UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Equilibrado de líneas de montaje en paralelo con estaciones multilínea y dimensionado de buffers

Harry Nick Aguilar Gamarra

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Programa de Doctorado en Cadena de Suministro y Dirección de Operaciones

Equilibrado de líneas de montaje en paralelo con estaciones multilínea y dimensionado de buffers

Tesis doctoral realizada por:

Harry Nick Aguilar Gamarra

Dirigida por:

Dr. Alberto García Villoria

Dr. Rafael Pastor Moreno

Instituto de Organización y Control de Sistemas Industriales,

Departamento de Organización de Empresas

Barcelona, Julio 2022

Resumen

Las líneas de montaje son sistemas de producción en masa las cuales tienen una relevancia en la fabricación de productos estándar y personalizados. Uno de los problemas de optimización más elementales en este ámbito es el problema de equilibrado de líneas de montaje (*Assembly Line Balancing Problem* (ALBP)). EL ALBP consiste en asignar un conjunto de tareas a una serie de estaciones, satisfaciendo a su vez una serie de restricciones, de tal forma que uno o más objetivos son optimizados.

Actualmente es común que las empresas dedicadas a la producción dispongan de más de una línea de montaje o líneas múltiples, ya sea para hacer frente a fluctuaciones en la demanda, por motivos de diseño, agrupar productos en diferentes líneas, etc. El uso de líneas múltiples, como por ejemplo las líneas ubicadas en paralelo, han suscitado el interés de los investigadores en los últimos años. Las líneas de montaje en paralelo son dos o más líneas que, si están ubicadas suficientemente cerca, pueden ser equilibradas conjuntamente utilizando estaciones compartidas (estaciones multilínea) entre líneas adyacentes. En una estación multilínea, el operario realiza las tareas asignadas a su estación de dos líneas adyacentes, en el tiempo de ciclo de cada línea.

Los problemas de equilibrado de líneas de montaje en paralelo *(Parallel Assembly Lines Balancing Problem* (PALBP)) pueden presentar líneas con tiempos de ciclo diferentes. Como consecuencia de trabajar con tiempos de ciclo diferentes, los sistemas con estaciones multilínea pueden tener que producir en lotes. En este caso, puede ser necesario el uso de buffers en dichas estaciones. La revisión del estado del arte del PALBP revela que no hay ningún estudio que manifieste el uso de algún tipo de almacenaje (o buffers) en el caso de producir en lotes. Y cabe mencionar que no considerar la necesidad (existencia y dimensionado) de buffers en el equilibrado de líneas de montaje en paralelo con estaciones multilínea y líneas con tiempos de ciclo diferentes, puede llevar a diseñar soluciones académicas no implementables en la industria.

Esta tesis doctoral trata el problema de equilibrado de líneas de montaje en paralelo y dimensionado de buffers (PALBP-B). En específico, el PALBP-B consiste en un sistema PALB con estaciones multilínea y líneas con tiempos de ciclo diferentes, en el cual existe la posibilidad de necesitar buffers. El problema de estudio en esta tesis se centra en el PALBP-B de líneas rectas y modelo único.

La resolución del PALBP-B implica solucionar dos (sub)problemas que, idealmente, se deberían resolver de manera simultánea: 1) equilibrado de líneas: asignación de las

tareas a las estaciones (y su definición como estación regular o multilínea); y 2) dimensionado del buffer en las estaciones multilínea.

La revisión del estado del arte manifiesta que el problema presentado en esta tesis no ha sido estudiado en la literatura, lo que conlleva a definir y formalizar el PALBP-B. En primer lugar, se caracteriza y se define el (sub)problema de dimensionado de buffers, y se presenta un modelo de programación lineal entera mixta (PLEM) y dos modificaciones respecto al modelo original, para su resolución óptima. Además, también se presentan procedimientos no exactos (heurísticos y metaheurístico) para resolver ejemplares de un tamaño realista. En segundo lugar, se presentan procedimientos no exactos (heurísticos y metaheurísticos) con el fin de resolver el PALBP-B, es decir, el (sub)problema de equilibrado de líneas y el (sub)problema de dimensionado de buffers (utilizando los procedimientos desarrollados anteriormente para este fin) de manera conjunta.

La evaluación de los procedimientos desarrollados para resolver el (sub)problema de dimensionado de buffers se realiza mediante un conjunto de ejemplares generados para este fin. Los procedimientos para resolver el PALBP-B se evalúan mediante ejemplares realistas extraídos de la literatura.

Abstract

Assembly lines are mass production systems which are relevant in the manufacture of standard and customized products. One of the most elementary optimization problems in this field is the Assembly Line Balancing Problem (ALBP). ALBP consists of assigning a set of tasks to a set of ordered stations, satisfying some specific constraints, in such a way that one or more objectives are optimized.

Nowadays, companies dedicated to production usually have more than one assembly line or multiple lines to face demand fluctuations, design reasons, group products in different lines, etc. The use of multiple lines located in parallel has attracted the interest of researchers in the last years. Parallel assembly lines are two or more lines that, if they are close enough, can be balanced together using shared stations (multi-line stations) between adjacent lines. In a multi-line station, the operator performs tasks assigned to his/her station of two adjacent lines in the cycle time of each line.

Parallel Assembly Lines Balancing Problem (PALBP) can have lines with different cycle times. As a consequence of working with different cycle times, the systems with multi-line stations may have to produce in batches. In that case, the use of buffers in multi-line stations may be needed. The review of the state-of-the-art of PALBP reveals that no studies consider the use of some type of buffers in the case of production in batches. It should be noted that not considering the need (existence and sizing) of buffers in the PALB with lines with different cycle times can lead to the design of academic solutions that cannot be implemented in the industry.

This doctoral thesis deals with the parallel assembly lines balancing problem and buffer sizing (PALBP-B). Precisely, PALBP-B consists of a PALB system with multi-line stations and lines with different cycle times, in which there is the possibility of needing buffers. The problem of study in this thesis focuses on the PALBP-B of straight lines and single model.

The PALBP-B resolution consists of two (sub)problems that should be solved simultaneously: 1) line balancing: assigning tasks to stations (and its definition as regular or multi-line stations); and 2) buffers sizing in the multi-line stations.

The review of the state-of-the-art reveals that the problem presented in this thesis has not been studied in the literature. This implies defining and formalising the PALBP-B. First, the buffer sizing (sub)problem is characterized and defined, and a mixed-integer linear programming (MILP) model and two modifications derived from it are presented

for its optimal resolution. In addition, non-exact procedures (heuristics and metaheuristic) are also presented to solve real-size instances. Secondly, non-exact procedures (heuristics and metaheuristics) are presented in order to solve the PALBP-B, that is, the line balancing (sub)problem and the buffer sizing (sub)problem (using the procedures developed previously for this purpose) together.

All the methods developed to solve the buffer sizing (sub)problem are evaluated via a computational experiment based on a set of instances generated for this purpose. The methods developed to solve the PALBP-B are evaluated via an exhaustive computational experiment based on a set of realistic instances from the literature.

Agradecimientos

En primer lugar, me gustaría dar mi más sincero agradecimiento a mis directores de tesis, el Dr. Alberto García Villoria y el Dr. Rafael Pastor Moreno, por sus valiosos comentarios, consejos y correcciones realizadas durante el desarrollado de la tesis.

También quiero agradecer al Instituto de Organización y Control de Sistemas Industriales (IOC) y a la Universitat Politècnica de Catalunya (UPC) por su ayuda en varias cuestiones técnicas y administrativas.

Finalmente, quiero dar las gracias a mi familia. A mi madre Gladis y a mi padre Lino por el apoyo incalculable que me han dado durante tanto tiempo y por la confianza que han depositado en mí. A mi esposa Rubí, por sus consejos, apoyo y comprensión durante la realización del doctorado.

Tabla de contenido

Listado de ilustraciones

Listado de tablas

Capítulo 1. Introducción

1.1 Introducción al problema de equilibrado de líneas de montaje (ALBP)

En el ámbito de la producción, un problema de optimización combinatorio habitualmente tratado es el equilibrado de líneas de montaje. Una línea de montaje consiste en un número finito de estaciones ubicadas en serie a lo largo de una cinta transportadora u otro sistema de transporte similar, donde una unidad de producto es transportada entre estaciones consecutivas. En cada estación, un cierto número de tareas son realizadas y, transcurrido un tiempo prefijado (tiempo de ciclo), la unidad es transportada a la siguiente estación.

El problema de equilibrado de líneas de montaje (*Assembly Line Balancing Problem* (ALBP)) consiste en asignar las tareas a realizar a las diferentes estaciones, de tal forma que uno o más objetivos son optimizados (Helgeson et al., 1954). Habitualmente, la asignación de tareas a las estaciones se realiza satisfaciendo unas restricciones impuestas, como pueden ser: relación de precedencias entre tareas, tiempo de ciclo, incompatibilidad entre tareas, etc.

