suma de los tiempos de operación de las tareas asignadas a dicha estación. El tiempo de ciclo, c, es el tiempo concedido a cada estación para llevar a cabo las tareas que tienen asignadas. Nótese, por tanto, que el tiempo de ciclo no puede ser menor que el mayor tiempo de carga entre todas las estaciones, ni es lógico que sea mayor que la suma de los tiempos de operación de todas las tareas, esto es: $\max_k\{t(S_k)\} \le c \le \Sigma_k t(S_k) = t_{sum}$.

Por otra parte, cada estación k presenta un tiempo muerto $I_k = c - t(S_k)$, mayor o igual a 0, y la suma de dichos tiempos parciales da lugar al tiempo muerto total, $I_{sum} = \Sigma_k I_k = m \cdot c - t_{sum}$, que está vinculado a la ineficiencia de la línea.

En general, los problemas de equilibrado de líneas de montaje (ALBPs) se centran en agrupar

Actas del IV Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB2005, pp.839-845
ISBN: Vol I:84-9732-440-4 Vol II: 84-9732-467-6

por estaciones de trabajo las tareas del conjunto *V*. Resumiendo, el objetivo es conseguir una agrupación de tareas que minimice la ineficiencia de la línea o su tiempo muerto total y que respete todas las restricciones impuestas a las tareas y a las estaciones.

La primera familia de problemas, denominada SALBP (*Simple Assembly Line Balancing*), [3], puede enunciarse como sigue: dados un conjunto de tareas, con sus tiempos de operación y un grafo de precedencias asociado, cada tarea se debe asignar a una única estación de manera que se satisfagan todas las restricciones de precedencia y que no haya ninguna estación cuyo tiempo de carga, $t(S_k)$, exceda el tiempo de ciclo *c*.

La familia SALBP presenta cuatro variantes: SALBP-1: minimizar el número de estaciones *m* dado un valor del tiempo de ciclo *c*; SALBP-2 minimizar el tiempo de ciclo *c* dado un número de estaciones *m*; SALBP-E: minimizar, a la vez, *c* y *m* considerando su relación con el tiempo muerto total o la ineficiencia de la línea; SALBP-F: dados *m* y *c* determinar la factibilidad del problema, y en caso afirmativo hallar una solución del mismo.

Cuando se añaden consideraciones adicionales a las de la familia SALBP, los problemas se conocen en la literatura como GALBPs (*General Assembly Line Balancing Problems*). Esta familia incluye, entre otros, problemas que consideran estaciones de trabajo en paralelo, [5], grupos forzados de tareas, [6], e incompatibilidades entre tareas, [1]. En [16] puede encontrarse un análisis y bibliografía actualizados, así como el estado del arte de diferentes procedimientos para SALBP's.

En cuanto a procedimientos de resolución, se dispone de un primer grupo de algoritmos greedy que emplean reglas de prioridad para asignar las tareas a las estaciones [8] [17]; un segundo grupo compuesto por procedimientos bajo un esquema *branch and bound*, [9] [10] [13] [15], que son los más eficaces actualmente; y un tercer grupo compuesto por metaheurísticas diversas [14] [2]. Casi todos ellos están orientados a resolver problemas SALBP-1 o SALBP-2.

El presente trabajo se ha organizado así: en la sección 2 se presenta un modelo matemático para SALBP; la sección 3 se centra en las heurísticas básicas para el problema y en la 4 ilustramos, a través de un ejemplo, algunas de sus deficiencias; en la sección 5 se propone una representación de las soluciones mediante cadenas de reglas y en la 6 se propone un método para combinarlas usando Scatter Search; tras ello, la sección 7 se dedica a probar el procedimiento propuesto con instancias de referencia SALBP; finalmente, la sección 8 recoge las conclusiones del trabajo.

