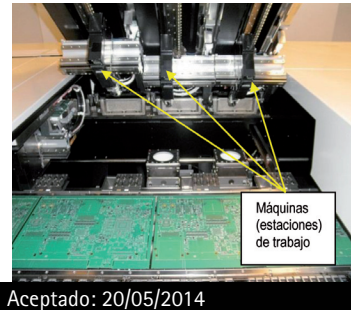


Equilibrado de líneas de montaje con ventanas de visibilidad.

Descripción del problema y procedimiento heurístico de resolución

BALANCING ASSEMBLY LINES WITH ACCESSIBILITY WINDOWS.
PROBLEM DESCRIPTION AND SOLVING HEURISTIC PROCEDURE

DOI: <http://dx.doi.org/10.6036/7051> | Recibido: 16/12/2013 • Aceptado: 20/05/2014



■ ■ ■ ■ ■
Gema Calleja-Sanz, Albert Corominas-Subías,
Alberto García-Villoria y Rafael Pastor-Moreno

Universitat Politècnica de Catalunya (UPC). Instituto de
Organización y Control de Sistemas Industriales (IOC). Dpto
Organización de Empresas. Avda. Diagonal, 647 – 08028
Barcelona. Tfno: +34 93 4054010. gema.calleja@upc.edu

ABSTRACT

- Assembly line balancing problems (ALBPs) arise when designing or modifying an assembly or production line and usually consist in assigning the tasks to the workstations along the line in order to meet some given objectives. Traditionally, it is assumed that there is exactly one workpiece in each workstation and that the workstation has access to the entire workpiece. In practice, this assumption may be sometimes too restrictive. An example is the case of the automated assembly of printed circuit boards. In this type of lines the length of the workpiece might be larger than the width of the workstation. Therefore, each workstation can only access to the limited portion of the workpieces that fall inside its accessibility window. In this work we present an extension of the ALBP to the case of workstations with accessibility windows (AWALBP). The problem is described and a simple-to-implement solving procedure is proposed, which consists of i) an algorithm to obtain a good initial solution and ii) a simulated annealing algorithm for improving the obtained solution. Finally, a computational experience that validates the proposed procedure is reported.
- **Keywords:** assembly line balancing, accessibility windows, heuristics, simulated annealing.

RESUMEN

Los problemas de equilibrado de líneas (*Assembly Line Balancing Problems*, ALBPs) surgen al diseñar o modificar una línea de montaje o de producción y consisten, usualmente, en asignar las tareas a las estaciones de la línea de forma que se cumplan ciertos objetivos. Tradicionalmente se considera que en cada estación hay exactamente una pieza y que la estación tiene acceso a toda la pieza. En la práctica, este supuesto puede ser demasiado restrictivo. Un ejemplo es el caso del ensamblaje automatizado de placas de circuitos impresos. En este tipo de líneas la longitud de la pieza puede ser mayor que el ancho de la estación que la procesa. Como consecuencia, cada estación solamente puede acceder a una porción limitada de las piezas que está dentro de su ventana de accesibilidad. En este trabajo se presenta la extensión del ALBP al caso de estaciones con ventanas de accesibilidad (AWALBP). Se describe el problema y se propone un procedimiento de resolución sencillo de implementar basado en i) un algoritmo para obtener una buena solución inicial y ii) un algoritmo de recocido simulado para mejorar la solución inicial. Finalmente se presentan los resultados computacionales que validan el procedimiento propuesto.

Palabras Clave: equilibrado de líneas, ventanas de accesibilidad, heurísticas, recocido simulado.

1. INTRODUCCIÓN

Entre los problemas de decisión que surgen al gestionar líneas de montaje o de producción, los denominados problemas de equilibrado de líneas de montaje o *assembly line balancing problems* (ALBPs) tienen una gran importancia en la planificación de la producción a medio plazo [1-3]. En una línea de montaje se dispone de un número de estaciones (o máquinas) por la que circulan las piezas a procesar. En cada estación se realiza un cierto número de tareas con una duración limitada (el llamado tiempo de ciclo). El problema ALBP consiste en asignar las tareas a las estaciones de acuerdo con un objetivo dado [4].

En la literatura típicamente se considera el siguiente escenario: cada pieza avanza de una estación hacia la siguiente, cada estación tiene acceso a toda una pieza y cada pieza únicamente puede ser procesada en una estación a la vez. En este trabajo, sin embargo, se considera un escenario diferente: las dimensiones de las piezas son mayores que el ancho (ventana de accesibilidad) de las estaciones. Como resultado, partes de

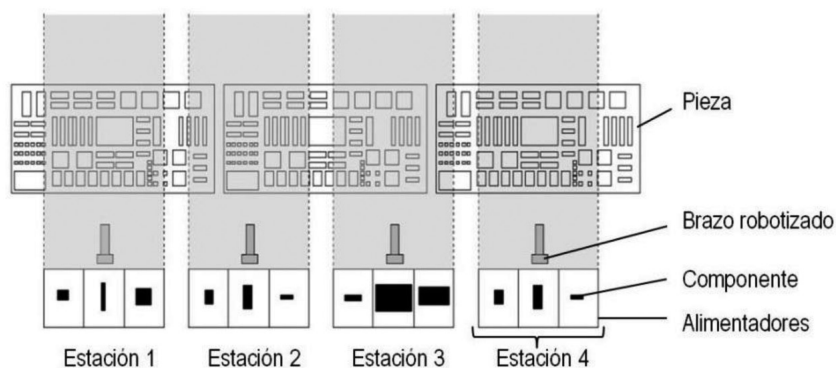


Fig. 1: Ejemplo de una línea de montaje con cuatro estaciones. El área gris representa las ventanas de accesibilidad

una misma pieza pueden ser procesadas en varias estaciones a la vez y una misma estación puede procesar partes de una pieza o de dos piezas consecutivas a la vez. En el ejemplo de la Fig. 1, las estaciones 1 y 4 procesan una parte de una pieza, mientras que las estaciones 2 y 3 procesan partes de dos piezas consecutivas. Este escenario da lugar a una extensión del ALBP conocida como *Accessibility Windows Assembly Line Balancing Problem* (AWALBP).