Es sabido que el ALBP es un problema de optimización combinatoria complejo, de naturaleza NP-difícil (Battaïa and Dolgui, 2013). En la práctica, esto significa que el tiempo de cálculo para garantizar una solución óptima con un procedimiento exacto puede crecer exponencialmente con el tamaño del problema.

Desde que Salveson (1955) realizara la primera formulación matemática del ALBP, varios investigadores han presentado diferentes estudios en este campo, donde se han propuesto una gran variedad de procedimientos de resolución exactos y no exactos, así como diferentes tipos de restricciones que pretenden acercar el ALBP a los problemas reales de la industria. Los diferentes trabajos presentados en la literatura del ALBP han sido recopilados y estudiados en extensas revisiones del estado del arte, como por ejemplo: Ghosh and Gagnon (1989); Erel and Sarin (1998); Becker and Scholl (2006); Boysen et al. (2007), (2008); Battaïa and Dolgui (2013); Sivasankaran and Shahabudeen (2014); Boysen et al. (2021); entre otros.

Las líneas de montaje se pueden clasificar considerando diferentes aspectos y características de éstas. Una clasificación común es considerar la forma de la línea (o *layout*), o también, la similitud de los productos procesados en una línea. No obstante, existen más formas de clasificar los ALBP, que los aquí expuestos.

En la Figura 1 se representan los tipos de *layouts* más utilizados comúnmente, donde las estaciones se representan en recuadros dorados y disponen de un operario.

- Líneas rectas (Figura 1a). Un conjunto de estaciones ubicadas a lo largo de una cinta o medio transportador.
- Líneas de dos lados (Figura 1b), Bartholdi (1993). Las líneas de dos lados son operadas por ambos lados de la línea (lado izquierdo y/o derecho), en el cual, un par de estaciones opuestas procesan una pieza simultáneamente. En este tipo de líneas cabe la posibilidad de que algunas tareas puedan ser asignadas a cualquiera de los dos lados, o que solo puedan ser asignadas a un lado (izquierda o derecha) de la línea.
- Líneas en forma de U (Figura 1c), Miltenburg and Wijngaard (1994). En este tipo de líneas en forma de U, la entrada y la salida están ubicadas relativamente cerca. Los operarios de ambas partes (entrada y salida) de la línea pueden desplazarse de una a otra parte, por lo tanto, pueden realizar el trabajo de ambos lados en el mismo tiempo de ciclo.

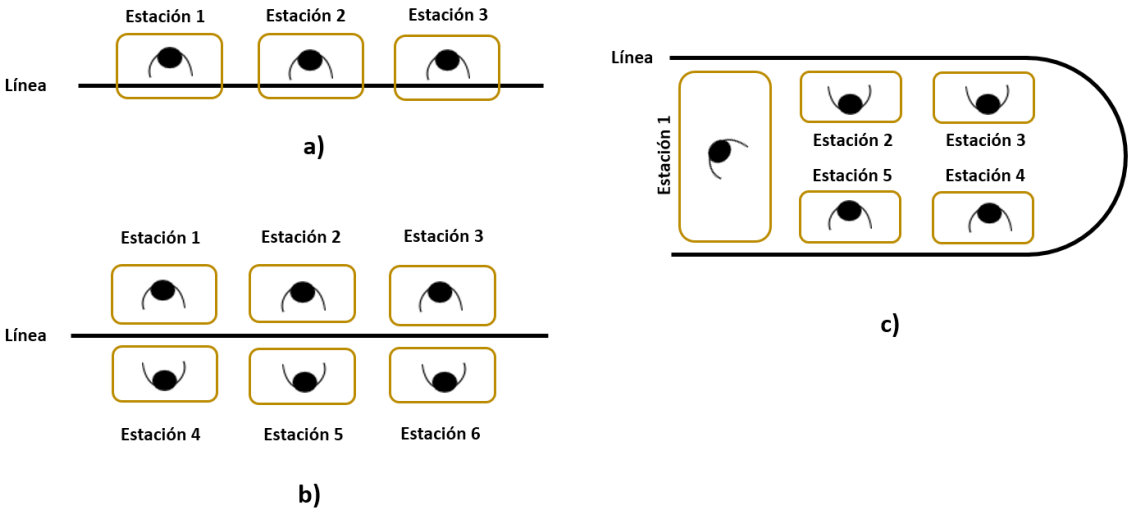

Figura 1. Tipos de *layout*: a) líneas rectas; b) líneas de dos lados; c) líneas en forma de U.

Como se ha introducido anteriormente, otra forma común de clasificar los ALBP es dependiendo de la similitud de los modelos (o productos) procesados en la línea. Se pueden distinguir: líneas de modelo único, un único producto es ensamblado en la línea; líneas de modelos mixtos, productos similares son ensamblados en la misma línea, por lo que los tiempos de configuración o de *setup* son negligibles; líneas multi-modelo, diferentes productos con diferentes procesos de producción son ensamblados en la misma línea, por lo que se requiere de un tiempo de *setup* entre un producto y otro.

Baybars (1986) propuso clasificar los ALBP en dos grupos, diferenciando entre el *simple* ALBP (SALBP) y el *general* ALBP (GALBP). El SALBP es el caso más simple del ALBP. Este problema se reduce a una línea de montaje recta y modelo único, donde, usualmente, solo se consideran restricciones de precedencia y tiempo de ciclo. Por otra parte, el GALBP incluye características y aspectos que aparecen en los problemas reales: tareas en paralelo (Pinto et al., 1975), estaciones en paralelo (Buxey, 1974), líneas en forma de U (Miltenburg and Wijngaard, 1994), líneas múltiples (Miltenburg, 1998), subgrafo de montaje alternativo (Capacho et al., 2009), tiempo de *setup* entre tareas (Martino and Pastor, 2010), restricciones de recursos (Corominas et al., 2011), minimización jerárquica de las cargas de trabajo de las estaciones (Pastor et al., 2012), estaciones con ventanas de accesibilidad (Calleja et al., 2013), entre otros.

Actualmente es común que las empresas que se dedican a la producción de bienes materiales oferten más de un producto o variante de éste. En estos casos, es posible que el sistema de producción o montaje esté compuesto por más de una línea, convirtiendo así el sistema productivo en un sistema de líneas múltiples.

En la literatura de las líneas múltiples, autores como Miltenburg (1998) y Sparling (1998) introdujeron el uso de estaciones multilínea, es decir, estaciones compartidas por más de una línea de montaje, donde un operario realiza las tareas asignadas a su estación, de dos o más líneas adyacentes. El uso de estaciones multilínea puede ayudar a reducir el número de recursos necesarios, por ejemplo, operarios.

Cabe destacar que el uso de estaciones multilínea es únicamente posible si las líneas están suficientemente cerca (son adyacentes), de esta forma se evita que el recorrido que deba realizar el operario de una línea a la otra no genere un aumento considerable en el tiempo de inactividad de la estación. Considerando el concepto de estación multilínea, Gökçen et al. (2006) propusieron equilibrar dos o más líneas ubicadas en paralelo considerando la posibilidad de utilizar estaciones multilínea, introduciendo así un nuevo problema en la literatura del ALB, conocido como el problema de equilibrado de líneas de montaje en paralelo (*Parallel Assembly Lines Balancing Problem* (PALBP)).

Las líneas múltiples, y por consecuencia las PALB, proporcionan ciertas ventajas respecto al uso de líneas individuales (sin estaciones multilínea) (Lusa, 2008; Aguilar et al., 2020): 1) puede reducir el número de operarios y estaciones requeridas; 2) reduce la sensibilidad a los fallos: al tener más de una línea, en caso de fallo de una de ellas, la producción de ésta se puede derivar a otra línea; 3) el uso del sistema PALB en el caso de líneas heterogéneas (diferentes productos, tiempos de ciclo, etc.), puede ayudar

a mejorar la eficiencia del sistema. Sin embargo, el uso de este tipo de líneas puede incrementar el coste de inversión en comparación a los sistemas de líneas individuales.

1.2 Introducción al problema de equilibrado de líneas de montaje en paralelo (PALBP)

Podemos definir el PALB como dos o más líneas ubicadas en paralelo, donde un operario puede realizar las tareas asignadas a su estación multilínea de dos líneas adyacentes.

En el PALBP es usual asumir las siguientes premisas (Gökçen et al., 2006; Özcan et al., 2009; Baykasoğlu et al., 2009; entre otros):

1) Dos o más líneas de montaje están dispuestas en paralelo.
2) El grafo de precedencias de cada producto es conocido.
3) Los tiempos de las tareas son conocidos.
4) Cada tarea es asignada solo a una estación.
5) Solo se permite una unidad de producto en el puesto de trabajo de un operario cada tiempo de ciclo de la línea.

