III Jornadas de Ingeniería de Organización
Barcelona, 16-17 de setiembre de 1999

HEURÍSTICAS DE BÚSQUEDA LOCAL APLICADAS AL PROBLEMA DE EQUILIBRADO DE LÍNEAS DE MONTAJE CON INCOMPATIBILIDADES ENTRE TAREAS*.

Bautista, J.; Carreras, M.; Domingo, E.; Companys, R.; Mateo, M.

Departamento de Organización de Empresas e
Instituto de Organización y Control de Sistemas Industriales
ETSEIB. Universidad Politécnica de Cataluña.
Emails de contacto: {bautista, mateo}@ioc.upc.es, companys@oe.upc.es

RESUMEN

El presente trabajo aborda el problema de equilibrado de líneas de montaje teniendo en consideración incompatibilidades entre tareas con el propósito de minimizar el número de estaciones de trabajo y secundariamente el tiempo de ciclo. Para resolver el problema se han propuesto procedimientos GRASP (Greedy Randomized Adaptative Search Procedure) adaptados de métodos heurísticos clásicos basados en reglas de prioridad y un algoritmo genético que centra la búsqueda de soluciones en el espacio de heurísticas. Se realiza, finalmente, una experiencia computacional para contrastar la calidad de las soluciones ofrecida por los procedimientos indicados.

Palabras clave: líneas de producción y montaje, equilibrado de líneas, heurísticas, Grasp, algoritmos genéticos.

1. INTRODUCCIÓN

El problema de equilibrado de líneas de montaje ALBP (*Assembly Line Balancing Problem*), según la clasificación propuesta en Baybars (1986), se puede dividir en dos grandes grupos: SALBP (Simple) y GALBP (General) El primero consiste en definir o diseñar un conjunto de estaciones de trabajo, con idéntico tiempo de ciclo, a partir de un conjunto de tareas con duraciones deterministas conocidas y que sólo se pueden realizar en una de las estaciones; las tareas pueden presentar relaciones de precedencia y la función objetivo admite dos variantes: (1) minimizar el número de estaciones para un tiempo de ciclo dado (SALBP-1) y (2) minimizar el tiempo de ciclo para un número de estaciones fijo (SALBP-2) El segundo grupo incluye el resto de problemas.

* Este trabajo ha sido parcialmente subvencionado por el proyecto TAP98-0494.

En el presente trabajo tratamos una extensión del problema SALBP-1, como objetivo primario, al que se han añadido incompatibilidades entre tareas (manos límpias-sucias, arriba-abajo, derecha-izquierda, etc.) de manera que dos tareas incompatibles no se pueden asignar a la misma estación; como objetivo secundario, se propone minimizar el tiempo de ciclo, tras fijar el número de estaciones de trabajo resultante al considerar el primer objetivo.

Para la resolución exacta de SALBP se han empleado diversos procedimientos, entre ellos los basados en *branch and bound*, como en Hoffmann (1992), aunque su aplicación sólo es viable para ejemplares de dimensiones pequeñas debido al carácter NP-hard del problema. Para resolver ejemplares con las dimensiones que requiere la industria son válidos los procedimientos heurísticos, tanto para el problema SALB-2, Ugurdag et all. (1997), como para el SALBP-1, Boctor (1995).

Para este último problema destacan las heurísticas *greedy* basadas en el empleo de reglas de prioridad que condicionan el orden de asignación de las tareas a las estaciones de trabajo. Las reglas se refieren o combinan aspectos tales como el tiempo de proceso de cada tarea, número de tareas siguientes a una dada, cotas sobre el número mínimo de estaciones para completar la asignación, etc. y sirven para establecer una ordenación de las tareas en cada iteración, con el propósito de seleccionar la más apropiada (según la regla) entre un conjunto de candidatas compatibles con la parte de la solución ya construida; las tareas candidatas son, evidentemente, aquéllas cuyas precedentes han sido asignadas y su tiempo de proceso no supera el tiempo disponible en la estación receptora. Usualmente, cada heurística de este tipo tiene asociada una sola regla, y la regla determina (salvo que se emplee el sorteo) la tarea que debe asignarse en cada paso. La aplicación de este tipo de algoritmos heurísticos suele ofrecer soluciones aceptables, en promedio, tanto más cuanto mayor sea el número de aspectos que combina la regla; no obstante, no puede concluirse que exista una única regla que supere a todas las demás ante cualquier ejemplar de problema; por otra parte, salvo que se incorpore el azar al procedimiento, siempre ofrecerán las mismas soluciones.