El AWALBP está presente, por ejemplo, en el montaje automatizado de placas de circuitos impresos (Printed Circuit Boards, PCBs). Una PCB es una superficie de material aislante provista de unas pistas o caminos de material conductor que sirven para interconectar un conjunto de componentes electrónicos. Cada tipo de placa tiene asociado un formato que indica la posición predefinida en la que debe ser colocado un componente específico (por ejemplo, una resistencia o un condensador). Hoy en día, la tecnología de montaje más avanzada agrupa las máquinas o estaciones en forma de celda flexible en la línea (ver Fig. 2), donde cada estación constituye un módulo de montaje de componentes. Cada estación contiene un brazo robotizado, el cual monta varios componentes en la superficie de la placa (ver Fig. 1). El montaje se realiza en paralelo, es

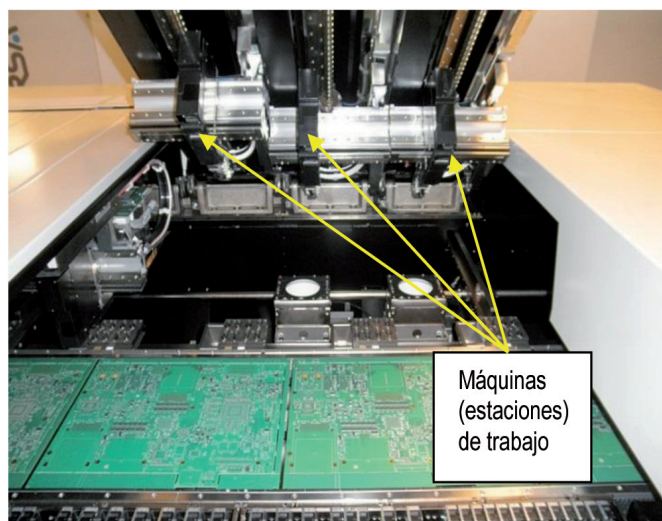


Fig. 2: Interior de una línea de montaje de placas de circuitos impresos con módulos robotizados [5]

decir, los robots montan componentes sobre las placas de forma simultánea. Las tareas de montaje son de tipo *pick & place* y consisten en coger (*pick*) un componente de un alimentador, trasladarlo hasta la placa y colocarlo (*place*) en su ubicación correspondiente.

El AWALBP fue descrito por primera vez por Gaudlitz en 2004 [6]. Posteriormente Calleja *et al.* [7] establecieron una clasificación de las variantes del problema. En este trabajo se considera la variante AWALBP-L2. Se trata de un problema de optimización combinatoria [8] compuesto de dos subproblemas, muy difíciles de resolver, para los que en este artículo se propone una única heurística.

El objetivo de este trabajo consiste en dar a conocer el AWALBP y proponer un procedimiento de resolución de fácil implementación,

consistente en i) una heurística para encontrar una solución factible inicial y ii) un algoritmo de recocido simulado para tratar de mejorar la solución obtenida.

La estructura del resto del trabajo es la que sigue. En la sección 2 se describe el AWALBP. En la sección 3 se presenta el procedimiento propuesto de resolución. En la sección 4 se reportan los resultados de la experiencia computacional que validan el procedimiento propuesto. Finalmente, en la sección 5 se muestran las conclusiones.

2. EL ACCESSIBILITY WINDOWS ASSEMBLY LINE BALANCING PROBLEM (AWALBP)

Una línea de montaje debe procesar un número (potencialmente infinito) de piezas idénticas. Sobre la línea se encuentran colocadas varias piezas simultáneamente y la distancia entre dos piezas consecutivas es constante. Cada estación de la línea debe realizar un conjunto específico de tareas sobre cada pieza. Cada tarea se ejecuta en una posición predefinida de la superficie de la pieza. En el caso del AWALBP, las piezas tienen mayores dimensiones que las estaciones, por lo que una tarea solamente puede llevarse a cabo si está dentro de la ventana de accesibilidad de la estación en la que debe ser ejecutada. La ventana de accesibilidad de la estación i está definida en un intervalo $[L_i, \dots, R_i]$ de la línea (ver Fig. 3). La posición inicial de las piezas en la línea está determinada por la distancia x , correspondiente a la distancia del borde derecho de la primera pieza en la primera etapa del montaje respecto al límite izquierdo de la primera estación de la línea. La distancia entre los bordes derechos (o izquierdos) de dos piezas consecutivas en la línea es A . El proceso de montaje se descompone en un número S de etapas estacionarias. Una etapa estacionaria es aquella etapa del ciclo de montaje en la que la línea (y las piezas) permanecen inmóviles, y las estaciones efectúan tareas sobre las piezas. Una vez finalizada una etapa estacionaria, la línea, con todas las piezas sobre ella, se desplaza según una determinada distancia, llamada paso de avance s ($s = 1, \dots, S$), y a continuación se inicia una nueva etapa estacionaria. Los pasos de avance son cíclicos: cada S pasos, cada pieza ocupa en la línea exactamente la misma posición que su predecesora (como se ve en la Fig. 3).