En la Figura 2 se representa un sistema PALB formado por tres líneas en paralelo (L1, L2 y L3), en el cual se pueden observar las estaciones, puestos de trabajo y etapas. Una estación es un espacio físico que puede cubrir uno (estación regular) o dos (estación multilínea) puestos de trabajos. Un puesto de trabajo es el lugar físico donde se realiza un conjunto de tareas de una línea. Las etapas son divisiones que sirven para representar la longitud total de las líneas. La Figura 2 muestra un ejemplo con 11 puestos de trabajo, 9 estaciones (de las cuales, 7 son estaciones regulares y 2 son estaciones multilínea), 9 operarios y 4 etapas (donde, por ejemplo, la etapa 1 incluye las estaciones 1, 2 y 3).

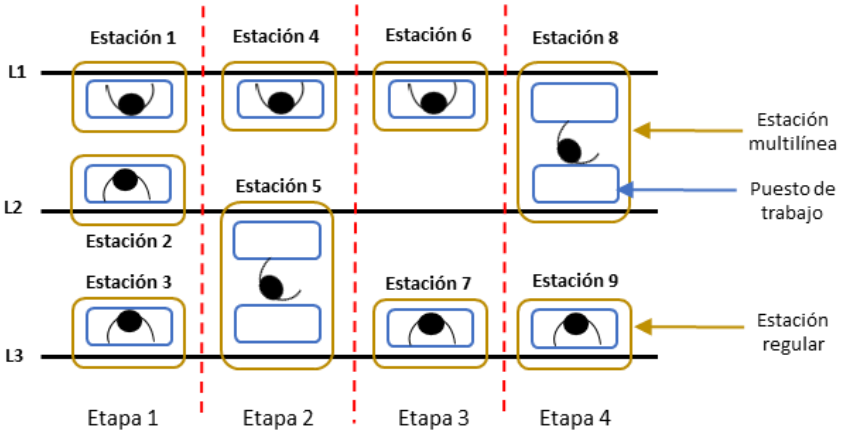

Figura 2. Esquema de un sistema PALB.

A continuación se muestra el modelo de programación lineal binaria (PLB) propuesto por Gökçen et al. (2006) para el PALBP. El modelo de PLB presentado asigna las tareas a las estaciones con el objetivo de minimizar el número de estaciones utilizadas para un tiempo de ciclo ct dado igual para todas las líneas.

Datos:

H	Número total de líneas
ct	Tiempo de ciclo de todas las líneas
n_h	Número de tareas en la línea h; $h = 1, \dots, H$
t_{hi}	Tiempo de proceso de la tarea i de la línea h; $h = 1, \dots, H$; $i = 1, \dots, n_h$
LB	Cota inferior del número de estaciones
UB	Cota superior del número de estaciones
IP_{hi}	Conjunto de las predecesoras inmediatas de la tarea i de la línea h; $h = 1, \dots, H$; $i = 1, \dots, n_h$

Variables:

$x_{hik} \in \{0,1\}$	1 si la tarea i en la línea h es asignada a la estación k, 0 en caso contrario: $h = 1, \dots, H$; $i = 1, \dots, n_h$; $k = 1, \dots, UB$
$U_{hk} \in \{0,1\}$	1 si la estación k es requerida en la línea h, 0 en caso contrario: $h = 1, \dots, H$; $k = 1, \dots, UB$
$y_k \in \{0,1\}$	1 si la estación k es requerida; 0 en caso contrario: $k = 1, \dots, UB$

Función objetivo:

$$\left[\text{MIN}\right] \sum_{k=LB+1}^{UB} y_k \tag{1}$$

Sujeto a:

$$\sum_{k=1}^{UB} x_{hik} = 1 \qquad i = 1, \dots, n_h;\ h = 1, \dots, H \tag{2}$$

$$\sum_{i=1}^{n_h} t_{hi} \cdot x_{hik} + \sum_{i=1}^{n_{h+1}} t_{(h+1)i} \cdot x_{(h+1)ik} \le ct \cdot y_k \qquad k = 1, \dots, UB;\ h = 1, \dots, H-1 \tag{3}$$

$$\sum_{i=1}^{n_h} x_{hik} - n_h \cdot U_{hk} \le 0 \qquad h = 1, \dots H;\ k = 1, \dots, UB \tag{4}$$

$$U_{hk} + U_{(h+a)k} \le 1 \qquad h = 1, \dots, H-2;\ a = 2, \dots, H-h;\ k = 1, \dots, UB \tag{5}$$

$$\sum_{k=1}^{UB}(UB-k+1)\cdot(x_{hrk}-x_{hik})\geq 0 \qquad h=1,\ldots,H;\ r\in IP_{hi};\ i=1,\ldots,n_h \tag{6}$$

La función objetivo (1) consiste en minimizar el número de estaciones utilizadas. (2) asegura que todas las tareas son asignadas solo una vez a una estación. (3) asegura que la carga de trabajo de las estaciones no excede el tiempo de ciclo. (4) y (5) aseguran que un operario que trabaja en una estación k y línea h puede realizar tareas de solo una línea adyacente. (6) impone la relación de precedencias.

A continuación se muestra un ejemplo, utilizando los grafos de precedencias de la Figura 3, para mostrar la utilidad del PALB con estaciones multilínea respecto a un equilibrado individual de líneas en paralelo (sin estaciones multilínea).

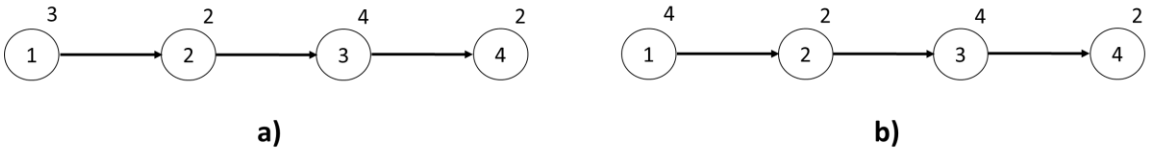

Figura 3. a) grafo de precedencias del producto A, b) grafo de precedencias del producto B.

La Figura 3 muestra los grafos de precedencias de dos productos A y B, asignados a las líneas L1 y L2, respectivamente. El número dentro del nodo hace referencia a la tarea y el número superior derecho a su tiempo de proceso. El tiempo de ciclo de todas las líneas es $ct = 4$. La Figura 4 muestra una solución óptima para SALBP, con 4 estaciones en la línea 1 y 4 estaciones en la línea 2, haciendo un total de 8 estaciones y operarios. El número dentro del puesto de trabajo hace referencia a la tarea asignada.

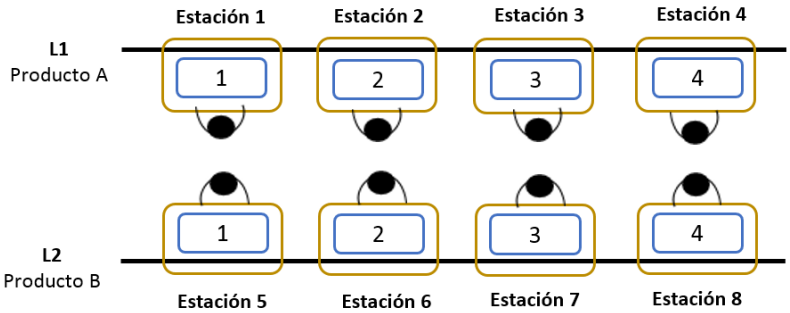

Figura 4. Representación de una solución óptima del ejemplo de la Figura 3 al utilizar un equilibrado de líneas individual.

Sin embargo, usando el enfoque del PALB, se puede reducir el número de estaciones utilizadas.

La Figura 5 muestra una solución óptima para PALBP, con 6 estaciones y operarios, de las cuales 4 son estaciones regulares y 2 son estaciones multilínea (estación 3 y 6). El número dentro del puesto de trabajo hace referencia a la tarea asignada.

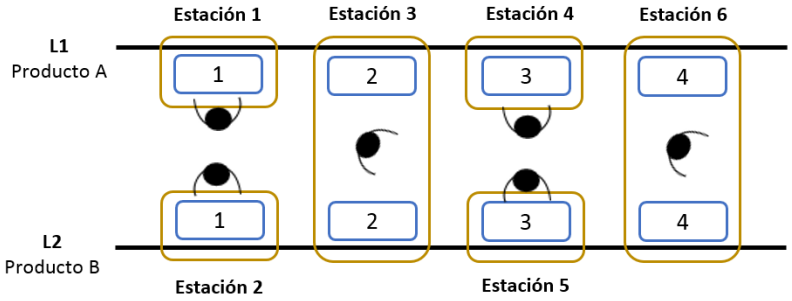

Figura 5. Representación de una solución óptima del ejemplo de la Figura 3 al utilizar el PALB con estaciones multilínea.

En el ejemplo anterior se observa que al utilizar el concepto del PALB se ha podido reducir el número de estaciones respecto al equilibrado individual de las líneas, pasando de 8 a 6 estaciones y operarios.

En la industria pueden existir sistemas PALB donde las líneas trabajen con tiempos de ciclo diferentes, incluso si fabrican el mismo producto. A continuación, se muestra un ejemplo utilizando los grafos de precedencias de la Figura 6 para mostrar la resolución de un sistema PALB con líneas con tiempos de ciclo diferentes.

