Equilibrado de Líneas de Montaje mediante *Beam ACO*

Joaquín Bautista[1], Christian Blum[2], Jordi Pereira[3]

Resumen—**El presente artículo estudia la aplicación de la metaheurística *Beam ACO* al problema de Equilibrado de Líneas de Montaje SALBP-1. Tras introducir el problema, se presenta el algoritmo propuesto, resultado de la hibridación de algoritmos de colonias de hormigas con la heurística *Beam Search*. Los resultados de la experiencia computacional llevada a cabo muestran que el algoritmo propuesto es competitivo frente a los disponibles en la literatura, mejorando los resultados ofrecidos por las heurísticas anteriores.**

Palabras clave—**Equilibrado de Líneas de Montaje, Heurísticas Híbridas, *Beam Search*, Algoritmos de Hormigas.**

I. INTRODUCCIÓN

El problema de equilibrado de líneas de montaje, en inglés *Assembly Line Balancing Problem* (ALBP), consiste en el reparto de las tareas en las que se ha dividido el ensamblaje de un producto entre las m estaciones de trabajo que conforman la línea de montaje.

En general la obra en curso fluye por las estaciones de trabajo que se vinculan mediante un sistema de transporte que permite que la obra en curso se desplace de estación en estación.

La formulación clásica del problema cuenta con una división del conjunto V de n tareas elementales en que se ha dividido el montaje de una unidad de producto acabado, debiendo asignarse cada tarea j a una única estación. Cada tarea j tienen asociado un atributo vinculado con el tiempo de operación requerido, $t_j>0$, para su ejecución, que se ha determinado en función de las tecnologías de fabricación y de los recursos empleados. Adicionalmente al atributo temporal, la tecnología y la propia naturaleza del producto hacen que cada tarea j tenga un conjunto de tareas precedentes inmediatas, P_j, las cuales deben estar concluidas antes de que se pueda iniciar la operación j. Estas restricciones suelen representarse mediante un grafo acíclico $G(V,A)$ de precedencias cuyos vértices representan las tareas y cada arco dirigido (i,j) representa que la tarea i debe finalizarse antes de iniciar la tarea j en la línea

Una solución consiste en una asignación de un subconjunto de tareas S_k ($S_k \subseteq V$), denominado carga de la estación, a cada estación k ($1 \le k \le m$), respetando las relaciones de precedencia entre una pareja de tareas i,j, si $i \in S_h$ y $j \in S_k$, se debe cumplir que $h \le k$.

Dada una asignación de tareas, cada estación k presenta un tiempo de carga de estación $t(S_k)$ que es igual a la suma de las duraciones de las tareas asignadas a la estación k, $\sum_{j \in Sk} t_j$.

Tras un periodo inicial, la línea de montaje alcanza el régimen permanente de fabricación. A partir de entonces, las unidades de producto fluirán por la línea a cadencia constante, y cada estación k dispondrá de un tiempo c, denominado tiempo de ciclo, para realizar las tareas que tiene asignadas. El tiempo de ciclo no puede ser menor que el máximo tiempo de carga de estación: $c=\max_k\{t(S_k)\}$, ni debe ser mayor que la suma de las duraciones de las tareas de V: $c=\sum_k t(S_k)=t_{sum}$, y determinará la tasa de producción r de la línea ($r=1/c$), equivalente al número de unidades producidas por unidad de tiempo.

Una manera de medir la eficiencia de la línea está asociada al tiempo no productivo asociado a ella. Cada estación k presenta un tiempo muerto $I_k = c - t(S_k)$. La suma de dichos tiempos parciales da lugar al tiempo muerto total, $I_{sum} = \sum_k I_k = m \cdot c - t_{sum}$, que es el tiempo concedido a la línea que no se utiliza en operaciones de montaje.

En resumen, los problemas de equilibrado de líneas de montaje ALBP (Assembly Line Balancing Problem) están enfocados a agrupar de manera eficiente y coherente las tareas del conjunto V en las estaciones de trabajo. Se trata de 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. Según las condiciones establecidas y los diferentes objetivos planteados, aparecen diversas variantes del problema cuyo tratamiento debe enfocarse particularmente. En el presente trabajo se estudiará el problema clásico conocido como SALBP-1, en que se conoce a priori el tiempo de ciclo y se intenta minimiza el número de estaciones de trabajo para maximizar la eficiencia de la línea.