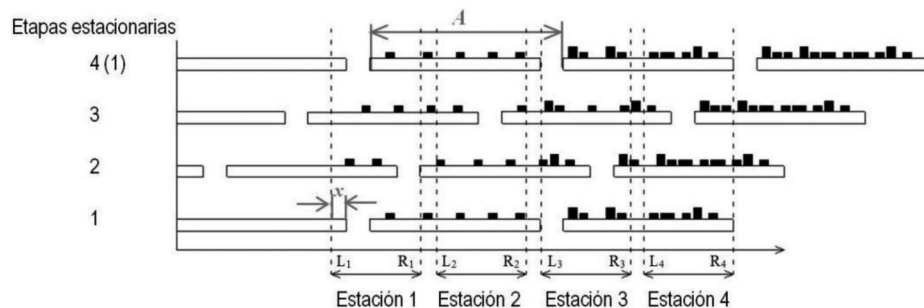


Fig. 3: Posición de las piezas de un ciclo de montaje con tres etapas estacionarias ($S = 3$)

La Fig. 3 muestra el proceso de montaje en una línea con cuatro estaciones y un ciclo de tres etapas estacionarias (y por tanto, tres pasos de avance). El movimiento de las piezas a través de las estaciones sigue un patrón cíclico llamado *esquema de avance*. En la primera etapa estacionaria del ciclo de montaje, una nueva pieza entra en la línea. Después de la última etapa del ciclo, una pieza completa abandona la línea y se inicia un nuevo ciclo.

EL AWALBP da lugar a varias versiones del problema, tal como se describe en [7]. Este trabajo se centra en la resolución del AWALBP-L2, el cual consiste en determinar:

- 1) Un esquema de avance, formado por:
 - El número S de etapas estacionarias (igual al número S de pasos de avance).
 - El valor de los desplazamientos de los pasos de avance $\Delta \cdot \delta_1, \dots, \Delta \cdot \delta_s$, donde Δ es una distancia llamada paso elemental, que depende de la tecnología de la línea, y δ_s ($s=1, \dots, S$) es el número de pasos elementales recorridos en el paso de avance s .
 - El valor de la posición inicial de las piezas en la línea $x \geq 0$.
- 2) La asignación de cada una de las tareas a una de las etapas estacionarias del ciclo. Tal como se describe en el caso real descrito en Müller-Hannemann y Weihe [9], en este trabajo se considera que cada tarea solamente se puede ejecutar en una estación.

Una solución factible del problema debe cumplir las dos condiciones siguientes. En primer lugar, que la suma de las distancias de todos los pasos de avance de un ciclo sea igual a la distancia A (ver Fig. 3). La segunda condición impone que, para cada tarea, su posición esté dentro de la ventana de accesibilidad de su correspondiente estación durante la etapa estacionaria a la que está asignada y es ejecutada.

La función objetivo a minimizar (1) es el tiempo de ciclo TC , que equivale a minimizar la suma de: i) el tiempo correspondiente al desplazamiento de los pasos de avance (igual al tiempo para transportar las piezas a velocidad constante más un tiempo adicional T por cada paso de avance) y ii) las duraciones, C_s , de las S etapas estacionarias:

$$TC = T \cdot S + \sum_{s=1}^S C_s \quad (1)$$

3. PROCEDIMIENTO DE RESOLUCIÓN

En trabajos anteriores se abordó la resolución del AWALBP mediante procedimientos exactos [7], [10]. Se consiguió resolver óptimamente ejemplares realistas de grandes dimensiones en un tiempo computacional práctico (dentro de un límite máximo de una hora). Sin embargo, para los ejemplares de mayores dimensiones el problema no se pudo resolver en el tiempo máximo de una hora establecido. En este trabajo se propone un procedimiento de

resolución heurístico que permite resolver ejemplares de cualquier dimensión y que puede ser implementado de forma sencilla. La idea fundamental consiste en hallar, en primer lugar, una solución inicial mediante una heurística. Opcionalmente, en una segunda etapa, se puede tratar de mejorar la solución inicial obtenida mediante un algoritmo de recocido simulado.

En la Fig. 4 se muestra un esquema del procedimiento propuesto. Los pasos principales son los siguientes:

Paso 1: Mediante una heurística, se genera una solución para cada uno de los posibles valores de x . De esta manera, la heurística proporciona tantas posibles soluciones iniciales como posibles valores de x .

Paso 2: Entre las soluciones obtenidas en el *Paso 1*, se fija, como *solución inicial*, aquella con el menor valor del tiempo de ciclo, y se anota dicho valor, $\overline{TC}$. Por otra parte, se guarda el valor del menor número de etapas estacionarias obtenido entre todas las soluciones del *Paso 1*, $\overline{E}$, el cual permite obtener una cota inferior del tiempo de ciclo, CI .

Paso 3: Si el valor del menor tiempo de ciclo obtenido coincide con el valor de la cota inferior del tiempo de ciclo ($\overline{TC} = CI$), la solución inicial es óptima y finaliza el procedimiento. En caso contrario, se intenta mejorar la solución inicial mediante el *Paso 4*.

Paso 4: Se ejecuta un procedimiento de recocido simulado con un tiempo computacional predeterminado.

En los apartados 3.1 y 3.2 se detalla el funcionamiento de la heurística y el cálculo de la cota, respectivamente. Finalmente, en el apartado 3.3 se describe el procedimiento de recocido simulado.

3.1. HEURISTICA

Para obtener una solución heurística inicial, el AWALBP se descompone en dos subproblemas: i) hallar un esquema de avance y ii) una asignación de tareas a etapas estacionarias, los cuales se detallan en los apartados 3.1.1 y 3.1.2, respectivamente.

3.1.1. Esquema de avance

La heurística propuesta calcula un esquema de avance factible, para cada uno de los posibles valores de la posición