Figura 6. a) grafo de precedencias del producto A, b) grafo de precedencias del producto B.

La Figura 6 muestra los grafos de precedencias de dos productos A y B, asignados a las líneas L1 y L2, respectivamente. El tiempo de ciclo ct_h de la línea h ($h = 1, \dots, H$) es: $ct_1 = 4$ para línea 1, y $ct_2 = 16$ para línea 2.

La Figura 7 muestra una solución óptima para SALBP, con 3 estaciones en la línea 1 y 3 estaciones en la línea 2, haciendo un total de 6 estaciones y operarios. El número dentro del puesto de trabajo hace referencia a la tarea asignada.

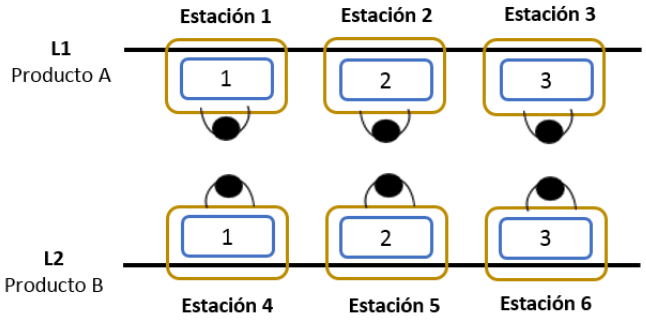

Figura 7. Representación de una solución óptima del ejemplo de la Figura 6 al utilizar un equilibrado de líneas individual.

Trabajar con líneas con tiempos de ciclo diferentes, dificulta el control de la restricción del tiempo de ciclo en las estaciones multilínea. En este caso, para resolver el PALB con líneas con tiempos de ciclo ct_h diferentes, el problema se puede reducir al equivalente de trabajar con el mismo tiempo de ciclo para todas las líneas. Para ello, en la literatura es usual utilizar el procedimiento conocido como mínimo común múltiplo (*Least Common Multiple* (LCM)) propuesto por Gökçen et al. (2006). Al aplicar el procedimiento LCM se obtiene un tiempo de ciclo lcm para todas las líneas, que se calcula: $lcm = m.c.m.\{ct_1, \dots, ct_H\}$. Trabajar con el tiempo de ciclo lcm hace que los sistemas, con estaciones multilínea, tengan que producir en lotes de q_h unidades, donde: $q_h = \frac{lcm}{ct_h}$ $(h = 1, \dots H)$. En consecuencia, los tiempos de proceso de las tareas t_{hi} se tienen que normalizar al tiempo lcm de la siguiente forma: $t'_{hi} = t_{hi} \cdot q_h$ $(i = 1, \dots, n_h;$ $h = 1, \dots H)$.

Usualmente en la literatura del PALBP se asumen que las unidades de producto fluyen, de una en una, a través de una línea de montaje. Por este motivo, no se considera la necesidad de buffers en las estaciones. Sin embargo, el procedimiento LCM implica que las líneas puede ser que tengan que producir en lotes. Por lo descrito anteriormente, parece necesario considerar la necesidad de buffers en las estaciones al utilizar el procedimiento LCM u otro similar.

A continuación se muestra la resolución del ejemplo propuesto al utilizar el procedimiento LCM. Se obtiene un tiempo común para las líneas $lcm = m.c.m.\{4, 16\} = 16$, con lotes $q_1 = \frac{16}{4} = 4$ unidades y $q_2 = \frac{16}{16} = 1$ unidad. Los tiempos de las tareas del producto asignado a la línea 2 mantienen su valor ya que en la línea 2 se produce en lotes de $q_2 = 1$ unidad. En la línea 1 se produce en lotes de $q_1 = 4$ unidades, por tanto, los tiempos de las tareas del producto de la línea 1 se tienen que normalizar: $t'_{1,1} = 4 \cdot 4 = 16$; $t'_{1,2} = 2 \cdot 4 = 8$; $t'_{1,3} = 4 \cdot 4 = 16$. La Figura 8 muestra una solución óptima para

PALBP, con 5 estaciones y operarios, de las cuales 4 son estaciones regulares y 1 es una estación multilínea (estación 3). El número dentro del puesto de trabajo hace referencia a la tarea asignada.

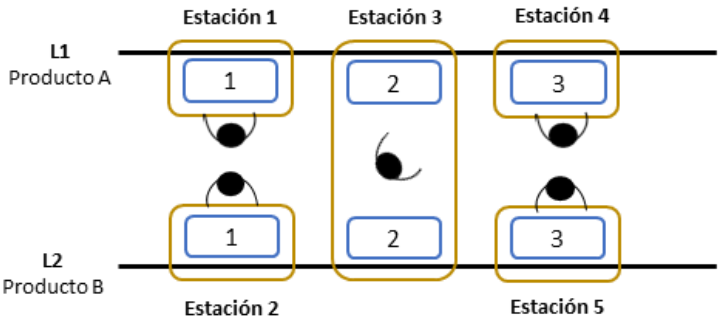

Figura 8. Representación de una solución óptima del ejemplo de la Figura 6 al utilizar el PALB con estaciones multilínea.

De la solución del ejemplo anterior se observa que en el tiempo de ciclo $lcm = 16$ (aquí llamado superciclo) se deben procesar 4 unidades de producto A en línea 1 y 1 unidad de producto B en línea 2.

En las estaciones regulares, cada ct_h, los operarios solo trabajan sobre una unidad en su estación de la línea h; de esta forma no es necesario el uso de buffers en estas estaciones. En cambio, en la estación multilínea (estación 3), el operario debe respetar el tiempo de ciclo ct_h de cada línea, porque, al trabajar en dos líneas con tiempos de ciclo diferentes, es posible que tenga que adelantar (o atrasar) la realización de las tareas de una unidad de una de las líneas, para poder realizar las tareas de la unidad de la otra línea de mayor (menor) tiempo de ciclo. Es en este caso cuando surge la necesidad de disponer de buffers en las estaciones multilínea y obviarla puede hacer no implementable, en un sistema industrial real, una solución académica del PALBP.

Para facilitar la comprensión de lo introducido anteriormente, la Figura 9 muestra una propuesta de solución (la secuenciación y programación de las tareas asignada a la estación multilínea) para dividir el tiempo de trabajo del operario (de la estación 3) entre ambas líneas. En dicha figura se puede observar: cuando el operario está realizando las tareas de una unidad (recuadros grises) de la línea L1 y de la L2, así como el instante en que empieza y finaliza cada tarea; cuando una unidad aún no ha sido procesada (recuadros naranjas); y, finalmente, cuando ya ha sido procesada (recuadros verdes). Las unidades procesadas y no procesadas pueden permanecer en su línea (L1 o L2) o en el buffer de sus respectivas líneas (Buffer L1 o Buffer L2). Las unidades se representan con el símbolo u (o con el símbolo $u0$, para aquellas unidades que ya hay en los buffers al inicio del superciclo, es decir, instante 0 aquí representado)

acompañadas de un número que hace referencia a la unidad del lote (valor entre 1 y q_h). Las tareas se representan con el símbolo T y acompañadas de un número (índice i) que hace referencia a la tarea en cuestión.

Se recalca que los tiempos de procesos normalizados t'_{hi} solo se utilizan para realizar el equilibrado de las líneas, para la división del trabajo del operario en la estación multilínea se utilizan los tiempos de procesos originales t_{hi}.

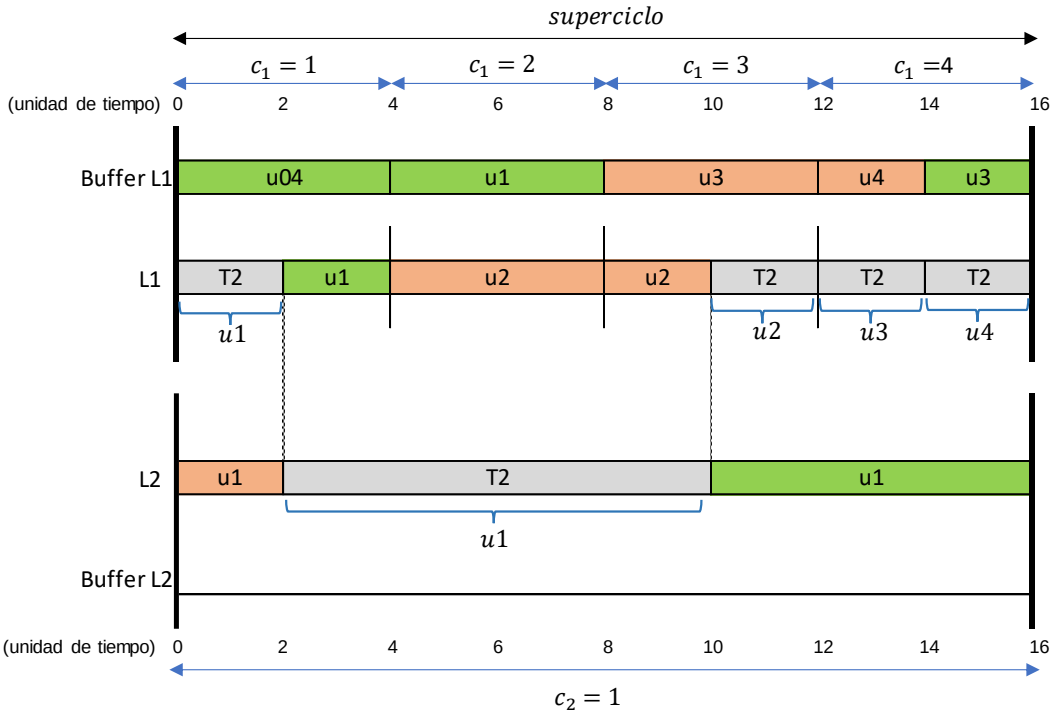

Figura 9. Solución para la estación multilínea 3 de la solución de la Figura 8 del ejemplo propuesto en la Figura 6.