La exposición de la investigación desarrollada se ha organizado como sigue. La sección 2 presenta la literatura disponible sobre el problema partiendo de

[1]Nissan Chair, Escola Técnica Superior d'Enginyeria Industrial de Barcelona. Universitat Politècnica de Catalunya, Barcelona. joaquin.bautista@upc.edu
[2]ALBCOM, Dept. Llenguatges i Sistemas Informàtics. Universitat Politècnica de Catalunya, Barcelona. cblum@lsi.upc.edu
[3]Departament d'Organització d'Empreses. Universitat Politècnica de Catalunya, Barcelona. jorge.pereira@upc.edu

la clasificación inicial propuesta por Baybards [4]. La sección 3 presenta los procedimientos desarrollados para la resolución del problema. Primero se describe el uso de una técnica conocida como parametrización, para mejorar la calidad ofrecida por una regla heurística de prioridad en la construcción de soluciones iniciales. Posteriormente, se muestra el algoritmo *Beam ACO* [6] propuesto, que combina aspectos de los algoritmos de colonias de hormigas ACO [10], con la búsqueda *Beam Search*, [19]. Finalmente, en la sección 4 se presentan los resultados de una experiencia computacional para el problema y, en la sección 5 se ofrecen las conclusiones y las líneas futuras de investigación.

II. ESTADO DEL ARTE

Según la clasificación de Baybards [4], los problemas de equilibrado de líneas de montaje pueden dividirse en dos categorías principales: SALBP (problemas de equilibrado de líneas de montaje simples) y GALBP (problemas de equilibrado de líneas de montaje generales).

La familia SALBP presenta cuatro variantes: SALBP-1: minimizar el número de estaciones m dado un valor fijo del tiempo de ciclo c, que será la tratada en este trabajo; SALBP-2 minimizar el tiempo de ciclo c (maximizar la tasa de producción r) dado un número fijo de estaciones m; SALBP-E: minimizar simultáneamente c y m considerando su relación con el tiempo muerto total o la ineficiencia de la línea, con unos límites en el número máximo y mínimo de estaciones de la línea, $m_{min}=m=m_{max}$ y SALBP-F: dados m y c determinar la factibilidad del problema, y en caso afirmativo hallar una asignación válida.

Cuando se añaden otras consideraciones a las de la familia SALBP, los problemas se conocen en la literatura bajo la denominación GALBP (General Assembly Line Balancing Problem). Esta familia incluye a aquellos problemas con restricciones adicionales como por ejemplo la consideración de estaciones en paralelo, [8], [27], las agrupaciones forzadas de tareas, [9], las posibles incompatibilidades entre tareas, [1], diferencias entre estaciones, [18], y restricciones espaciales, [3], entre otras. Salvo contadas excepciones, todos los problemas industriales forman parte de esta última categoría, aunque la gran diversidad de características encontradas en entornos productivos, por una parte, y la semejanza de la definición básica del problema, tal como se ha mostrado en el apartado I del presente trabajo, hacen que el estudio de las formulaciones SALBP haya imperado en la literatura.

En cuanto a procedimientos de resolución, la literatura recoge una gran variedad de procedimientos, mayoritariamente basados en las formulaciones SALBP. Un primer grupo de algoritmos está formado por los denominados "greedy", basados en reglas de prioridad o procedimientos enumerativos parciales, véase [20], [11]. Un segundo grupo se compone por procedimientos enumerativos, principalmente bajo el esquema *branch and bound*, [14], [15], [22] y [26], siendo estos procedimientos los más efectivos para la familia SALBP. Un tercer grupo final está compuesto por aplicaciones de diversas metaheurísticas, véase [21] y [3]. La ventaja principal de esta gama de procedimientos son los buenos resultados que ofrecen y es la facilidad de adaptación a casos reales, hecho que dificulta el uso de los procedimientos enumerativos para la resolución de problemas prácticos, [5].