Una segunda clase de heurísticas constituida por los algoritmos GRASP (*Greedy Randomized Adaptative Search Procedure*) se ha empleado con éxito en diversas aplicaciones de la Ingeniería de Organización, tal como aparece en la recopilación de González en Díaz et all. (1996)

Finalmente, una tercera clase de heurísticas es los métodos de búsqueda local o procedimientos de exploración de entornos, Díaz et all. (1996), entre ellos se encuentran: los procesos de escalado (HC), recocido simulado (SA), búsqueda tabú (TS) y los algoritmos genéticos (GA). Esta clase de heurísticas proporciona vías alternativas para buscar soluciones en un espacio delimitado por la definición de un vecindario. No obstante, si la definición del vecindario es general y válida para cualquier problema combinatorio, se pierde entonces el conocimiento sobre el problema específico que sí es tenido en cuenta por las heurísticas *greedy*.

Aquí se proponen procedimientos que incorporan los aspectos positivos de las tres familias de heurísticas indicadas: 1) el conocimiento sobre el SALBP que ofrecen las reglas de prioridad específicas del problema y 2) la posibilidad deseable en todo problema combinatorio de generar soluciones en el espacio de búsqueda.

2. HEURÍSTICAS BÁSICAS GREEDY Y GRASP

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

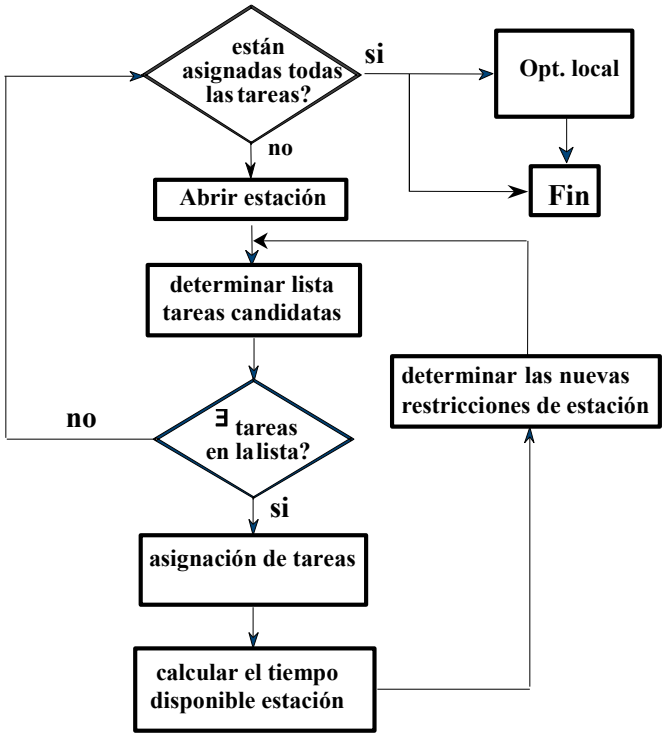

Figura 1: Esquema del procedimiento de obtención de una solución para el problema SALBP.

El esquema es válido tanto para las heurísticas *greedy* deterministas como para GRASP. Las primeras efectúan la asignación de la tarea que presenta mejor valor del índice asociado a la regla (o reglas) de prioridad que distingue al algoritmo; la selección se realiza entre un conjunto de tareas compatibles (con la parte de la solución construida) que satisfacen las restricciones de precedencia, tiempo e incompatibilidad; a la solución hallada 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 (GRWASP: *Greedy Randomized Weighted Adaptative Search Procedure*); en cualquier caso a la solución hallada en la primera fase se aplica un procedimiento de optimización local. Evidentemente,

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 14 reglas básicas.

3. HEURÍSTICAS DE BÚSQUEDA LOCAL

Los métodos de búsqueda local se emplean para explorar entornos constituidos por soluciones denominadas vecinas. Una forma de definir el vecindario (conjunto de soluciones vecinas a una concreta) consiste en caracterizar una solución mediante una secuencia de elementos (p.e. una secuencia de tareas de ensamblado que establece el orden en que se deben asignar a las estaciones de trabajo) y realizar intercambios de elementos entre posiciones de la secuencia siguiendo unas reglas. Está claro que este tipo de definición de vecindario es general y no explota la información específica del problema a resolver, aunque ofrece la ventaja de ser universal y aplicable a todo problema se secuencias.