Como se ha especificado, en cada puesto de trabajo y en cada tiempo de ciclo ct_h, entra una unidad sin procesar y sale una procesada de cada línea h; en L1 en los instantes 0, 4, 8, 12 y 16 (instante 0 del siguiente superciclo), y en L2 en los instantes 0 y 16 (instante 0 del siguiente superciclo). En la solución mostrada en la Figura 9, el operario está trabajando en la unidad $u1$ en L2 entre los instantes 2 y 10, por lo que no se procesa la unidad $u2$ que llega a L1 en su segundo ciclo ($c_1 = 2$, donde $c_h = 1, \dots, q_h$ es el ciclo de la línea h). Se verifica, de este modo, la necesidad de disponer de un buffer en L1 y, en este caso, de un tamaño de una unidad. Este buffer contiene la unidad ya procesada $u04$ al inicio del superciclo; es decir, la unidad $u4$ procesada en el ciclo 4 ($c_1 = 4$) del superciclo anterior. De esta forma en el ciclo $c_1 = 2$ se dispone en el Buffer L1 de la unidad $u1$ ya procesada, la cual pasa a la siguiente estación en el instante 8. Y la unidad $u2$ que llega a L1 en el ciclo $c_1 = 2$, se procesa en el ciclo $c_1 = 3$ entre los instantes 10 y 12.

Cabe destacar que la existencia y dimensionado de los buffers no solo depende del conjunto de tareas asignadas a la estación multilínea, sino también de la secuencia de estas y de su programación. En la Figura 10 se representa una propuesta de solución alternativa (diferente secuenciación y programación de las tareas), que presenta la necesidad de disponer de un buffer en L1 de tamaño dos unidades (con las unidades ya procesadas $u03$ y $u04$ al inicio del superciclo; es decir, las unidades $u3$ y $u4$, respectivamente, procesadas en el ciclo 4 del superciclo anterior). Se demuestra, así, la importancia de la secuenciación y programación de las tareas asignadas a una estación multilínea.

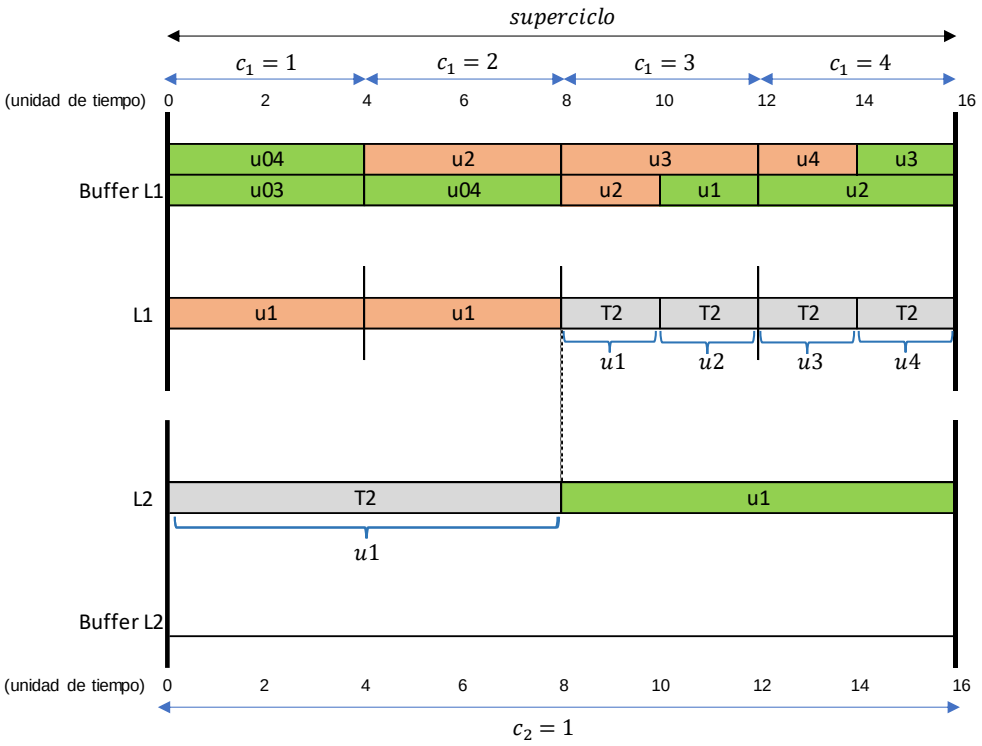

Figura 10. Solución alternativa para la estación multilínea 3 de la solución de la Figura 8 del ejemplo propuesto en la Figura 6.

Como se ha introducido anteriormente, es imprescindible considerar los buffers (existencia y dimensionado) para que una solución académica del PALBP con líneas con tiempos de ciclo diferentes pueda ser implementada en la realidad. En este caso hay que tener presente una serie de posibles circunstancias en su implementación (Pastor et al., 2021): 1) el tamaño del buffer depende de la asignación de tareas (equilibrado de líneas) a la estación multilínea, y de la secuenciación y programación de estas; 2) se puede dar el caso que no exista una solución factible para el conjunto de tareas asignadas a la estación multilínea, lo que afecta directamente al equilibrado de las líneas; 3) el tamaño de los productos puede impedir una solución con buffers; 4) unos lotes q_1 y q_2 muy grandes, incluso con productos pequeños, pueden imponer

buffers muy voluminosos que hagan no factible una solución; y 5) en cualquier caso, los buffers presentan un coste que hay que considerar en la implementación.

1.3 Estado del arte del PALBP

Desde que Gökçen et al. (2006) introdujeran y publicaran el primer estudio del PALBP, varios autores han publicado diferentes trabajos enfocados en este problema. A pesar de ser un problema relativamente nuevo, el PALBP ha recibido la atención de varios investigadores y ya se han presentado varios estudios hasta la fecha. Algunos de estos estudios son estados del arte extensos donde se hace una revisión en profundidad de los trabajos presentados, véase Lusa (2008) y Aguilar et al. (2020).

A continuación, se presenta el estado del arte de la literatura del PALBP. Los diferentes trabajos se han agrupado considerando el *layout* de las líneas (rectas, de dos lados, y en forma de U) y el número de modelos procesados en una línea (único y mixtos). Todos los artículos presentados a continuación se encuentran clasificados en la tablas del Anexo A (véase en el documento externo “Anexos”), de acuerdo con el tipo de línea y modelo.

PALBP de líneas rectas y modelo único

Gökçen et al. (2006) presentan un modelo de programación lineal binaria (PLB) y dos procedimientos heurísticos (*pasive case* y *active case*) para resolver el problema PALB de líneas rectas y modelo único. La heurística *pasive case*, se enfoca en el caso de trabajar con el mismo producto y tiempo de ciclo, en todas las líneas. El procedimiento consiste en equilibrar las líneas por separado, con cualquier procedimiento para SALBP, y posteriormente juntar dos estaciones opuestas, siempre y cuando la carga de trabajo de cada una de las estaciones a juntar no sea mayor a la mitad del tiempo ciclo. El *active case* se enfoca en el caso de trabajar con el mismo producto y con líneas con el mismo o diferente tiempo de ciclo. En el caso de trabajar con líneas con tiempos de ciclos diferentes, se procede a utilizar el procedimiento LCM (introducido en el apartado anterior). El *active case* es un procedimiento greedy, con un criterio aleatorio de asignación de tareas candidatas a estaciones. Una tarea es candidata si: no ha sido asignada; sus predecesoras ya se han asignado; y su tiempo de proceso no es mayor al tiempo muerto o disponible (diferencia entre el tiempo de ciclo y la carga de trabajo) de la estación en curso. Aunque los autores en su estudio consideran la posibilidad de que las líneas trabajen con tiempos de ciclo diferentes, en su experimento computacional, únicamente consideran líneas con el mismo tiempo de ciclo.

Benzer et al. (2007) presentan un *network-model* basado en el camino más corto para resolver el problema propuesto por Gökçen et al. (2006).