2. Un modelo matemático para el SALBP

Para modelar SALBP, se emplea la notación adicional siguiente:

- E_j, L_j Índices de la primera y última estación, respectivamente, a la que puede asignarse la tarea *j*.
- UB Cota superior del número de estaciones.
- x_{jk} Variable de decisión binaria que toma el valor 1 si la tarea $j \in J$ se asigna a la estación $k \in K$, y (0 en otro caso).

Con la notación anterior las siguientes restricciones definen soluciones factibles para SALBP-F.

$$\sum_{k=E_j}^{L_j} x_{jk} = 1 \qquad j \in J \tag{1}$$

$$\sum_{k=1}^{UB} \max_{1 \le j \le n} \{x_{jk}\} \le m \tag{2}$$

$$\sum_{j=1}^{n} t_j x_{jk} \le c \qquad k \in K \tag{3}$$

$$\sum_{k=E_i}^{L_i} k x_{ik} \le \sum_{k=E_j}^{L_j} k x_{jk} \qquad (i, j \in J) \wedge (i \in P_j) \tag{4}$$

$$x_{ik} \in \{0,1\} \qquad i \in J, k \in K \tag{5}$$

Las igualdades (1) aseguran que cada tarea se asigna a una única estación, mientras que las desigualdades (2) y (3) aseguran que se utilizan como máximo *m* estaciones de trabajo y que el tiempo de carga en cada estación no excede el tiempo de ciclo, respectivamente; por su parte, las desigualdades (4) sirven para garantizar el cumplimiento de las relaciones de precedencia entre tareas; finalmente, las expresiones (5) sirven para definir las variables de decisión como binarias.

Como hemos mencionado anteriormente, en SALBP-F, los elementos *m* y *c* son valores fijos, mientras que, en el resto de problemas, uno o más de esos elementos actúan como función objetivo. En particular cuando se consideran SALBP-1 y SALBP-2, las expresiones (6) y (7) corresponden a sus respectivas funciones objetivo:

$$\min imize\ Z_1(x) = m = \sum_{k=1}^{UB} \max_{1 \le j \le n} \{x_{jk}\} \tag{6}$$

$$\min imize\ Z_2(x) = c = \max_{1 \le k \le UB} \left\{ \sum_{j=1}^{n} t_j x_{jk} \right\} \tag{7}$$

La función objetivo de SALBP-E es minimizar $m \cdot c$, siendo la solución óptima del problema obvia

(m=1, c=t_{sum}.) si sólo se considera (1) a (5) como restricciones del problema; por ello, se impone al problema un rango de valores posibles para m, esto es: $m_{min} \leq m \leq m_{max}$.

3. Heurísticas básicas greedy y GRASP

En la figura 1 se presenta el esquema de un procedimiento heurístico para generar una solución del problema SALBP.

0. Inicialización
$k = 1$, $NA = V$, $WTP_i = |TP_i|$, $r = c$

1. Verificar factibilidad
{precedencia}
$\forall i \in NA$, if $WTP_i = 0$, $B = B \cup \{i\}$
{tiempo}
$\forall i \in B$, if $t_i \leq r$, $F = F \cup \{i\}$

2. Asignación
if $F = \emptyset$, {abrir nueva estación}
  $I = I+r$, $r = c$
  Paso 1
else {seleccionar tarea de F}
  i^* = índice tarea seleccionada
  NA, B, $F = NA$, B, $F - \{i^*\}$
  $r = r - t_{i^*}$
  $\forall i \in TS_{i^*}$, $WTP_i = WTP_i - 1$
  if $NA \neq \emptyset$, Paso 1

3. $I = I+r$, Alto

Figura 1: Esquema del procedimiento de obtención de una solución para el problema SALBP.

El esquema es válido tanto para las heurísticas *greedy* deterministas como para GRASP.

Las heurísticas greedy orientadas a estaciones efectúan la asignación de las tareas, una a una, en función de una regla de prioridad que distingue al algoritmo. El proceso de selección se realiza entre un conjunto de tareas compatibles con la parte de la solución ya construida (denominado conjunto de candidatas), las cuales satisfacen todas las restricciones de precedencia y tienen un tiempo de operación menor o igual que el tiempo disponible en la estación abierta. Cuando se obtiene una solución se puede aplicar o no un procedimiento de optimización local.