Las dos metaheurísticas comentadas utilizan una técnica indirecta para resolver instancias SALBP-1, consistente en la resolución iterativa de instancias SALBP-2, tal como se describe a continuación: dada una solución inicial con m estaciones de trabajo, se aplica la metaheurística propuesta con un número fijo de estaciones m-1 en busca de un tiempo de ciclo c' tan pequeño como sea posible. Si se encuentra una solución tal que $c'=c$, la solución es a su vez válida para el SALBP-1 con m-1 estaciones. Este proceso se aplica de forma iterativa hasta que se alcanza una condición de final. La aproximación adoptada se justifica por la dificultad para definir buenos vecindarios para la búsqueda local utilizando la formulación SALBP-1.

Otra característica que comparten ambos procedimientos y los procedimientos basados en búsqueda arborescente es la reversibilidad de las instancias. Dada una instancia SALBP-1, su instancia reversa es aquella que, manteniendo la duración de las tareas y el tiempo de ciclo, se obtiene por el cambio de sentido de todas las relaciones de precedencia. La solución obtenida puede transformarse en una solución a la instancia original intercambiando las tareas asociadas a cada estación, $S_k \leftarrow S_{m-k+1}$ ($1=k=m$). El uso de esta característica se fundamenta en que la modificación de las relaciones de precedencia puede disminuir la dificultad del problema, [22].

Un análisis más extenso de la bibliografía y de los procedimientos disponibles para la resolución de los problemas de equilibrado puede consultarse en [23] para la familia SALBP y en [5] para la familia GALBP.

La motivación del actual trabajo es doble. En primer lugar y, aunque los procedimientos enumerativos resultan especialmente efectivos para la resolución de los problemas SALBP, el interés en el desarrollo de heurísticas que ofrezcan buenos resultados para este tipo de problemas es grande, ya que los métodos de enumeración sólo son capaces de resolver la formulación SALBP e incluso las más pequeñas modificaciones en ella hacen que los procedimientos no sean aplicables, como es el caso

de la mayoría de problemas reales en que las condiciones del problema de equilibrado que se han descrito en la primera sección del presente trabajo se enriquecen con consideraciones especiales del sistema productivo concreto. Por otra parte, las heurísticas mas efectivas para la resolución del problema simple usan la estrategia explicada anteriormente, siendo nuestro objetivo en este trabajo el desarrollo de un procedimiento que pueda atacar la resolución del problema de forma directa.

El procedimiento propuesto en la siguiente sección es un híbrido entre los algoritmos de colonias de hormigas (ACO) [10] con la metaheurística *Beam Search* [19], un procedimiento de exploración arborescente desarrollado en su inicio para la resolución de problemas de secuenciación. El resultado, denominado *Beam ACO*, [6], se basa en la construcción paralela de diferentes soluciones, utilizando un procedimiento de acotación para limitar las soluciones que construye el algoritmo. El algoritmo se beneficia de la existencia de cotas eficientes y baratas, en términos de requerimientos computaciones, para el problema estudiado.

III. ALGORITMO PROPUESTO

En esta sección se presenta la implementación realizada de la metaheurística *Beam ACO* al problema SALBP-1. La primera parte se centra en la parametrización de la regla de prioridad que es la base de la metaheurística desarrollada. La segunda parte de esta sección estudia la metaheurística en sí.

A. Una regla de prioridad parametrizada

Las reglas heurísticas de prioridad son el método constructivo más frecuentemente utilizado para la resolución de problemas de secuenciación, como el equilibrado de líneas de montaje, ya sea por separado, en combinación o insertado en otros procedimientos. La propuesta *Beam ACO* presentada en este trabajo hace uso de una regla de este tipo. Esta subsección estudia una posible mejora de los resultados ofrecidos por la regla de prioridad escogida mediante la parametrización [12].