Çerçioğlu et al. (2009) presentan una metaheurística basada en el recocido simulado (*Simulated Annealing* (SA)) y realizan el mismo experimento computacional de Gökçen et al. (2006).

Özcan et al. (2009) desarrollan un algoritmo de búsqueda tabú (*Tabu Search* (TS)) con el objetivo de minimizar dos objetivos simultáneamente: el número de estaciones y la diferencia de carga de trabajo entre estaciones.

Baykasoğlu et al. (2009) desarrollan un algoritmo basado en optimización por colonias de hormigas (*Ant Colony Optimization* (ACO)) con el objetivo minimizar el número de estaciones y la diferencia de carga de trabajo entre estaciones.

Scholl and Boysen (2009) presentan un modelo de PLB, y un procedimiento (conocido como ABS) basado en un *Branch and Bound* y en una heurística de asignación de modelos a línea. El modelo de PLB considera el desplazamiento del operario entre líneas, sin embargo, este desplazamiento no es tenido en cuenta en el experimento computacional. Mediante un experimento computacional los autores demuestran la superioridad de ABS respecto a los procedimientos presentados anteriormente.

Guo and Tang (2009) presentan un procedimiento basado en búsqueda dispersa (*Scatter Search* (SS)), con el objetivo de minimizar el número de estaciones. El SS propuesto utiliza tres procedimientos basados en el *active case* para construir una solución. Los autores realizan un experimento computacional donde demuestran el buen rendimiento del SS.

Kara et al. (2010) presentan dos modelos de programación de objetivos (*Goal Programming* (GP)) para resolver el problema planteado en Gökçen et al. (2006) con un triple objetivo: minimizar el número de estaciones, el tiempo de ciclo y el número de tareas asignadas a una estación. Además, cada línea de montaje puede trabajar con un tiempo de ciclo diferente, en cuyo caso se utiliza el procedimiento LCM.

Özbakir et al. (2011) presentan un ACO el cual utiliza múltiples colonias, con el objetivo de minimizar el número de estaciones y la diferencia de carga de trabajo entre estaciones. En el ACO propuesto cada una de las colonias tiene asignada una regla de prioridad, la cual es utilizada por las hormigas para generar una solución del problema. El rastro de feromonas utilizado por las hormigas está compuesto por una matriz de feromonas local (cada colonia dispone de una matriz local que se actualiza con el recorrido realizado por las hormigas de dicha colonia) y una matriz de feromonas global

(que se actualiza con el mejor recorrido aportado entre todas las colonias). Un experimento computacional demuestra la superioridad del ACO respecto a los demás procedimientos.

Baykasoğlu et al. (2012) modifican el ACO presentado en Özbakir et al. (2011) para resolver el problema con tiempos estocásticos, considerando tiempos de tarea y de ciclo inciertos. Estos tiempos son representados como números difusos de tipo triangular.

Grangeon and Norre (2012) presentan una metaheurística basada en SA. El SA propuesto utiliza un procedimiento orientado a tareas para construir una solución factible del problema. Los autores realizan el mismo experimento computacional que en Gökçen et al. (2006).

Kara and Atasagun (2013) añaden otro tipo de factores y características al problema propuesto inicialmente en Gökçen et al. (2006). Los autores consideran la posibilidad de que el tiempo requerido para realizar una tarea puede variar dependiendo del operario que realice la tarea, las herramientas utilizadas y si el operario recibe ayuda de un asistente. Este problema fue presentado por Faaland et al. (1992) para el ALBP y es conocido como *resource dependent* ALBP (RDALBP). Los autores presentan un modelo de PLB con el objetivo de minimizar el coste asociado al número de estaciones, asistentes y la asignación del equipo. Sin embargo, no realizan ningún experimento computacional. Los autores también consideran el uso del procedimiento LCM en el caso de que las líneas trabajen con tiempos de ciclo diferentes.

Araújo et al. (2015) presentan un modelo de programación lineal entera mixta (PLEM), una metaheurística basada en TS y otra basada en un algoritmo genético (*Genetic Algorithm* (GA)). Los autores se enfocan en el problema de asignar un conjunto de operarios con diferentes habilidades, a una línea de montaje, con el objetivo de minimizar el tiempo de ciclo. Un experimento computacional demuestra un mejor rendimiento del TS respecto al modelo de PLEM y GA. En el caso de que las líneas trabajen con tiempos de ciclo diferentes, se utiliza el procedimiento LCM.

Çil et al. (2017) se enfocan en el PALBP con líneas robóticas. Los autores consideran el uso de robots con diferentes niveles de rendimiento en las líneas. Proponen un modelo de PLEM para ejemplares pequeños y tres metaheurísticas basada en la búsqueda en haz (*Beam Search* (BS)), con el objetivo de minimizar el tiempo de ciclo. Los autores realizan un experimento computacional para evaluar el rendimiento de sus procedimientos.

Özcan (2018) propone un modelo de PLEM y un TS para el PALBP con tiempos estocásticos. El modelo de PLEM se adapta al problema con datos estocásticos. El TS desarrollado utiliza una lista tabú dinámica. El autor realiza un experimento computacional. El autor también considera el uso del procedimiento LCM en el caso de que las líneas trabajen con tiempos de ciclo diferentes.

Özcan (2019) introduce en el PALBP, el problema de secuenciación de tareas asignadas a una estación (Andrés et al. (2008) y Pastor et al. (2010)). En su estudio se considera el tiempo de desplazamiento de un operario dentro de una estación y entre estaciones (en el caso de utilizar una estación multilínea). El autor presenta un modelo de PLB y una metaheurística basada en SA, con función objetivo de minimizar el número de estaciones. El autor realiza un experimento computacional. En el caso de tener líneas con tiempos de ciclo diferentes, el autor recomienda el uso del procedimiento LCM.

Özbakır and Seçme (2020) se enfocan en el PALBP con tiempos estocásticos, como Özcan (2018). Pero a diferencia de Özcan (2018), los autores consideran la probabilidad de no terminar las tareas dentro del tiempo de ciclo, considerando en este caso un coste añadido. Los autores presentan un modelo de PLEM basado en el propuesto por Özcan (2018), pero con la función objetivo de minimizar el coste asociado al número de estaciones, el coste de tareas incompletas y el coste de equipamiento. Así mismo, los autores desarrollan una hiperheurística basada en SA. En la hiperheurística, una tarea candidata es asignada a una estación de acuerdo a una regla de prioridad, que es seleccionada de entre un conjunto de reglas de prioridad disponibles. Un estudio computacional demuestra el mejor rendimiento de la hiperheurística basada en SA respecto al método clásico SA. Además, se considera la posibilidad de que las líneas trabajen con tiempos de ciclo diferentes, en este caso, se utiliza el procedimiento LCM.

PALBP de líneas rectas y modelos mixtos

Esmaeilian et al. (2009) introducen el problema de líneas de modelos mixtos al PALBP. Los autores proponen un procedimiento greedy para resolver el problema, con el objetivo de minimizar el tiempo de ciclo. Esmaeilian et al. (2011) desarrollan un modelo de PLB y un TS, el cual utiliza una heurística greedy (con una regla de prioridad compuesta) para obtener una solución inicial con la que inicializar el TS. Esmaeilian et al. (2015) presentan un modelo de PLB multiobjetivo: minimizar el número de estaciones y el coste de los operarios. En el problema se considera que los operarios tienen diferentes niveles de habilidades y estos son asignados a las estaciones de acuerdo a su nivel de habilidades. En los trabajos mencionados se considera la posibilidad de que

las líneas trabajen con tiempos de ciclo diferentes, en este caso, utilizan el procedimiento LCM.

Özcan et al. (2010a) consideran la secuenciación de los diferentes modelos en las líneas. Los autores desarrollan una metaheurística basada en SA con el objetivo de minimizar el número de estaciones y la diferencia de carga de trabajo entre estaciones.

Chutima and Yothaboriban (2017) se centran en aspectos como la relación entre las habilidades del operario y las tareas asignadas a su estación. Los autores exponen que cada trabajador tiene una habilidad primaria y una habilidad secundaria en la que son menos competentes; la idea principal es asignar el trabajador adecuado (habilidad) al trabajo correcto (estación de trabajo). Desarrollan un algoritmo de optimización basada en biogeografía (*Biogeographic-based optimisation* (BBO)) con el objetivo de minimizar 4 objetivos: 1) el número de estaciones; 2) la longitud de las líneas; 3) la diferencia de carga de trabajo entre estaciones; y 4) la diferencia de habilidades (de los operarios) entre las estaciones. En el caso de trabajar con líneas con tiempos de ciclo diferentes se utiliza el procedimiento LCM.

PALBP de líneas de dos lados y modelo único

Özcan et al. (2010b) presentan el primer estudio dirigido al equilibrado de dos o más líneas de montaje de dos lados en paralelo. Los autores desarrollan una metaheurística basada en TS con el objetivo de minimizar el número de estaciones. El estudio considera la posibilidad de que las líneas puedan trabajar con diferentes tiempos de ciclo; en este caso se sugiere el uso del procedimiento LCM para obtener un tiempo de ciclo común para las líneas.