En los algoritmos GRASP, se puede limitar el número de candidatas en cada iteración y además en el proceso de asignación se sortean las tareas candidatas en función de una probabilidad de selección que puede depender del valor del índice de aquéllas; en cualquier caso a la solución hallada en la primera fase se aplica un procedimiento de optimización local [12].

Como es lógico, para definir una heurística es necesario fijar al menos una regla de prioridad en el proceso de selección. En el Anexo I se presenta una muestra compuesta por 15 reglas de prioridad elementales.

4. Un ejemplo ilustrativo

En la figura 2 se muestra un conjunto formado por 12 componentes que deben ser montados en una línea a la que se concede un tiempo de ciclo de 1 minuto. En la tabla 1 se resume las duraciones (en segundos) y ligaduras de precedencia inmediata entre tareas.

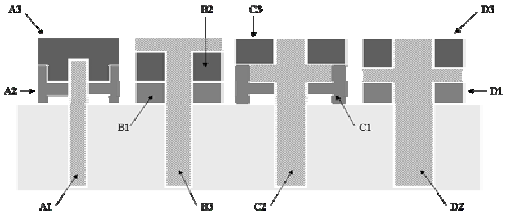

Figura 2: Ensamblado de conjunto en línea.

El ejemplo propuesto se ha resuelto mediante el procedimiento greedy presentado en la sección 3 empleando, en el paso correspondiente a la asignación de tarea en la estación abierta, las reglas de prioridad 1 a 14 que se relacionan en el anexo 1.

j	t_j	P_j	j	t_j	P_j
A1	6	-	C1	11	-
A2	7	A1	C2	10	C1
A3	6	A2	C3	10	C2
B1	8	-	D1	15	-
B2	8	B1	D2	17	D1
B3	8	B2	D3	14	D2

Tabla 1. Datos del ejemplo ilustrativo: tarea, tiempo de proceso (en segundos) y tareas precedentes inmediatas.

Cualquiera de las 14 soluciones obtenidas, empleando individualmente las 14 reglas (1 a 14) deterministas mencionadas y sin recurrir a una optimización local posterior, requiere 3 estaciones de trabajo. No obstante, es fácil hallar a simple vista soluciones óptimas para este ejemplar que requieren sólo dos estaciones de trabajo; por ejemplo: una primera estación E1={D1, D2, D3, A1, B1} y una segunda estación E2={C1, C2, C3, A2, A3, B2, B3}.

Dicha solución se puede obtener empleando la regla de prioridad 1 (Mayor tiempo de operación) en las tres primeras decisiones de asignación de tarea, la regla 14 (Menor tiempo de operación) en la cuarta decisión, de nuevo la regla 1 en la quinta decisión y cualquiera de las 15 reglas en las decisiones restantes hasta completar el proceso de asignación de las 12 tareas. Esta cadena de 12 decisiones se puede representar mediante el vector de reglas (1,1,1,14,1,*,*,*,*,*,*,*).

Con este sencillo ejemplar no se pretende cuestionar la calidad de las reglas, sino la forma de aplicarlas y poner en evidencia la rigidez de las heurísticas greedy "puras" incluso cuando la regla de prioridad empleada sea muy refinada.

5. Representación de soluciones mediante reglas de prioridad.

Los procedimientos constructivos basados en reglas de prioridad calculan para las distintas tareas valores asociados a la regla de prioridad elegida, que están basados en los tiempos de ejecución de las tareas y en las relaciones de precedencia dadas. En cada iteración se asigna a la primera estación de trabajo disponible aquella tarea todavía no asignada con el mejor valor respecto a la regla de prioridad elegida.

Los procedimientos constructivos clásicos para SALPB utilizan la misma regla a lo largo de todo el proceso de asignación.