En el contexto del equilibrado de líneas de montaje, una heurística basada en reglas de prioridad inicia la construcción de una solución con la apertura de una primera estación (k=1), y procede asignando sucesivamente tareas a ella hasta que no se pueden asignar más tareas; en tal caso, se cierra dicha estación y se abre una estación nueva. En cada iteración, se asigna a la estación en curso la tarea candidata con mayor prioridad; siendo el conjunto de tareas candidatas aquellas tareas cuyas precedentes ya han sido asignadas y que requiere menos tiempo y espacio que bs disponibles en la estación en construcción. Cuando no se pueden asignar más tareas a la estación abierta, ésta se cierra y se abre la estación siguiente k+1; finalizando el procedimiento cuando no quedan más tareas por asignar.

El presente trabajo utiliza una regla que prioriza de forma conjunta la duración de una tarea y el número de sus tareas sucesoras (1).

$$h_j = \frac{t_j}{c} + \frac{\left|F_j^*\right|}{\max_{i \in V}\left|F_i^*\right|} \qquad (1)$$

donde F_j^* es el conjunto de todas las tareas sucesoras a la tarea j.

Aunque el uso de reglas de prioridad es muy popular, debido a su bajo coste computacional, y se han presentado muchas reglas alternativas, cada regla es únicamente capaz de ofrecer una solución a cada instancia del problema. Una manera de superar este inconveniente es mediante la parametrización de cada término que interviene en su cálculo (2).

$$h_j = k_1 \cdot \frac{t_j}{c} + k_2 \cdot \frac{\left|F_j^*\right|}{\max_{i \in V}\left|F_i^*\right|} \qquad (2)$$

donde k_1,$k_2 \in$ [-1,1]. Dada una instancia del problema, es plausible que diferentes valores de k_1 y k_2 ofrezcan diferentes soluciones, y por tanto un objetivo natural es buscar el juego de valores idóneo. A modo de ejemplo se muestra el resultado obtenido para diferentes valores de k_1 y k_2 en la figura 1. Se ha construido una malla donde mapear el resultado obtenido por la regla de prioridad para diferentes juegos parámetros. El eje x representa el valor adoptado por el parámetro k_1, mientras el eje y representa el valor del parámetro k_2. Los puntos en negro representan la obtención de la solución óptima para ese juego de parámetros. Como puede observarse, la combinación positiva de ambos parámetros llevan a la obtención de soluciones óptimas, aunque la regla (1) no la encontraría (punto k_1 =1, k_2 =1).

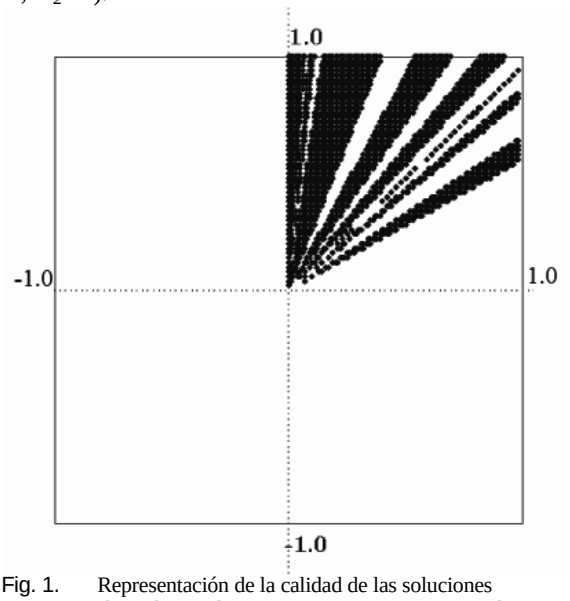

Fig. 1. Representación de la calidad de las soluciones obtenida para la instancia ARC83 con tiempo de ciclo 3786 para diferentes valores de parámetros

Para obtener el mejor juego de parámetros se ha optado por un procedimiento aleatorio (que se denominará RAND) para determinar dichos valores, tras probar la calidad ofrecida por un algoritmo de hormigas con codificación real, [25], y un procedimiento de optimización mediante cúmulos de partículas, Particle Swarm Optimization, [24], cuyos resultados no sobrepasaban la calidad obtenida por el enfoque que se presenta a continuación.