Kucukkoc et al. (2013) desarrollan un algoritmo basado en el ACO. El procedimiento propuesto tiene como objetivos minimizar el número de estaciones y la longitud de línea. Por otra parte, un factor de ponderación permite decidir la relevancia de ambos objetivos. En el caso de tener líneas con tiempos de ciclo diferentes, los autores recomiendan el uso del procedimiento LCM.

Kucukkoc and Zhang (2013) presentan un procedimiento basado en un algoritmo GA con el objetivo de minimizar el número de estaciones. Los autores también consideran el uso del procedimiento LCM en el caso de que las líneas trabajen con tiempos de ciclo diferentes.

Ağpak and Zolfaghari (2015) desarrollan 5 modelos de PLEM, con diferentes función objetivo y con la posibilidad de añadir varias restricciones al modelo. Los autores clasifican el problema y sus extensiones de las siguientes formas: Tipo I/LL, para un

tiempo de ciclo dado, minimizar la longitud de línea; Tipo I/K, para un tiempo de ciclo dado, minimizar el número de estaciones; Tipo II-K, para un número dado de estaciones, minimizar el tiempo de ciclo; Tipo II-LL, para una longitud de línea dada, minimizar el tiempo del ciclo; y minimizar el coste total, que incluye el coste de: la estación, la longitud de la línea, y de la mano de obra. Además, se presentan varias restricciones de asignación: asignar una tarea a una estación de trabajo específica; sincronización a la hora de mover o realizar tareas; tareas que tienen que ser asignadas a la misma estación o en diferentes estaciones; distancia entre tareas (p.ej. en tiempo); y restricciones relacionadas con el área (espacio o longitud) de la estación.

Kucukkoc and Zhang (2015d) introducen el problema Type-E en la literatura PALBP. El problema Type-E busca maximizar la eficiencia de la línea a través de minimizar el producto del tiempo de ciclo y el número de estaciones. Los autores presentan un modelo de PLEM para la descripción formal del problema y un algoritmo basado en el ACO. El ACO desarrollado utiliza tres reglas de prioridad (tiempo de proceso más pequeño, máxima suma del tiempo propio y de las tareas sucesoras, y selección al azar) que son escogidas por las hormigas al azar, y utilizadas como información heurística. En el caso de tener líneas con tiempos de ciclo diferentes, los autores recomiendan el uso del procedimiento LCM.

Kucukkoc and Zhang (2015a) desarrollan un algoritmo GA y un modelo de programación no lineal entera mixta (PNLEM) para formalizar el problema; continuando con el estudio presentado por Kucukkoc and Zhang (2013). El problema presenta restricciones de asignación de tareas: tareas que deben ser realizadas en la misma estación y tareas que no pueden ser realizadas en la misma estación. Por otra parte, también se considera la posibilidad que las líneas pueden trabajar con diferentes tiempos de ciclo, utilizando el procedimiento LCM.

Tapkan et al. (2016) consideran el tiempo de desplazamiento en las estaciones multilínea y formalizan el problema como un modelo de PNLEM. Debido a la complejidad del problema, los autores desarrollan dos metaheurísticas. La primera basada en el algoritmo de abejas (*Bee Algorithm* (BA)), y la segunda en el algoritmo de colonia de abejas artificiales (*Artificial Bee Colony* (ABC)). Ambas metaheurísticas tienen como objetivo minimizar: el número de estaciones, la variación de la carga de trabajo entre las estaciones y el tiempo de recorrido entre estaciones. Además, se considera la posibilidad de que las líneas trabajen con tiempos de ciclo diferentes, en este caso, se utiliza el procedimiento LCM.

Yadav et al. (2020) consideran la posibilidad de compartir herramientas entre estaciones adyacentes, con el objetivo de reducir el número de estaciones utilizadas. Los autores presentan un modelo de PLEM con el objetivo de minimizar el número de estaciones y el número de herramientas utilizadas.

PALBP de líneas de dos lados y modelos mixtos

Zhang and Kucukkoc (2013) introducen, por primera vez en la literatura, el problema de las líneas de modelos mixtos a los problemas PALB de líneas de dos lados.

Kucukkoc and Zhang (2014c) desarrollan un algoritmo basado en el ACO con el fin de equilibrar las líneas y secuenciar los modelos simultáneamente. El algoritmo presentado es conocido como *Agent Based* ACO y *Sequencing* (ABACO/S). Tres objetivos son utilizados para evaluar las soluciones obtenidas: minimizar el número de estaciones, la variación de la carga de trabajo entre las estaciones y la longitud de línea. La función objetivo consta de factores de ponderación que permiten decidir la relevancia de los tres objetivos. Por otra parte, todas las posibles secuencias de los modelos en las líneas son generadas en el procedimiento ABACO/S. Además, el procedimiento LCM es utilizado en el caso de trabajar con líneas con tiempos de ciclo diferentes.

Kucukkoc and Zhang (2014a) presentan una versión modificada del algoritmo ABACO/S propuesto por Kucukkoc and Zhang (2014c). En este trabajo, a diferencia de Kucukkoc and Zhang (2014c), no se considera como parte de la función objetivo la variación de la carga de trabajo entre las estaciones, y la secuenciación de los modelos en las líneas es un dato prefijado. En este estudio el algoritmo ABACO es mejorado por la inclusión de diez heurísticas greedy (basadas en reglas de prioridad de la literatura), las cuales son seleccionadas aleatoriamente por las hormigas para obtener una solución. Por otra parte, los objetivos pueden ser ponderados por el usuario según su importancia, y las líneas pueden trabajar con tiempos de ciclo diferentes, en cuyo caso se utiliza el procedimiento LCM.

Kucukkoc and Zhang (2014b) presentan la continuación de los estudios de Kucukkoc and Zhang (2014c) y (2014a). En este estudio, como en Kucukkoc and Zhang (2014c), la secuenciación de los modelos en las líneas es considerado como un elemento a calcular en el procedimiento ABACO/S; el problema es modelizado con un modelo de PLEM y el algoritmo ABACO/S es mejorado por la inclusión de las mismas diez heurísticas greedy presentadas en Kucukkoc and Zhang (2014a). El algoritmo ABACO/S presentado por los autores puede usar dos modelos de secuenciación diferentes: 1) secuenciación de modelos combinatorios; y 2) secuenciación de modelos aleatorios. En

la secuenciación de modelos combinatorios, el algoritmo prueba todas las secuencias posibles. Y, en la secuenciación de modelos aleatorios, el algoritmo prueba un número definido de secuencias generadas aleatoriamente. La elección del modelo de secuenciación es determinada por el usuario. Por otra parte, se considera la posibilidad que las líneas trabajen con tiempos de ciclo diferentes, en tal caso se utiliza el procedimiento LCM.

Kucukkoc and Zhang (2016a) presentan un algoritmo llamado *Agent-Based* ACO-*Genetic Algorithm* (ABACO/S-GA). El algoritmo GA es integrado en ABACO/S para generar diferentes secuencias de modelos. Asimismo, para proporcionar información heurística y aumentar la capacidad de búsqueda local del algoritmo, cada una de las hormigas puede seleccionar diez heurísticas greedy como en Kucukkoc and Zhang (2014a). El objetivo principal es minimizar el número de estaciones y la longitud total de la línea, pudiendo ser estos objetivos ponderados por factores según su importancia. Los autores recomiendan el uso del procedimiento LCM, en caso de trabajar con líneas con tiempos de ciclo diferentes.

Kucukkoc and Zhang (2016b) modifican el algoritmo ABACO/S propuesto por Kucukkoc and Zhang (2014a), para construir un algoritmo ABACO flexible. El procedimiento presentado no considera la secuenciación de los modelos en las líneas, sino que selecciona el tiempo máximo de la tarea *i* de entre todos los modelos *m* asignados a una línea. De esta forma se garantiza que la suma de todas las tareas asignadas a una determinada estación no supera el tiempo de ciclo. Los principales objetivos son minimizar el número de estaciones y la longitud de línea, los cuales pueden ser ponderados de acuerdo a su relevancia. Además, se considera la posibilidad que las líneas puedan trabajar con tiempos de ciclo diferentes, en este caso se utiliza el procedimiento LCM.

PALBP de líneas en forma de U y modelo único

Kucukkoc and Zhang (2015b) presentan una nueva configuración de línea la cual combina las ventajas de las líneas en paralelo y las líneas en forma de U. Los autores desarrollaron una heurística llamada *Parallel U-line Heuristic* (PUH) para resolver el problema. El PUH es una heurística greedy de un solo paso, que utiliza, como regla de prioridad, la combinación de dos criterios de la literatura: máximo número de sucesoras y, máxima suma del tiempo propio y de las tareas sucesoras. El principal objetivo del procedimiento es minimizar el número de estaciones y la longitud de las líneas. El problema considera la posibilidad de que las líneas puedan trabajar con tiempos de ciclo diferentes, utilizando el procedimiento LCM.