En este trabajo proponemos un procedimiento constructivo que combina distintas reglas y que supone una generalización de los procedimientos clásicos. En particular, en las diferentes iteraciones, pueden utilizarse distintas reglas. Denotemos por $\mathfrak{R} = \{R^1, R^2, ..., R^k\}$ el conjunto de reglas de prioridad. Un esquema de un Procedimiento Constructivo de Combinación de Reglas (PCCR) es el siguiente:

Initialización: $NA=J$ (conjunto de tareas no asignadas).
Para ($i = 1, \ldots, n$) *hacer*:
  Seleccionar $r(i) \in K$.
  Sea $j_{r(i)} \in NA$ el resultado de aplicar la regla de prioridad $R^{r(i)} \in \mathfrak{R}$ al conjunto de tareas no asignadas NA.
  Actualizar $NA := NA \setminus \{j_{r(i)}\}$.
Finpara

Nótese que el índice de tarea asignado en la iteración i, $j_{r(i)}$, depende de la regla de prioridad $R^{r(i)}$ así como de los índices que tareas que ya han sido asignadas. En el contexto del esquema anterior, el (PCCR) que aplica la regla de prioridad $R^{r(i)}$ en la iteración $i \in J$ se representa mediante el vector de índices de reglas $\boldsymbol{r} = (r(1), r(2), \ldots, r(n))$. La solución obtenida puede determinarse a partir del vector $\boldsymbol{r}$, y se denotará por $\boldsymbol{s} = (j_{r(1)}, j_{r(2)}, \ldots, j_{r(n)})$. Su valor se denota por $m(\boldsymbol{s})$. Nótese que cuando para algún $i \in J$ la regla de prioridad $R^{r(i)}$ es no determinista, el resultado $j_{r(i)}$ puede no ser único. El conjunto de todas las posibles soluciones obtenidas mediante el procedimiento constructivo $\boldsymbol{r} = (r(1), r(2), \ldots, r(n))$ se llama Conjunto Solución de $\boldsymbol{r}$, y se denota $Sol(\boldsymbol{r})$.

6. Usando Scatter Search

El esquema anterior puede integrarse en una meta-estrategia en la que nuevas PCCRs se generan mediante combinaciones de reglas existentes. El hecho de que una regla, digamos R^i, aparezca varias veces en PCCR se puede interpretar como asignar una mayor ponderación a la regla R^i. Por tanto, esta metodología sigue la misma filosofía que la de generar combinaciones numéricamente ponderadas de reglas ya existentes como en Scatter Search (SS) (ver [7] [11]).

Las estrategias antecesoras para combinar reglas de decisión fueron ya propuestan hace más de 40 años en el contexto de secuenciación de tareas job shop scheduling, [4]. Como señaló Glover, [7], el enfoque está motivado por la suposición de que la información sobre el relativo interés de las posibles alternativas se captura de manera distinta por las diferentes reglas, y de que esa información puede aprovecharse de manera más efectiva cuando se integra mediante un mecanismo de combinación que cuando se trata mediante una estrategia estándar seleccionando las

distintas reglas de una en una, de manera aislada unas de otras. Más recientemente, esta filosofía se ha generalizado y formalizado en SS, (ver, p.ej. [11]). A diferencia de otros métodos evolutivos, SS se basa en la premisa de que métodos y diseños sistemáticos para crear nuevas soluciones implican beneficios significativos sobre los derivados del recurso a la aleatoriedad.