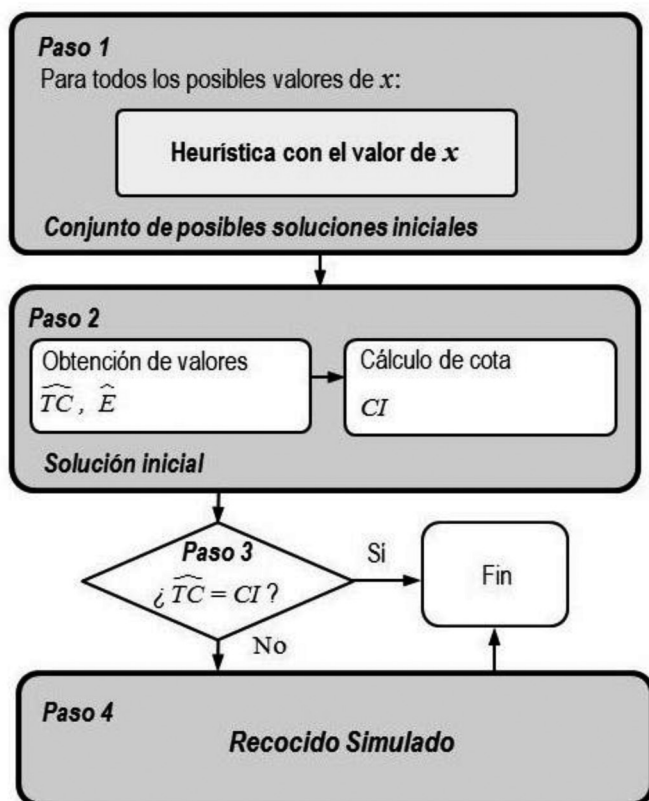


Figura 4: Procedimiento de resolución propuesto

inicial x , mediante un algoritmo diseñado a medida (*algoritmo Avance*).

La idea básica del *algoritmo Avance* consiste en determinar el valor máximo de los pasos de avance de la línea de forma que cada tarea sea accesible en su estación correspondiente en, al menos, una de las etapas estacionarias. El funcionamiento del algoritmo, para un valor determinado de la posición inicial x , es el siguiente. En cada iteración se anota, para cada una de las estaciones de la línea, la distancia (recuérdese, medida en pasos elementales) de la posición de la primera tarea situada a la izquierda de la estación (fuera de su ventana de accesibilidad) al límite derecho de la estación. El valor de esta distancia indica la distancia máxima que puede recorrer una tarea para que siga siendo accesible desde su estación correspondiente. El mínimo valor entre las distancias calculadas para todas las estaciones se denota como $max_desplaz_línea$ y proporciona el valor del primer paso de avance de la línea (δ_1). En la Fig. 5 se muestran dos instantáneas correspondiente a las dos etapas ($s = 1$ y $s = 2$) de un ciclo en una línea con tres estaciones (A , B y C) y $x = 3$. Los rectángulos en negro representan las tareas a realizar e indican la estación en la que se deben procesar. En la primera etapa, la primera tarea a la izquierda de las estaciones A , B y C se podría desplazar como máximo una distancia de 6, 8 y 10 pasos elementales antes de sobrepasar la ventana de accesibilidad, respectivamente, por lo que $max_desplaz_línea = 6$. Nótese que, dado que la suma de los pasos de avance en un ciclo ha de ser igual a la distancia A , para calcular el valor de un paso de avance δ_s se debe tener en cuenta el desplazamiento acumulado hasta el momento en la línea, d . Por tanto, el valor de δ_s se calcula mediante la expresión $\delta_s = \min(A-d, max_desplaz_línea)$. Para el primer paso de avance,

se tiene, con $d = 0$, $\delta_1 = \min(10-0,6)$, y por tanto $\delta_1 = 6$. Tal como muestra la Fig. 5, una vez producido el primer paso de avance, las piezas ocupan una nueva posición en la línea. De nuevo se anota la distancia máxima de desplazamiento para cada estación: 10, 12 y 15 pasos elementales, y su valor mínimo correspondiente, $max_desplaz_línea = 10$. Por tanto, para el segundo paso de avance se obtiene un valor $\delta_2 = \min(10-6,6)$. El algoritmo finaliza cuando el desplazamiento acumulado en la línea d es igual a A . En el caso presentado, el desplazamiento acumulado es $d = \delta_1 + \delta_2 = 10 = A$, por lo que finaliza el algoritmo y se obtiene un ciclo de dos etapas estacionarias con pasos de avance $\delta_1 = 6$ y $\delta_2 = 4$ (para $x = 3$).

3.1.2. Asignación de tareas a etapas estacionarias

Una vez generado un esquema de avance se debe determinar, para dicho esquema de avance, la asignación de cada tarea a una de las etapas estacionarias del ciclo. Para ello se propone un modelo de programación lineal entera (PLE) llamado *modelo Tarea*. El *modelo Tarea* calcula la asignación óptima, tal como se describe a continuación.

Datos

- m número de estaciones ($i = 1, \dots, m$)
- N número de tareas ($j = 1, \dots, N$)
- J_i conjunto de tareas que deben ser realizadas en la estación i , donde $J_i \cap J_{i'} = \emptyset$
 $i = 1, \dots, m; i' = 1, \dots, m; i \neq i'$
- p_j tiempo de proceso de la tarea j ($j = 1, \dots, N$)
- S número de pasos de avance del ciclo
- Π_j conjunto de etapas estacionarias en las que la tarea j es accesible desde la estación donde debe ser realizada ($j = 1, \dots, N$).

Variables

- $y_{js} \in 0,1$ $y_{js} = 1$ si la tarea j se realiza en la etapa estacionaria s ($j = 1, \dots, N; s \in \Pi_j$)
- C_s duración de la etapa estacionaria s ($s = 1, \dots, S$)

Modelo Tarea

$$[MIN] z = \sum_{s=1}^S C_s \quad (2)$$

$$\sum_{s \in \Pi_j} y_{js} = 1 \quad j = 1, \dots, N \quad (3)$$

$$\sum_{j \in J_i | s \in \Pi_j} p_j \cdot y_{js} \leq C_s \quad i = 1, \dots, m; s = 1, \dots, S \quad (4)$$

El objetivo (2) es la minimización de la duración de las etapas estacionarias. Las restricciones (3) imponen que cada tarea debe ser asignada a una, y solamente a una, etapa estacionaria, y (4) garantizan que la duración de las etapas estacionarias no es inferior al tiempo de proceso en ninguna de las estaciones.

3.2. OBTENCIÓN DE LA COTA

A partir de todas las soluciones generadas en el apartado 3.1 se obtiene una cota inferior del valor del tiempo de ciclo, como se indica a continuación.