En cada iteración de RAND, se asigna un valor a cada parámetro en el rango de valores que puede adoptar de forma uniforme. Utilizando estos parámetros, se construyen dos soluciones a la instancia, la primera de la instancia original y la segunda de la instancia con relaciones de precedencia invertidas, transformando posteriormente su solución a la equivalente para la instancia original.

Para comprobar los resultados ofrecidos por RAND, se aplicó el algoritmo a las 269 instancias de Scholl [22].

A continuación se comparan los resultados ofrecidos por la regla parametrizada y por la regla original. En la tabla I pueden observase los resultados ofrecidos por ambos algoritmos. Para cada algoritmo, se cita el número de soluciones óptimas encontradas, segunda columna, el error medio, tercera columna, indicado el promedio de estaciones suplementarias que necesita la mejor solución encontrada por cada algoritmo, y en la cuarta columna, aparece el tiempo medio de ejecución necesario para encontrar la mejor solución.

TABLA I
RESULTADOS DE LA APLICACIÓN DE LA REGLA DE PRIORIDAD Y DEL ALGORITMO RAND

	#opt	Error	T (s)
Prioridad	164	0.41	0.000017
RAND	200	0.26	0.0019

Los resultados demuestran que incrementando en una fracción de segundo el tiempo de computación, el algoritmo RAND es capaz de encontrar juegos de parámetros que permiten resolver 36 instancias adicionales que utilizando la regla de prioridad original. El algoritmo propuesto, por tanto, se iniciará con la búsqueda de unos parámetros adecuados para la instancia que se pretende resolver.

B. La metaheurística Beam ACO para el SALBP-1

El algoritmo *Beam ACO* propuesto para la resolución del problema SALBP-1 se describe a continuación.

Inicialmente se determina mediante el algoritmo RAND un valor idóneo para los parámetros, por un tiempo igual a 0.5 segundos. La regla de prioridad obtenida se utiliza para determinar la mejor solución conocida, s_{bsf}, y la información heurística del algoritmo mediante (3), (4) y (5). Nótese que debido a que los valores pueden ser negativos, es necesario normalizar dichos valores para que siempre se encuentren en el intervalo de valores reales positivos.

$$h_{\min} = \min\{h_j \mid j = 1,...,n\} \quad (3)$$

$$h_{\max} = \max\{h_j \mid j = 1,...,n\} \quad (4)$$

$$h_j^{aco} = \frac{n_j - n_{\min} + 1}{h_{\max}};\ j = 1,...,n \quad (5)$$

Posteriormente se inicializan los valores de feromona, t_{jk}=0.5, entre tareas (j=1,…,n) y estaciones (k=1,…,m).

Aprovechando la revesibilidad de la instancia, el procedimiento intenta resolver la instancia directa y reversa paralelamente. En cada iteración del algoritmo, a_o hormigas construyen soluciones para el problema original, mientras que a_r hormigas lo hacen para el problema reverso. Cualquier solución obtenida para la instancia inversa es subsecuentemente convertida en una solución de la instancia original.

El procedimiento constructivo que utiliza cada una de las a_o y a_r hormigas se describe a continuación. Cada hormiga intenta construir asignaciones de tareas a las estaciones abiertas mediante la asignación de tareas que cumplan con las reglas de precedencia y limitaciones de tiempo de ciclo.

Al contrario que en el procedimiento basado en reglas de prioridad y otras propuestas anteriormente recogidas en la literatura, cada hormiga intenta determinar una asignación para la estación en curso de k_{ext} maneras diferentes, cuyo valor ha sido fijado a 50 en los experimentos posteriores.

La asignación de tareas a cada una de las k_{ext} diferentes se realiza tarea a tarea. La tarea que se escoge entre las candidatas seguirá un comportamiento determinista o probabilista.

El conjunto de tareas candidatas, T^c, se determina como todas aquellas tareas tal que saturen la estación, su tiempo disponible en la estación en curso sea igual al tiempo ejecución de la tarea, o en caso de no existir ninguna tarea de este tipo, el conjunto de todas las tareas candidatas, por reglas de precedencia y tiempo disponible en la estación en curso.