A continuación proponemos un algoritmo de SS para generar las combinaciones ponderadas que definen las distintas PCCP's. Las componentes de nuestro algoritmo son:

- Conjunto de Referencia que consiste en un conjunto de elementos, cada uno de los cuales está asociado a un PCCR diferente. Los $\boldsymbol{r}$ del Conjunto de Referencia se evalúan a través de sus conjuntos soluciones. Es decir, para un PCCR dado $\boldsymbol{r}$ del Conjunto de Referencia, seleccionamos un elemento de su Conjunto Solución $\boldsymbol{s} \in \text{Sol}\backslash\{\boldsymbol{r}\}$ y le asignamos un valor $val(\boldsymbol{r})=m(\boldsymbol{s})$. Inicialmente, el Conjunto de Referencia está formado por todos los PCCR's que utilizan la misma regla en cada iteración. Es decir, si $\boldsymbol{r^i} = (i, i, \ldots, i)$ denota el PCCR que utiliza la regla R^i en cada iteración, el Conjunto de Referencia inicial es $RS = \{\boldsymbol{r^i} : i \in K\}$.
- Método de Combinación de Soluciones. Utilizamos una matriz de frecuencias cuyos elementos $Fr(i,j)$ son el número de PCCR's del conjunto de referencia (contados desde el principio del proceso) que utilizan la regla R^i en el paso j. Sean $\boldsymbol{p}=(p(1), p(2), \ldots, p(n))$ y $\boldsymbol{q}=(q(1), q(2), \ldots, q(n))$ dos PCCR's del Conjunto de Referencia que deseamos combinar. Las componentes de la solución combinada son las siguientes:

$$r(j) = \begin{cases} p(j) & \text{si} \quad p(j) = q(j); \\ p(j) & \text{si} \quad Fr(p(j), j) \geq Fr(q(j), j); \\ q(j) & \text{en otro caso.} \end{cases}$$

- Método de actualización del Conjunto de Referencia.
- Método de generación de subconjunto del conjunto de referencia sobre el cual operar, para producir un subconjunto de sus soluciones como base para crear soluciones combinadas.

7. Experiencia computacional

Se eligieron tres familias de problemas SALBP (disponibles en http://www.wiwi.uni-jena.de/Entscheidung/alb/_albdata.htm) Barthold Scholl, Barthol2, aun cuando en este lugar se albergan muchas más familias, se seleccionaron éstas por considerar que son las que presentan el mayor reto. Barthold consta de 8 problemas, con 148 tareas, cuyos tiempos de procesamiento se encuentran en un rango entre 1 y 83 unidades de tiempo. Los tiempos de ciclo considerados varían entre 403 y 805. Ninguno de estos problemas admite una solución óptima que sea producida usando una sola de las reglas enunciadas, al combinar las diferentes reglas mediante el método de SS descrito, se encontró el óptimo para siete de los ocho casos, los resultados se encuentran en la Tabla 2. Donde TC significa Tiempo de Ciclo.

TC	Puras	SS	OPT	TC	Puras	SS	OPT
403	15	14	14	564	11	10	10
434	14	13	13	626	10	10	9
470	13	12	12	705	9	8	8
513	12	11	11	805	8	7	7

Tabla 2

La familia Scholl consta de 26 problemas, de 297 tareas, con tiempos de procesamiento entre 5 y 1386, con tiempos de ciclo variando desde 1394 hasta 2787 unidades de tiempo. De estos ejemplares 2 de ellos pueden ser resueltos óptimamente mediante alguna regla pura, a los restantes, se les aplicó el método de combinaciones SS, y se logró reducir el número de estaciones usadas en dos casos, alcanzando el óptimo en uno de ellos. Los resultados se muestran en la Tabla 3. Por último la familia Barthol2 formada por 27 problemas de 148 tareas, tiempos de procesamiento entre 1 y 83 y tiempos de ciclo desde 84 hasta 170. Ninguno de ellos es resuelto usando sólo una regla, en 5 ejemplares fue posible reducir el número de estaciones usadas, pero en ningún caso se consiguió alcanzar el óptimo, esta familia resultó ser la más difícil en términos del método propuesto, la Tabla 4 contiene estos resultados.