En Storer (1992), se han propuesto otras alternativas para la definición de vecindarios basadas en la relación que existe entre una heurística h y la solución s que se obtiene al aplicarla a un ejemplar del problema p: $h(p) = s$. Esta relación permite establecer vecindarios en el espacio de los problemas y en el espacio de las heurísticas.

La definición de vecindarios en el espacio de los problemas se basa en la idea de perturbar aleatoriamente los datos del problema, dentro de unos límites razonables, y después aplicar una heurística para obtener una solución; obviamente, la solución se evalúa posteriormente con los datos originales.

Por su parte, la clave para definir vecindarios en el espacio de las heurísticas está en la posibilidad de desarrollar nuevas versiones parametrizadas del conjunto de heurísticas disponibles y específicas de un problema. Esta última alternativa se puede desarrollar al menos de dos formas distintas en el SALBP:

1. Definir una nueva regla híbrida mediante una combinación lineal ponderada del conjunto de reglas originales disponibles de asignación ρ_i: $\rho = \sum_{\forall i} \pi_i \rho_i$
2. Dividir las decisiones de asignación de las tareas a la estación abierta en grupos o ventanas y asociar a cada ventana una regla. Esta segunda vía sugiere la posibilidad de caracterizar una solución a través de un vector de reglas; esto es: $r = (\rho_{[1]}, \rho_{[2]}, ..., \rho_{[N]})$; donde N es el número de tareas y $\rho_{[k]}$ es la regla que se aplica en la k-ésima decisión de asignación.

Centrándonos en la segunda alternativa para realizar búsquedas en el espacio de heurísticas, podemos concluir que dado un vector de reglas r, y un procedimiento A, se puede definir una heurística h a partir del par (r, A): $h = h(r,A)$; en particular, son válidas todas las heurísticas resultantes de la combinación de un vector de reglas y el procedimiento presentado en la figura 1. Para fijar ideas, a

continuación se describe un método para generar soluciones en el espacio de heurísticas bajo la forma de un algoritmo genético.

Notación

I número de elementos de la población
L número de generaciones
p instancia del problema a resolver
Π_r población de ascendentes de secuencias de reglas
Π_h población de ascendentes de heurísticas
Π_s población de ascendentes de soluciones
Δ_r población de descendientes de secuencias de reglas
Δ_h población de descendientes de heurísticas
Δ_s población de descendientes de soluciones
Λ_r población de descendientes mutados de secuencias de reglas
Λ_h población de descendientes mutados de heurísticas
Λ_s población de descendientes mutados de soluciones
Ω_r población de elementos regenerables de secuencias de reglas
r_i i-ésimo elemento de Π_r o Δ_r o Λ_r,o Ω_r
h_i i-ésimo elemento de Π_h o Δ_h o Λ_h o Ω_h
s_i i-ésimo elemento de Π_s o Δ_s o Λ_s o Ω_s

Algoritmo *GA*

0. Inicializar:
   0.1 Generar una población inicial Π_r de I secuencias homogéneas y distintas de reglas: $\Pi_r = \{ r_i = (\rho_{[1]},..., \rho_{[N]}) : \rho_{[1]} = ..= \rho_{[N]} \}$
   0.2 Definir la población inicial de heurísticas: $\Pi_h = \{ h_i = h_i(r_i, A) : r_i \in \Pi_r \}$
   0.3 Generar la población de soluciones de p y obtener: $\Pi_s = \{ s_i = h_i(p) : h_i \in \Pi_h \}$
   0.4 Determinar la aptitud (fitnees) de los elementos de Π_s de la forma:

$$f_j = -NE_j + \left[\frac{\sum_{i=1}^{N} t_i}{C} \right]^+ + \frac{\sqrt{\sum_{k}^{NE_j} (c_k - C)^2}}{C\sqrt{NE_j}}$$

donde:

t_i duración de la tarea i
C tiempo de ciclo
NE_j número de estaciones de la j-ésima solución
c_k tiempo de ocupación en la k-ésima estación ($1 \le k \le NE_j$)

   0.5 Guardar como incumbentes el par (h^*,s^*) que presenta mayor aptitud

Iterar L veces a través de los pasos siguientes:

1. Selección de ascendentes: Constituir $I/2$ parejas de elementos de Π_r en función de la aptitud de los elementos de Π_s .
2. Selección de parejas a cruzar:
   2.1 Determinar la probabilidad de cruce de la generación en curso: $P_c = P_c(\alpha)$
   2.2 Asignar un número aleatorio a cada pareja de secuencias de reglas
   2.3 Decidir para cada pareja de secuencias de reglas si hay cruce o no en función de su número aleatorio y de P_c dependiente de la homogeneidad de la población α.

   con $\alpha = \frac{1}{I(I-1)} \sum_{i=1}^{I} \frac{1}{N} \sum_{j \neq i} h_{i,j}$

   donde $h_{i,j} \in \{0,1\} \wedge [\, h_{i,j} = 1 \Leftrightarrow \rho_{[k]}(\in r_i) = \rho_{[k]}(\in r_j)\ \forall k\ ,\ 1 \leq k \leq N\,]$
3. Generar descendientes:
   3.1 Cruzar las parejas de secuencias de reglas seleccionadas, generando dos descendientes por cada pareja como mínimo. Se crea Δ_r.
   3.2 Generar Δ_h y Δ_s a partir de Δ_r de forma análoga a 0.2 y 0.3, respectivamente.
   3.3 Determinar la aptitud de los elementos de Δ_s .Si algún elemento de Δ_s presenta mejor aptitud que el de la solución incumbente, guardar como heurística y solución incumbentes el par (h^*,s^*) asociado a dicho elemento.
4. Mutar descendientes:
   4.1 Determinar la probabilidad de mutación de la generación en curso: $P_m = P_m(\alpha)$
   4.2 Asignar un número aleatorio a cada elemento de Δ_r.
   4.3 Decidir qué elementos de Δ_r se mutan o no en función de su número aleatorio y de P_m.
   4.4 Mutar los elementos de Δ_r seleccionados. Se crea Λ_r.
   4.5 Generar Λ_h y Λ_s a partir de Λ_r de forma análoga a 0.2 y 0.3, respectivamente.
   4.6 Determinar la aptitud de los elementos de Λ_s Si algún elemento de Λ_s presenta mejor aptitud que el de la solución incumbente, guardar como heurística y solución incumbentes el par (h^*,s^*) asociado a dicho elemento.
5. Regeneración de la población:
   5.1 Construir la población de elementos regenerables: $\Omega_r \leftarrow \Pi_r + \Delta_r + \Lambda_r$
   5.2 Determinar la aptitud de los elementos de las poblaciones Δ_s y Λ_s como se indica en 0.4.
   5.3 Seleccionar I elementos del conjunto Ω_r en función de la aptitud de los elementos de los conjuntos Π_s , Δ_s y Λ_s.

4. EXPERIENCIA COMPUTACIONAL

Se han programado 13 heurísticas Greedy deterministas (que se corresponden con las reglas del Anexo I); 12 algoritmos GRASP: 6 tradicionales y 6 con probabilidad de selección de las tareas directamente proporcional al valor de la regla empleada (tanto GRASP tradicional como revisado incorpora las reglas 1, 4, 5, 8, 11 y 12); y 7 variantes de un algoritmo genético (según operadores de mutación, cruce y proceso de regeneración) generador de soluciones en el espacio de heurísticas y con probabilidades de cruce y mutación dependientes del índice

de homogeneidad, α, de la población: $P_c = 1 - 0.5\alpha$ y $P_m = 0.05 + 0.95\alpha$. Con estos procedimientos se han resuelto 160 ejemplares divididos en 4 grupos según el número de tareas: 20, 40, 60 y 80, y con valores de *order strength* de Mastor y ratio de incompatibilidad de Rachamadugu comprendidos entre 0.05 y 0.75. Los resultados globales obtenidos, media de la razón entre el valor ofrecido para cada ejemplar por cada procedimiento y el mejor valor conseguido, se muestran en la Figura 2.