Cuando se escoge la tarea siguiendo un comportamiento determinista, se escoge la tarea que ofrece un valor máximo de P_j obtenido mediante (6).

$$P_j = \frac{\left(\sum_{i=1}^{k} t_{ji}\right) \cdot h_j^{aco}}{\sum_{l \in T^c}\left(\sum_{i=1}^{k} t_{li}\right) \cdot h_l^{aco}} \quad (6)$$

Esta fórmula utiliza la regla introducida por Merkle y Middendorf [17] en el contexto de los problemas de secuenciación y que ya había sido utilizada para el presente problema en anteriores implementaciones de algoritmos de colonias de hormigas para problemas de equilibrado [2]. Se basa en asignar las tareas a la estación actual no sólo teniendo en cuenta la conveniencia de asignar la tarea a la estación en construcción, sino que adicionalmente se tienen en cuenta aquellas estaciones a las que ya no se puede asignar la tarea.

En caso de que la elección sea de tipo probabilista, la elección se determina aleatoriamente con probabilidad P_j de seleccionar cada tarea, según marca la fórmula (6).

Cuando se dispone de k_{ext} asignaciones para la estación en curso, se aplica una cota inferior para determinar la solución parcial más prometedora, en este caso obtenida mediante (7) donde T es el conjunto de tareas no asignadas en la solución parcial en curso.

Entre todas ellas se selecciona la mejor y se pasa a construir una nueva estación, hasta encontrar una asignación completa. En caso que el valor de cota obtenido fuera superior al de la mejor solución conocida, el proceso de construcción se aborta.

$$LB = \frac{\sum_{i \in T} t_i}{c} \qquad (7)$$

Cabe citar que el procedimiento de selección coincide con el de una búsqueda *Beam* con ancho de ventana igual a 1.

Adicionalmente al proceso constructivo anteriormente citado, se realizan dos operaciones, una encaminada a intensificar la búsqueda, consistente en actualizar la información ofrecida por las feromonas, y otra a diversificarla, asociada a determinar el estancamiento de la búsqueda y su reinicio.

La actualización de feromonas utiliza la mejor solución obtenida por la iteración en curso, o en caso de no haber obtenido ninguna solución al abortar todas las búsquedas por la cota, la mejor solución conocida. Para toda estación, k, y tarea, j, el valor de feromona t_{jk} se actualiza mediante (8) y (9).

$$t_{jk} = \min\left\{\max\left\{t_{\min}, t'\right\}, t_{\max}\right\} \qquad (8)$$

$$t' = t_{jk} + r \cdot \left(d_{jk} - t_{jk}\right) \qquad (9)$$

Donde r=0.1, t_{min}=0.01, t_{max}=0.99 y d_{jk}=1 si la tarea j se ha asignado a la estación k en la mejor solución encontrada, y 0 en caso contrario.

Por su parte para determinar el estancamiento de la búsqueda, se utiliza un factor de convergencia según (10) y (11). Si este valor alcanza un valor menor que 0.05, se reinician los valores de la matriz de feromonas.

$$cf = 2 \cdot \left(\frac{\sum_{j=1}^{n} \sum_{k=1}^{|S_{bsf}|} t^{*}_{jk}}{n \cdot |S_{bsf}| \cdot (t_{\max} - t_{\min})} \right) \qquad (10)$$

$$t^{*}_{jk} = \min\left\{t_{\max} - t_{jk}; t_{jk} - t_{\min}\right\} \qquad (11)$$

Puede encontrarse una descripción más detallada del algoritmo presentado en [7] bajo el contexto de la resolución de problemas de equilibrado con restricciones adicionales de espacio para componentes y utillaje, [3].

IV. EXPERIENCIA COMPUTACIONAL

Los algoritmos propuestos en la sección anterior se han implementado en ANSI C++ y se han compilado usando el programa GCC 3.2.2. Los resultados de la experiencia computacional se han obtenido en un ordenador con procesador Pentium 4 (3.06 Ghz.) y 1 Gb. de memoria.

A continuación se muestran los resultados obtenidos para la colección de instancias de Scholl disponible en la página web del mismo autor, http://www.assembly-line-balancing.de. Para cada instancia se han realizado 10 ejecuciones independientes del algoritmo, reportando la mejor solución encontrada y se ha impuesto un límite de 120 segundos por instancia, deteniendo la búsqueda si se ha encontrado una solución óptima.