TC	Puras	SS	OPT	TC	Puras	SS	OPT
1394	52	51	50	1883	38	38	37
1422	51	50	50	1935	37	37	36
1452	49	49	48	1991	36	36	35
1483	48	48	47	2049	35	35	34
1515	47	47	46	2111	34	34	33
1548	46	46	46	2177	33	33	32
1584	45	45	44	2247	32	32	31
1620	44	44	44	2322	31	31	30
1659	43	43	42	2402	30	30	29
1699	42	42	[41,42]	2488	29	29	28
1742	41	41	40	2580	28	28	27
1787	40	40	39	2680	27	27	26
1834	39	39	38	2787	26	26	25

Tabla 3

TC	Puras	SS	OPT	TC	Puras	SS	OPT
84	52	52	51	115	38	38	37
85	52	51	50	118	37	37	36
87	51	50	49	121	36	36	35
89	50	49	48	125	35	35	34
91	48	48	47	129	34	34	33
93	47	47	46	133	33	33	32
95	47	46	45	137	32	32	31
97	45	45	44	142	31	31	30
99	44	44	43	146	30	30	29
101	43	43	42	152	29	29	28
104	42	42	41	157	28	28	27
106	42	41	40	163	27	27	26
109	40	40	39	170	26	26	25
112	39	39	38				

Tabla 4

8. Conclusiones

Para el problema SALBP se han propuesto diferentes reglas heurísticas, desafortunadamente para una gran cantidad de ejemplares de este problema, emplear una sola de estas reglas no conduce a la solución óptima, por lo cual se propone usar una mezcla de estas reglas. Al aplicar el método propuesto a un conjunto de ejemplares se pudieron clasificar en tres posibles subconjuntos: aquellos problemas que pueden ser resueltos con una sola regla, éstos fueron los menos. Problemas que no pueden ser resueltos con una sola regla, pero que al aplicar el método SS para combinar estas reglas, se consiguen mejores soluciones, algunas de ellas óptimas. Y un tercer grupo que son aquellos problemas para los cuales, el método de combinaciones de reglas no mejora los resultados obtenidos usando una sola regla. A partir de esta conclusión es plausible pensar que en vez de aplicar diferentes reglas, invariantes, en diferentes puntos de decisión, usar las combinaciones para cambiar las reglas. La lógica detrás de esto es que cada regla puede basarse en una característica útil del problema, y si esto es cierto, entonces, alguna combinación que pondere el criterio de evaluación de las reglas, de alguna manera puede contener implícitamente más criterio que una sola regla por separado, y puede dar mejores resultados.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la red española de procedimientos metaheurísticos HEUR, TIN2004-20061E. El primer autor ha sido parcialmente financiado por el proyecto DPI2004-03475 del MEC del Gobierno Español, la empresa Nissan y por la Cátedra Nissan de la Universitat Politècnica de Catalunya. El tercer autor ha sido parcialmente financiado por la Cátedra de Investigación en Ingeniería Industrial del Tecnológico de Monterrey. Agradecemos también a Fred Glover sus inestimables comentarios acerca de la combinación ponderada de reglas.

Referencias

[1] Agnetis, A., Ciancimino, A., Lucertini, M. y Pizzichella, M. Balancing Flexible Lines for Car Components Assembly. International Journal of Production Research (1995) 33, 333-350

[2] Bautista, J. y Pereira, J. Ant Algorithms for Assembly Line Balancing. Lecture Notes in Computer Science (2002) 2463, Springer, Berlín 65-75.

[3] Baybars, I. A survey of exact algorithms for the simple assembly line balancing problem. Management Science (1986) 32 (8) 909-932.

[4] Crowston, W.B., Glover, F., Thompson, G.L. y Trawick. J.D. (1963). "Probabilistic and

Parametric Learning Combinations of Local Job Shop Scheduling Rules", ONR Research Memorandum No. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA.

[5] Daganzo, C.F y Blumfield, D.E. Assembly System Design Principles and Tradeoffs, International Journal of Production Research (1994) 32, 669-681

[6] Deckro, R.F. Balancing Cycle Time and Workstations. IIE Transactions (1989) 21, 106-111

[7] Glover, F., "A Template for Scatter Search and Path Relinking" en Artificial Evolution, Lecture Notes in Computer Science, 1363, J.-K. Hao, E. Lutton, E. Ronald , M. Schoenauer and D. Snyers, Eds. Springer, 1998, pp. 13-54.