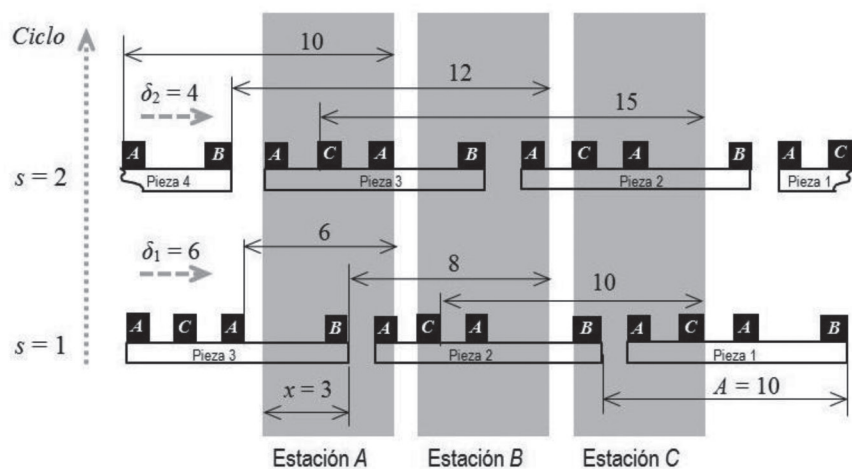


Fig. 5: Determinación del esquema de avance

Entre las soluciones obtenidas, se identifica la solución con el menor número de etapas estacionarias y se anota el valor de dicho número, $\bar{E}$. La heurística desarrollada tiene la propiedad de que, dado un valor de la posición inicial x , genera un esquema de avance factible con el menor número posible de etapas estacionarias S . Por tanto, se obtiene una cota inferior del tiempo de ciclo CI sumando las cotas inferiores de los términos de la función objetivo (1), es decir, i) $T \cdot \bar{E}$ y ii) una cota inferior de $\sum_{s=1}^S C_s$, correspondiente a la suma de las duraciones de las tareas de la estación más cargada, es decir, $\max_{i=1, \dots, m} \sum_{j \in J_i} p_j$.

Una vez obtenida la cota CI , el siguiente paso del procedimiento (Paso 3 de la Fig. 4) consiste en comparar el valor del menor tiempo de ciclo de las soluciones generadas, $\overline{TC}$, con el valor de la cota inferior CI . Si coinciden, significa que la solución inicial es óptima. En caso contrario, se intenta mejorar la solución inicial ejecutando un procedimiento de recocido simulado durante un tiempo computacional predeterminado.

3.3. RECOCIDO SIMULADO

El recocido simulado (o *Simulated Annealing*, SA) es una metaheurística propuesta por Kirkpatrick *et al.* en 1983 [11] para resolver problemas de optimización complejos. Una metaheurística puede definirse como “un marco algorítmico general que puede ser aplicado a diferentes problemas de optimización y que precisa relativamente de pocas modificaciones para ser adaptado a un problema específico” [12]. Desde su aparición, SA ha sido utilizado con éxito en la resolución de un amplio número de problemas de optimización combinatoria [13]. El SA tiene la particularidad de permitir, con una cierta probabilidad, movimientos hacia soluciones peores, con el fin de escapar de óptimos locales.

El funcionamiento básico del recocido simulado es el siguiente.

Se parte de una solución inicial (obtenida mediante una heurística), que inicialmente es la solución en curso. En cada iteración se evalúa una solución del vecindario de la solución en curso, seleccionada aleatoriamente. El vecindario de una solución está constituido por el conjunto de sus soluciones vecinas, considerando que una solución vecina de otra es *parecida*, y se obtiene transformando la primera mediante una regla de transformación. Tras evaluar la solución vecina, si ésta es mejor que la solución en curso, se acepta como la nueva solución en curso. En caso contrario, se admite como nueva solución en curso con una determinada probabilidad que depende de i) el nivel de empeoramiento respecto a la solución inicial, y ii) el valor de un parámetro llamado temperatura, el cual decrece cada cierto número de iteraciones. El proceso descrito se

repite hasta que se cumple una condición de parada dada (por ejemplo, haber superado un determinado tiempo de computación o un número preestablecido de iteraciones). La Fig. 6 muestra el pseudocódigo del recocido simulado.

Para aplicar el procedimiento de SA en la resolución del AWALBP es necesario tomar varias decisiones. Algunas de estas decisiones son generales, mientras que otras son específicas del problema a resolver. Las decisiones específicas del AWALBP corresponden a la generación de una solución inicial, la definición del vecindario de cada solución ($N(s)$) y una manera de evaluar las soluciones vecinas ($f(s)$). Las decisiones generales consisten en determinar una temperatura inicial, una ley de reducción de temperatura $A(t)$ y una condición de parada.

3.3.1. Generación de una solución inicial

En este caso la solución inicial se obtiene como resultado del Paso 2 del procedimiento de la Fig. 4.

Sea $f(s)$ la función objetivo de la solución s , a minimizar

Sea $N(s)$ el vecindario de la solución s

Sea $A(t)$ la nueva temperatura calculada a partir de t

1. Especificar los parámetros:
2. t_0 (temperatura inicial)
3. itt (número de iteraciones en el que la temperatura permanece constante)
4. $t := t_0$
5. $s :=$ Generación de la solución inicial
6. **mientras** no se alcance el criterio de parada **hacer**:
7. $i := 0$
8. **mientras** $i < itt$ **hacer**:
9. $s' :=$ elegir al azar una solución del vecindario $N(s)$
10. $\Delta := f(s') - f(s)$
11. $\Delta \leq 0$ entonces $s := s'$
12. si $\Delta > 0$ entonces $s := s'$ con probabilidad $\exp(-\Delta/t)$
13. $i := i + 1$;
14. **fmientras**
15. $t := A(t)$
16. **fmientras**
17. **mostrar** la mejor solución encontrada

Fig. 6: Pseudocódigo del recocido simulado

3.3.2. Vecindario de las soluciones

El vecindario se genera en el espacio de los esquemas de avance y se representa mediante la unión de los tres vecindarios siguientes: i) cesión de pasos elementales entre dos etapas consecutivas, ii) inserción de una nueva etapa estacionaria y iii) variación del valor del desplazamiento inicial. El esquema de avance vecino se obtiene de uno de los tres vecindarios citados con una probabilidad asociada, las cuales se determinaron mediante el software CALIBRA de Adenso-Díaz y Laguna [14], obteniéndose los valores de 0,75, 0,1 y 0,15, respectivamente. A continuación se aplica el modelo *Tarea* para obtener la asignación de tareas y el tiempo de ciclo correspondiente al esquema de avance dado.