La tabla 2 resume los resultados obtenidos para las 269 instancias de la colección de instancias y los compara con SALOME, [22], que es un procedimiento de enumeración implícita bidireccional, los procedimientos de búsqueda local PrioTabu y EurTabu, [21], que se distinguen por el uso de una regla de prioridad o el procedimiento enumerativo Eureka, [14], para construir la primera solución, el algoritmo genético híbrido HGA [13] y un algoritmo de hormigas anteriormente [3] propuesto por dos de los autores del presente trabajo. Todos estos procedimientos pueden considerarse como el estado del arte en cuanto a la resolución de problemas de equilibrado de líneas de montaje desde diferentes aproximaciones heurísticas y exactas. Para cada algoritmo se reporta el número de soluciones óptimas obtenidas, *#opt*, y el tiempo de ejecución medio por instancia en segundos. Indicar que para dos de las instancias de la colección no se conoce la solución óptima; se utiliza la mejor solución conocida. Por otra parte los tiempos de ejecución no son directamente comparables debido a los diferentes ordenadores utilizados y condiciones de finalización que tienen en cuenta, aunque se resalta el tiempo utilizado por Beam-ACO es lo suficientemente pequeño para su utilización en circunstancias reales.

TABLA II
RESULTADOS OBTENIDOS POR BEAM-ACO EN COMPARACIÓN CON OTROS PROCEDIMIENTOS DISPONIBLES EN LA LITERATURA

Algoritmo	#opt	Tiempo (s)
SALOME [22]	227	98.9
PrioTabu [21]	200	101.8
EurTabu [21]	214	62.6
ANTS [3]	227	13.84
HGA [13]	214	N.d.
Beam-ACO	245	1.92

Podemos observar que la propuesta presentada es capaz de resolver más instancias que los anteriores algoritmos disponibles. En particular, obtiene mejores resultados que la propuesta anterior basada en colonias de hormigas.

TABLA III
RESULTADOS OBTENIDOS PARA LAS INSTANCIAS SCHOLL. SE MUESTRA EL TIEMPO DE CICLO, C, LA MEJOR SOLUCIÓN CONOCIDA, BKS, Y LOS RESULTADOS OFRECIDOS POR EL ALGORITMO DE HORMIGAS [3], LA BÚSQUEDA TABÚ [16] Y EL ALGORITMO BEAM-ACO PRESENTADO EN EL ARTÍCULO.

c	bks	ANTS	TABU	Beam-ACO
1394	50	52	51	51
1422	50	51	**50**	**50**
1452	48	50	49	49
1483	47	49	48	48
1515	46	48	47	47
1548	46	**46**	**46**	**46**
1584	44	46	45	45
1620	44	**44**	44	44
1659	42	44	43	43
1699	42	**42**	**42**	**42**
1742	40	41	41	41
1787	39	40	40	40
1834	38	39	39	39
1883	37	38	38	38
1935	36	37	37	37
1991	35	37	36	**35**
2049	34	35	35	35
2111	33	34	34	34
2177	32	33	33	**32**
2247	31	32	32	32
2322	30	31	31	31
2402	29	30	30	30
2488	28	29	29	29
2580	27	28	28	**27**
2680	26	27	27	**26**
2787	25	26	26	**25**

Adicionalmente, la tabla III muestra los resultados obtenidos para las 26 instancias basadas en el grafo de precedencias SCHOLL y que resultan las más complejas disponibles actualmente en la literatura. Los resultados muestran que Beam-ACO puede resolver más instancias de forma óptima, 9 de 26, que las dos propuestas metaheurísticas más recientes.

V. CONCLUSIONES Y LÍNEAS FUTURAS

En este trabajo se ha propuesto un algoritmo de hormigas híbrido para la resolución del SALBP-1. El algoritmo propuesto se obtiene de la hibridación de los algoritmos ACO con Beam-Search. La nueva propuesta puede afrontar la resolución del SALBP-1 de forma directa, al contrario que otras aproximaciones anteriores. Los resultados muestran que la implementación resultante forma parte del estado del arte para la resolución del problema.