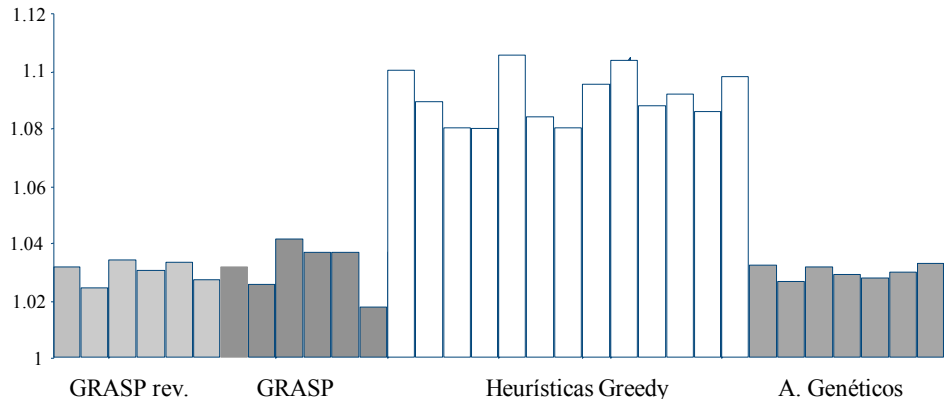

Figura 2: Resultados obtenidos para 160 ejemplares resueltos por 32 procedimientos.

5. CONCLUSIONES

El presente trabajo propone una vía para explorar soluciones en el problema SALBP con incompatibilidades entre tareas. La idea básica consiste en incorporar a los procedimientos de búsqueda local (HC, SA, TS, AG) o a los de búsqueda adaptativa con azar (GRASP) el conocimiento que aportan las heurísticas *greedy* deterministas específicas del problema. De esta forma se puede caracterizar una solución mediante una secuencia de reglas de prioridad o tomar decisiones con sorteo de los elementos compatibles condicionado por dichas reglas. Para ilustrar esta idea se han descrito AG y GRASP cuyas explotaciones ha ofrecido resultados satisfactorios.

6. REFERENCIAS

BAYBARS, I. (1986); "A survey of exact algorithms for the simple assembly line balancing problem". *Management Science*, Vol 32, Nº 8, pp 909-932.
BOCTOR F.F. (1995); "A Multiple-rule Heuristic for Assembly Line Balancing". Journal of the Operational Research Society, 46, pp 62-69.
DÍAZ, A.; GLOVER, F.; GHAZIRI, H; GONZÁLEZ, J.M.; LAGUNA, M.; MOSCATO, P.; TSENG, F. (1996). *Optimización heurística y redes neuronales*. Paraninfo.
HOFFMANN, T.R. (1992): "Eureka. A hybrid system for assembly line alancing". *Management Science*, V 38, Nº 1, pp 39-47.

STORER, R.H.; WU, S.D.; VACCARI, R. (1992): "New search spaces for sequencing problems with application to Job Shop Scheduling". *Management Science.* V 38, Nº 10, pp1495-1509.
UGURDAG, H.F.; RACHAMADUGU, R.; PAPACHRISTOU, CH.A. (1997): "Designing paced assembly lines with fixed number of stations". *European Journal of Operational Research*, V 102, Nº 3, pp 488-501.

ANEXO I

Nomenclatura

i, j Índices de tarea
N Número de tareas
C Tiempo de ciclo
t_i Duración de la tarea i.,
IS_i Conjunto de tareas siguientes inmediatas a la tarea i.
S_i Conjunto de tareas siguientes a la tarea i.
TP_i Conjunto de tareas precedentes a la tarea i.
Li Nivel de la tarea i en el grafo de precedencias.

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

Nombre	Regla
1.*Longest Processing Time*	$v(i) = t_i$
2.*Greatest Number of Inmediate Successors*	$v(i) = \| IS_i \|$
3.*Greatest Number of Successors*	$v(i) = \| S_i \|$
4.*Greatest Ranked Positional Weight*	$v(i) = t_i + \sum t_j \ (j \in S_i)$
5.*Greatest Average Ranked Positional Weight*	$v(i) = (t_i + \sum t_j \ (j \in S_i)) / (\| S_i \| + 1)$
6.*Smallest Upper Bound*	$v(i) = -UB_i = -N - 1 + [(t_i + \sum t_j \ (j \in S_i))/C]^+$
7.*Smallest Upper Bound / Number of Successors*	$v(i) = -UB_i / (\| S_i \| + 1)$
8.*Greatest Processing Time / Upper Bound*	$v(i) = t_i / UB_i$
9.*Smallest Lower Bound*	$v(i) = -LB_i = -[(t_i + \sum t_j \ (j \in TP_i)) / C]^+$
10. *Minimum Slack*	$v(i) = -(UB_i - LB_i)$
11.*Maximum Number Successors / Slack*	$v(i) = \| S_i \| / (UB_i - LB_i)$
12.*Bhattcharjee & Sahu*	$v(i) = t_i + \| S_i \|$
13. *Etiquetas Kilbridge & Wester*	$v(i) = -L_i$