[8] Hackman, S.T., Magazine, M.J. y Wee, T.S. Fast, Effective Algorithms for Simple Assembly Line Balancing Problems. Operations Research (1989) 37, 916-924.

[9] Hoffmann, T.R. Eureka. A hybrid system for assembly line Balancing. Management Science, (1992) 38 (1), 39-47.

[10] Johnson R.V. Optimally balancing assembly lines with "FABLE". Management Science (1988) 34, 240-253

[11] Martí, R., Laguna, M. y Glover, F. (2003), Principles of Scatter Search, European Journal of Operational Research, por aparecer (disponible en http://www.uv.es/~rmarti/.

[12] Rachamadugu R. y Talbot, B. Improving the equality of workload assignments in assembly lines. International Journal of Production Research (1991) 29, 619-633

[13] Scholl, A. y Klein, R. Balancing Assembly lines effectively – A computational comparison, European Journal of Operational Research (1999) 114,50-58

[14] Scholl A. y Voss, S. Simple assembly line balancing – Heuristic approaches, Journal of Heuristics (1996) 2, 217-244.

[15] Scholl, A. Balancing and Sequencing of Assembly Lines, Physica-Verlag, Heidelberg (1999)

[16] Scholl A. y Becker, C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Jenaer Schriften zur Wirtschaftswissenschaft 20/03, FSU Jena (2003). Por aparecer en: EJOR, special issue on "Balancing of Automated Assembly and Transfer Lines".

[17] Talbot, F.B., Patterson, J.H. y Gehrlein, W.V. A comparative evaluation of Heuristic Line Balancing Techniques, Management Science (1986) 32, 430-454.

Anexo I

Nomenclatura:
i, j Índices de tarea
n Número de tareas
c Tiempo de ciclo
t_i Tiempo de operación de i.
IS_i Conjunto de tareas siguientes inmediatas a i.
TS_i Conjunto de tareas siguientes a i.
TP_i Conjunto de tareas precedentes a i.
Li Nivel de la tarea i en el grafo de precedencias.
Z Conjunto de tareas candidatas en curso

Asignar la tarea z^* : $v(z^*)=\max_{i\in Z}[v(i)]$

Nombre	Regla
1. Mayor tiempo de operación	$v(i) = t_i$
2. Mayor número de sucesoras inmediatas	$v(i) = \lvert IS_i \rvert$
3. Mayor número de sucesoras.	$v(i) = \lvert TS_i \rvert$
4. Mayor peso posicional	$v(i) = t_i + \Sigma_{j\in TS_i} t_j$
5. Mayor peso medio posicional	$v(i) = \frac{t_i + \Sigma_{j\in TS_i} t_j}{\lvert TS_i \rvert + 1}$
6. Menor cota superior: UB	$v(i) = -UB_i = -n - 1 + \left\lceil \frac{t_i + \Sigma_{j\in TS_i} t_j}{c} \right\rceil$
7. Menor UB / sucesoras	$v(i) = -UB_i / (\lvert TS_i \rvert + 1)$
8.Mayor tiempo de operación / UB	$v(i) = t_i / UB_i$
9.Menor cota Inferior: LB	$v(i) = -LB_i = -\left\lceil \frac{t_i + \Sigma_{j\in TP_i} t_j}{c} \right\rceil$
10. Menor holgura.	$v(i) = -(UB_i - LB_i)$
11. Sucesoras / Holgura	$v(i) = \lvert TS_i \rvert / (UB_i - LB_i)$
12.Bhattcharjee & Sahu	$v(i) = t_i + \lvert TS_i \rvert$
13. Kilbridge & Wester	$v(i) = -L_i$
14. Menor tiempo de operación	$v(i) = -t_i$
15. Aleatorio.	*