En el futuro se plantea analizar la importancia de cada componente algorítmico en el resultado global de la heurística, la consideración de diferentes parámetros y la extensión a otros problemas de equilibrado de líneas de montaje.

AGRADECIMIENTOS

El presente trabajo ha sido parcialmente financiado por el proyecto DPI2004-03475 del gobierno español. También agradecemos a Nissan Spanish Industrial Operations y a la Cátedra Nissan UPC el apoyo dado a este trabajo.

REFERENCIAS

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

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

[3] J. Bautista, J. Pereira, Ant Algorithms for a Time and Space constrained Assembly Line Balancing Problem, European Journal of Operational Research, to appear. doi:10.1016/j.ejor.2005.12.017

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

[5] C. Becker, A. Scholl, A survey on problems and methods in generalized assembly line balancing, European Journal of Operational Research, 168(3):694-715, 2006

[6] C. Blum, Beam-ACO – Hybridizing ant colony optimization with beam search: An application to open shop scheduling, Computers & Operations Research 32(6):1565-1591, 2005.

[7] C.Blum, J. Bautista, J. Pereira, Beam-ACO applied to Assembly Line Balancing. Lecture Notes in Computer Science, 4150: 96-107, 2006

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

[9] R.F. Deckro, Balancing Cycle Time and Workstations, IIE Transactions 21:106-111, 1989.

[10] M. Dorigo, T. Stützle. ,nt Colony Optimization. MIT Press, Cambridge, MA, 2004.

[11] K. Fleszar, K.S. Hindi, An enumerative heuristic and reduction methods for the assembly line balancing problem, European Journal of Operational Research 145(3):606-620, 2003

[12] A. Fügenschuh, Parametrized greedy heuristics in theory and practice. Lecture Notes in Computer Science 3636, 21-31. 2005

[13]J.F. Gonçalves, J.R de Almeida, A hybrid genetic algorithm for assembly line balancing. Journal of Heuristics, 8:629-642, 2002.
[14]T.R. Hoffmann, Eureka. A hybrid system for assembly line balancing, Management Science 38(1):39-47, 1992.
[15]R.V. Johnson, Optimally balancing assembly lines with "FABLE", Management Science 34:240-253, 1988.
[16]D.L. Lapierre, A. Ruiz, P. Soriano, Balancing Assembly Lines with tabu search. European Journal of Operational Research, 168:826-837, 2006
[17]D. Merkle, M. Middedorf, H. Schmeck, Ant Colony Optimization for Resource Constrained Project Scheduling. GECCO-2000, 2000.
[18]G. Nicosia, D. Pacciarelli, A. Pacifici, Optimally balancing assembly lines with different workstations, Discrete Applied Mathematics 118:99-113, 2002
[19]P.S. Ow, T.E. Morton, Filtered beam search in scheduling. International Journal of Production Research, 26:297-307, 1988
[20]F.B.Talbot, J.H. Patterson, W.V. Gehrlein, A comparative evaluation of Heuristic Line Balancing Techniques, Management Science 32:430-454, 1986.
[21]A. Scholl, S. Voss, Simple assembly line balancing – Heuristic approaches, Journal of Heuristics 2:217-244, 1996.
[22]A.Scholl, R. Klein, Balancing Assembly lines effectively – A computational comparison, European Journal of Operational Research 114:50-58, 1999.
[23]A. Scholl, C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 168:666-693, 2006.
[24]Y. H. Shi, R.C. Eberhart, A modified particle swarm optimizer. In the proceedings of the IEEE International Conference on Evolutionary Computation, 66-73, IEEE Press 1988
[25]K. Socha, ACO for continuous and mixed-variable optimization. Lecture Notes in Computer Science, 3172:25-36, 2004.
[26]A. Sprencher, Dynamic search tree decomposition for balancing assembly lines by parallel search, International Journal of Production Research 41(7):1413-1430, 2003
[27]P.M. Vilarinho, A.S. Simaria, A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations, International Journal of Production Research 40(6):405-1420, 2002.