3.3.3. Función objetivo

El valor de la función objetivo $f(s)$ corresponde al tiempo de ciclo TC de la Ecuación (1).

3.3.4. Temperatura inicial

El parámetro de temperatura inicial t_0 se determinó mediante CALIBRA, obteniéndose un valor de 115. Adicionalmente, se determinó un valor de $itt = 1.400$.

3.3.5. Función de temperatura $A(t)$

La temperatura del procedimiento SA influye en la probabilidad de aceptar soluciones vecinas peores. Cuanto mayor es la temperatura, mayor es la probabilidad; y viceversa, cuanto menor es la temperatura, menor es la probabilidad (paso 12 de la Fig. 6). La función de reducción de la temperatura más común en la literatura, que produce buenos resultados, es la reducción geométrica, es decir, $A(t) = t \cdot \alpha$, donde $0 < \alpha < 1$ [11]. El valor de α es también un parámetro a determinar; en este caso se obtuvo con CALIBRA un valor de 0,9875.

3.3.6. Criterio de parada

Finalmente, el procedimiento para cuando la mejor solución encontrada hasta el momento coincide con la cota CI . Si no se cumple esta condición, se establece un límite de tiempo computacional de una hora.

4. RESULTADOS COMPUTACIONALES

Para evaluar la calidad del procedimiento propuesto se realizó la siguiente experiencia computacional. Se considera el juego de datos propuesto en [7], el cual está disponible online en <https://www.ioc.upc.edu/EOLI/research/>. Los ejemplares tienen las siguientes características. El paso elemental Δ es igual a 1 unidad de longitud (ul), el ancho accesible de las estaciones es de 10 ul, y el tiempo T es igual a 200 unidades de tiempo (ut). Los tiempos de proceso de las tareas fueron seleccionados aleatoriamente entre 100 y 150 ut. Se consideran seis rangos crecientes de longitud de pieza A_0 (de 11 a 40), cuatro rangos de número de estaciones m (de 5 a 40) y cinco rangos de número de tareas N (de 50 a 1.000) (Fig. 7). Se obtuvieron $6 \cdot 4 \cdot 5 = 120$ combinaciones de rangos, seleccionando aleato-

A_0	11-15	16-20	21-25	26-30	31-35	36-40
m	5-10	11-20	21-30	31-40		
N	50-200	201-400	401-600	601-800	801-1.000	

Fig. 7: Rangos de valores de A_0 , m y N utilizados en la experiencia computacional

riamente un valor dentro de cada rango. Finalmente, por cada combinación y se generaron 10 ejemplares, obteniendo así un juego de datos de 1.200 ejemplares.

La implementación del procedimiento se realizó como se describe a continuación. El algoritmo *Avance* se implementó en Java. El tiempo de ejecución para obtener un esquema de avance es 0,013s en promedio. El modelo matemático *Tarea* se resolvió utilizando el optimizador IBM ILOG CPLEX 12.2 en un tiempo promedio de 0,006 s. CPLEX es un optimizador comercial; sin embargo, existen optimizadores gratuitos que permiten resolver el modelo *Tarea* en un tiempo reducido. Un ejemplo es el optimizador Coin-OR LP, [15]. Se probó este optimizador y se resolvió el modelo *Tarea* en el mismo orden de tiempo que CPLEX (escasos milisegundos).

La heurística propuesta proporcionó un alto número de soluciones óptimas (demostrado por coincidencia con la cota CI del tiempo de ciclo). Específicamente, se obtuvo un 38,08% de óptimos (correspondiente a 457 ejemplares entre los 1.200 propuestos). Para los ejemplares restantes, se aplicó el algoritmo de recocido simulado, obteniéndose una mejora promedio de 3,62% y un total de 435 nuevos óptimos, aumentando así el porcentaje de óptimos demostrados hasta un 74,33%.

En la Tabla I se muestra un resumen detallado de los resultados obtenidos. Los valores mostrados corresponden a los valores promedio obtenidos para los diez ejemplares de cada rango. La primera fila, por ejemplo, muestra los valores promedio de los diez ejemplares generados con un número de tareas N comprendido entre 50 y 200, y un número de estaciones m entre 5 y 10. Los resultados se han organizado en seis grupos correspondientes a los rangos crecientes de longitud de pieza $A_0 = \{11-15, 16-20, 21-25, 26-30, 31-35, 36-40\}$. La primera columna de la tabla (N) indica el rango del número de piezas. La segunda, (m) el rango del número de estaciones. La tercera, ($T(s)$ *heurística*) el tiempo computacional promedio de la heurística en segundos. La cuarta, ($\% \text{ óptimos } \overline{TC} = CI$) el porcentaje de ejemplares para los que la heurística encuentra una solución inicial coincidente con la cota inferior del tiempo de ciclo. La quinta, ($T(s)$ *SA*) el tiempo computacional promedio del recocido simulado en segundos. Finalmente, la sexta columna ($\% \text{ mejora } TC$), indica el porcentaje de mejora promedio en el valor del tiempo de ciclo del recocido simulado respecto al valor inicial obtenido mediante la heurística. Nótese que si los diez ejemplares de una combinación de rangos se resuelven óptimamente mediante la heurística, en ese caso no se aplica el recocido simulado, lo que se señala con un asterisco en la quinta y sexta columna.

El procedimiento propuesto resulta ser muy efectivo, ya que en un 38,08% de los casos se obtiene una solución óptima de forma directa en el *Paso 1*, y en un 36,25% se consigue obtener nuevas soluciones óptimas aplicando recocido simulado.

N	m	$A_0 = 11-15$				$A_0 = 16-20$				$A_0 = 21-25$			
		$T(s)$ heurística	% óptimos $\overline{TC}=CI$	$T(s)$ SA	% mejora TC	$T(s)$ heurística	% óptimos $\overline{TC}=CI$	$T(s)$ SA	% mejora TC	$T(s)$ heurística	% óptimos $\overline{TC}=CI$	$T(s)$ SA	% mejora TC
50-200	5-10	0,99	100	*	*	0,79	80	3.599	0,00	0,83	30	516	1,91
	11-20	0,98	90	0,16	1,08	0,77	60	1.801	1,20	1,04	10	1.602	4,68
	21-30	1,01	100	*	*	0,79	20	2.250	1,13	1,29	30	2.473	4,68
	31-40	1,02	70	3.599	0,21	0,93	30	3.085	0,81	1,20	10	3.599	2,95
201-400	5-10	1,00	100	*	*	0,92	100	*	*	1,00	60	900	0,76
	11-20	1,06	100	*	*	0,92	100	*	*	0,98	30	515	4,21
	21-30	1,00	100	*	*	0,85	70	3.599	0,34	0,99	40	1.206	3,42
	31-40	1,07	100	*	*	0,89	60	2.699	0,48	1,17	30	3.085	5,95
401-600	5-10	1,05	100	*	*	0,99	100	*	*	0,93	60	900	3,25
	11-20	1,07	100	*	*	0,98	90	3.599	0,68	1,06	40	601	2,08
	21-30	0,99	100	*	*	0,96	60	2.701	0,10	1,13	10	402	5,91
	31-40	1,08	100	*	*	0,96	90	3.599	1,65	1,12	0	1.081	2,81
601-800	5-10	1,27	100	*	*	1,07	100	*	*	1,07	50	720	0,72
	11-20	1,23	100	*	*	1,03	80	3.599	0,00	1,25	20	900	2,37
	21-30	1,20	100	*	*	1,01	80	3.599	0,26	1,30	20	451	3,47
	31-40	1,33	100	*	*	1,04	60	1.800	1,88	1,35	0	362	4,65
801-1000	5-10	1,30	100	*	*	1,17	100	*	*	1,07	40	1.800	1,18
	11-20	1,27	100	*	*	1,20	100	*	*	1,17	20	900	1,21
	21-30	1,26	100	*	*	1,10	80	3.599	0,00	1,19	20	1.350	1,82
	31-40	1,27	100	*	*	1,15	80	3.599	1,23	1,28	20	452	3,19
N	m	$A_0 = 26-30$				$A_0 = 31-35$				$A_0 = 36-40$			
		$T(s)$ heurística	% óptimos $\overline{TC}=CI$	$T(s)$ SA	% mejora TC	$T(s)$ heurística	% óptimos $\overline{TC}=CI$	$T(s)$ SA	% mejora TC	$T(s)$ heurística	% óptimos $\overline{TC}=CI$	$T(s)$ SA	% mejora TC
50-200	5-10	0,87	30	2.057	2,72	0,82	10	2.800	3,77	1,00	20	3.151	4,39
	11-20	1,09	10	3.199	1,97	1,06	0	3.599	5,02	1,22	0	3.599	3,98
	21-30	1,06	20	3.599	1,04	1,26	0	3.599	2,43	1,19	0	3.599	2,12
	31-40	1,17	0	3.599	1,43	1,31	0	3.599	3,13	1,42	0	3.239	3,29
201-400	5-10	0,87	60	901	1,39	0,89	10	1.601	4,31	0,95	10	2.400	4,52
	11-20	0,95	40	1.801	5,22	0,95	0	2.880	3,96	1,16	0	3.239	3,38
	21-30	0,85	10	2.799	3,37	1,14	0	2.882	2,95	1,20	0	3.599	5,16
	31-40	1,11	0	3.599	2,50	1,23	0	3.599	4,99	1,22	0	3.599	3,80
401-600	5-10	1,11	20	1.800	2,06	1,07	10	2.000	4,31	1,18	10	1.209	3,82
	11-20	1,01	20	2.250	2,96	0,99	10	3.199	5,17	1,23	0	2.885	5,69
	21-30	0,98	10	3.599	3,89	1,01	0	2.881	5,10	1,22	0	3.599	4,49
	31-40	0,93	0	3.239	3,26	1,18	0	3.599	7,39	1,30	0	3.599	4,68
601-800	5-10	1,24	60	0	1,70	1,21	20	901	2,54	1,19	0	2.160	3,78
	11-20	1,13	0	1.803	3,44	0,96	10	3.599	2,91	1,29	0	2.520	3,91
	21-30	1,20	20	2.701	4,63	1,10	0	3.243	4,45	1,31	0	2.880	5,96
	31-40	1,11	10	3.201	4,15	1,17	0	3.240	5,76	1,50	0	3.599	4,57
801-1000	5-10	1,25	40	1.200	2,10	1,20	0	1.440	3,22	1,30	0	1.081	3,44
	11-20	1,29	20	452	3,48	1,17	10	2.004	4,59	1,32	0	2.520	4,63
	21-30	1,32	20	2.249	2,32	1,30	0	2.162	6,19	1,33	0	3.240	4,56
	31-40	1,21	20	3.149	3,17	2,48	0	2.520	7,62	1,39	0	3.599	6,73

Tabla I. Resultados computacionales obtenidos para el procedimiento propuesto

5. CONCLUSIONES

En este trabajo se presenta un problema de equilibrado de líneas con ventanas de accesibilidad conocido como *Accessibility Windows Assembly Line Balancing Problem* (AWALBP), en el que las estaciones de la línea únicamente pueden acceder (y procesar) una porción limitada de las piezas en cada instante durante el proceso de montaje. Para resolver el AWALBP aquí tratado es necesario determinar un esquema de avance de las piezas en la línea y una asignación de tareas a etapas estacionarias, con el objetivo de minimizar el tiempo de ciclo. Se trata de un problema con escasas referencias en la literatura científica, a pesar de su amplia presencia, por ejemplo, en entornos automatizados de montaje de placas de circuitos impresos como los aquí descritos.

La resolución del AWALBP mediante métodos exactos puede requerir tiempos computacionales inasumibles para un entorno industrial, por lo que resulta conveniente recurrir a heurísticas. En este trabajo se propone un procedimiento de resolución compuesto por una heurística diseñada a medida, que proporciona una solución inicial, y un algoritmo de recorrido simulado para tratar de mejorar la solución inicial obtenida. Los pasos del procedimiento propuesto se han descrito con detalle para permitir que pueda ser reproducido con facilidad. Los resultados computacionales revelan que el método propuesto proporciona un alto porcentaje de soluciones óptimas en un tiempo computacional reducido.

Entre las ventajas del procedimiento propuesto, destacan las siguientes:

- Complejidad computacional relativamente baja en comparación con procedimientos exactos.
- Implementación sencilla, facilitada por una descripción detallada de los pasos de la heurística propuesta.
- Procedimiento flexible, al utilizar un algoritmo de recorrido simulado, reconocido como una de las técnicas más generales en optimización heurística, fácilmente aplicable a las características del problema tratado.
- Eficiencia, observada en un alto porcentaje de soluciones óptimas obtenidas en un tiempo computacional reducido.

La heurística propuesta, sin embargo, presenta el inconveniente de que no garantiza encontrar la solución óptima, aunque sí permite hallar soluciones razonablemente buenas.

Actualmente se están desarrollando nuevos procedimientos basados en la exploración de vecindarios para mejorar la solución de los ejemplares de mayor tamaño que, aún, no se resolvieron de forma óptima.

BIBLIOGRAFÍA

- [1] Baybars I. "A survey of exact algorithms for the simple assembly line balancing problem". *Management Science*. 1986. Vol. 32-8, p. 909-932. DOI: <http://dx.doi.org/10.1287/mnsc.32.8.909>.
- [2] Becker C, Scholl, A. "A survey on problems and methods in generalized assembly line balancing". *European Journal of*

Operational Research. 2006. Vol. 168-3, p. 694-715. DOI: <http://dx.doi.org/10.1016/j.ejor.2004.07.023>.

- [3] Boysen, N, Fliedner, M, Scholl A. "A classification of assembly line balancing problems". *European Journal of Operational Research*. 2007. Vol. 183-2 p. 674-693. DOI: <http://dx.doi.org/10.1016/j.ejor.2006.10.010>.
- [4] Burgos-Meneses, J, Burgos-Meneses, JM, Garzón-Aguirre, L, et al. "Algoritmo heurístico tipo Greedy para la solución de los problemas de líneas de ensamble". *DYNA Management*. 2013. Vol. 1-1. DOI: <http://dx.doi.org/10.6036/MN5714>.
- [5] Anónimo, Línea de montaje de placas de circuitos impresos con módulos robotizados. Disponible en web: <http://www.lewis-clark.com/product/assembleon-ax5-camera/> [consulta: 16 de abril de 2014].
- [6] Gaudlitz R. Optimization Algorithms for complex mounting machines in PC board manufacturing. Technical University of Darmstadt. Germany. Diploma Thesis, 2004.
- [7] Calleja, G, Corominas A, García-Villoria A, et al. "A MILP model for the Accessibility Windows Assembly Line Balancing Problem". *International Journal of Production Research*. 2013. Vol. 51-12, p. 3549-3560. DOI: <http://dx.doi.org/10.1080/00207543.2012.751514>.
- [8] García-Villoria A., Corominas A., Pastor R. "Còctel de heurísticas para resolver problemas difíciles". *DYNA*. 2012. Vol. 87-3, p.275-278. DOI: <http://dx.doi.org/10.6036/4508>.
- [9] Müller-Hannemann M, Weihe K. "Moving policies in cyclic assembly line scheduling". *Theoretical Computer Science*. 2006. Vol 351-3, p. 425-436. DOI: <http://dx.doi.org/10.1016/j.tcs.2005.10.023>.
- [10] Corominas A., Pastor R. "A MILP model for the Visibility Windows Assembly Line Balancing Problem: the case of Müller-Hannemann & Weihe problem". Working paper. Universitat Politècnica de Catalunya. Disponible en web: <http://upcommons.upc.edu/e-prints/bitstream/2117/7047/1/IOC-DT-P-2009-09.pdf>
- [11] Kirkpatrick, S, Gelatt CD, Vecchi, MP. "Optimization by Simulated Annealing". *Science*. 1983. Vol. 220-4598, p. 671-680. DOI: <http://dx.doi.org/10.1126/science.220.4598.671>.
- [12] Metaheuristics network. Disponible en web: <http://www.metaheuristics.net/index.php?main=1>, [consulta: 16 de abril de 2014].
- [13] García-Villoria, A, Pastor, R. "Simulated Annealing for improving the solution of the response time variability problem". *International Journal of Production Research*. 2003. Vol 54-16, p. 4911-4920. DOI: <http://dx.doi.org/10.1080/00207543.2013.775522>
- [14] Adenso-Díaz B, Laguna M. "Fine-tuning of algorithms using fractional experimental designs and local search". *Operations Research*. 2006. Vol 54-1, p-99-114. DOI: <http://dx.doi.org/10.1287/opre.1050.0243>.
- [15] Optimizador de código abierto COIN-OR-LP. Disponible en web: <http://www.neos-server.org/neos/solvers/lp:Clp/MPS.html> [consulta: 14 de abril de 2014].