PALBP de líneas en forma de U y modelos mixtos

Kucukkoc and Zhang (2015c) introducen el problema de líneas de modelos mixtos a las líneas en paralelo en forma de U. Posteriormente, Kucukkoc and Zhang (2017) adaptan la heurística PUH propuesta por Kucukkoc and Zhang (2015c) al problema de líneas de modelos mixtos. La heurística resultante, conocida como *modelo mixto* PUH (MPUH), considera todas las posibles combinaciones de secuencias de modelos. Además, cada línea de montaje puede trabajar con un tiempo de ciclo diferente, en cuyo caso se utiliza el procedimiento LCM.

Chutima and Jirachai (2020) presentan un algoritmo evolucionario multi objetivo adaptativo (AMOEA) basado en descomposición (AMOEA/D) e hibridado con un algoritmo BBO (AMOEA/D-BBO). El AMOEA/D es utilizado para resolver problemas multiobjetivo, el cual divide el problema en varios subproblemas mono objetivo y los resuelve de manera simultánea. Además, tiene la capacidad de ir variando (de manera controlada, es decir, dentro de un rango de valores) ciertos parámetros del algoritmo mientras se realiza la búsqueda de soluciones. Los autores desarrollan un AMOEA/D-BBO para minimizar de manera conjunta los siguientes objetivos: número de estaciones, diferencia de carga de trabajo entre estaciones, longitud de líneas, y diferencia entre tareas asignadas a la misma estación.

Conclusiones

De la revisión del estado del arte, se observa que más de la mitad de los estudios presentados consideran la posibilidad de que las líneas de montaje puedan trabajar con tiempos de ciclo diferentes. En este caso, es común el uso del procedimiento LCM propuesto en Gökçen et al. (2006) para reducir el problema al caso de trabajar con el mismo tiempo de ciclo. Al trabajar con un tiempo de ciclo común para las líneas, es posible que estas tengan que producir en lotes y, así, es probable que se necesiten áreas de almacenamiento (buffers) en las estaciones (Aguilar et al., 2020).

Cabe destacar que antes de la realización de esta tesis, en la literatura del PALBP no se consideraba el uso de ningún tipo de buffer en el caso de producir en lotes (Aguilar et al., 2020). Además, se observa que en los trabajos presentados se asume que en una estación no se permite más de una unidad de producto en un puesto de trabajo en el mismo ciclo. Finalmente, hay que tener presente que no considerar la necesidad (existencia y dimensionado) de buffers en el equilibrado de las líneas en paralelo, puede llevar a diseñar soluciones no implementables en la realidad.

Capítulo 2. Problema de equilibrado de líneas de montaje en paralelo con estaciones multilínea y dimensionado de buffers (PALBP-B)

2.1 Introducción

Como se ha observado en la sección anterior, existen varios trabajos del PALBP que consideran la posibilidad de que las líneas de montaje tengan tiempos de ciclo diferentes. En este caso, es posible que tengan que producir en lotes y sea imprescindible considerar la necesidad de buffers en las estaciones multilínea. Sin embargo, no se manifiesta en la literatura del PALBP, el uso de buffers en el caso de producir en lotes.

De acuerdo con lo anteriormente descrito, el problema tratado en esta tesis se centra en considerar la necesidad (existencia y dimensionado) de disponer buffers en el PALBP con estaciones multilínea, cuando las líneas trabajan con tiempos de ciclo diferentes; dicho problema se conocerá como PALBP-B.

En este problema de estudio no solo se pretende resolver el problema de equilibrado de líneas, sino, también, el (sub)problema de dimensionado de buffers en las estaciones multilínea. Ambos problemas guardan una fuerte relación, ya que el (sub)problema de dimensionado de buffers parte del conjunto de tareas asignadas a una estación multilínea. Dependiendo del conjunto de tareas asignadas a dicha estación multilínea, la solución del (sub)problema de dimensionado de buffers puede variar, lo que, a su vez, puede influir en la solución del equilibrado de líneas.

2.2 Definición del problema de estudio

El problema de estudio de esta tesis es el PALBP-B (es decir, un sistema PALB con estaciones multilínea, líneas con tiempos de ciclo diferentes y, existencia y dimensionado de buffers) de líneas rectas y modelo único. Al utilizar estaciones multilínea y trabajar con líneas con tiempos de ciclo diferentes, es probable que sea necesario utilizar buffers en las estaciones multilínea, como se demuestra en Pastor et al. (2021). Y, como se ha mencionado anteriormente, en la literatura del PALBP no se manifiesta el uso de buffers cuando se trabaja con este tipo de líneas con tiempos de ciclo diferentes.

El problema tratado en esta tesis (el cual se representa en la Figura 11) se puede dividir en dos (sub)problemas: 1) la asignación de las tareas a las estaciones de las líneas (y la definición como estación regular o multilínea), y 2) dada una asignación de tareas a una estación multilínea, secuenciar y programar dichas tareas con el objetivo de minimizar el tamaño total del buffer en ambas líneas. Idealmente ambos (sub)problemas se deberían de resolver de manera simultánea, aunque actualmente solo existen procedimientos para el resolver (sub)problema de equilibrado de líneas (ver Gökçen et al., 2006; Scholl and Boysen, 2009; Özcan, 2018; entre otros).

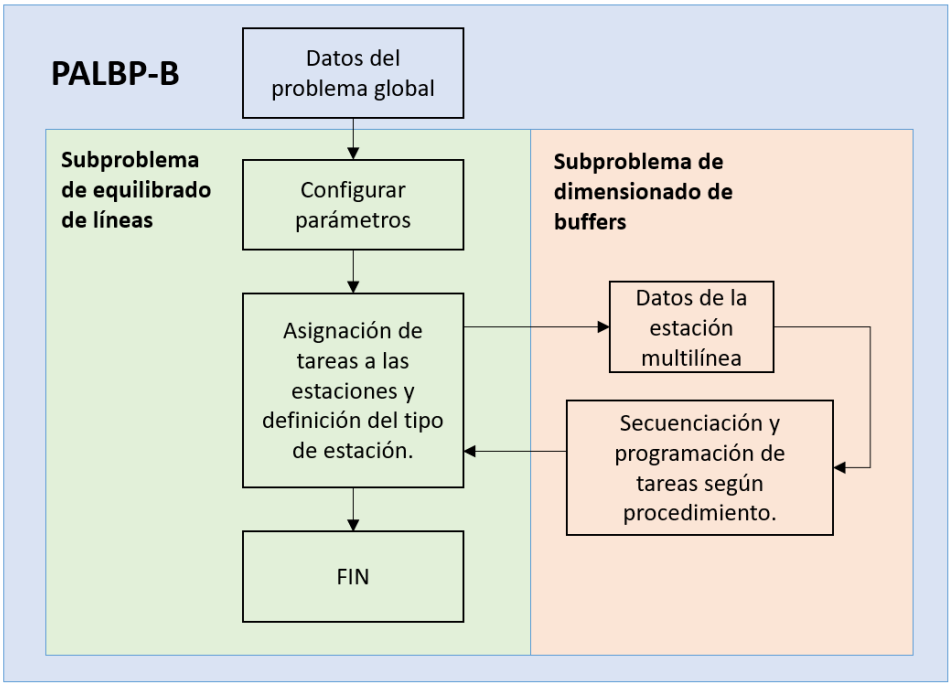

Figura 11. Esquema del PALBP-B.

2.2.1 Definición del PALBP-B

El problema de equilibrado de líneas en paralelo con estaciones multilínea y líneas con tiempos de ciclo diferentes, está compuesto por un conjunto de líneas rectas de modelo único. Las estaciones regulares y multilínea están operadas por un único operario que realiza las tareas asignadas a su estación. Las unidades de producto de cada línea fluyen a través de la línea de una en una y son transportadas entre estaciones al cumplirse el tiempo de ciclo ct_h de la línea h. Por lo tanto, cada tiempo de ciclo ct_h llega y sale una unidad de producto de cada estación de la línea h.

A continuación, se presentan las principales premisas asumidas en esta tesis para el problema de equilibrado de líneas en paralelo con estaciones multilínea y líneas con tiempos de ciclo diferentes:

- El grafo de precedencias de los productos es conocido.
- El número de líneas de montaje en paralelo son 2.
- La asignación de los productos a las líneas es conocido.
- Las líneas de montaje trabajan con tiempos de ciclo ct_h diferentes.
- El tiempo de proceso de las tareas es conocido y determinista.
- Cada tarea solo se puede asignar a una estación.
- Cada estación esta operada por un único operario.
- Los operarios tienen la capacidad de realizar cualquier tarea.
- El tiempo de desplazamiento de los operarios en las estaciones es negligible.
- Se dispone de un sistema adecuado para transportar las unidades de una estación a otra, con unos tiempos de transporte negligibles.
- Solo una unidad de producto es transportada de una estación a otra cada tiempo ct_h de la línea h.

2.2.2 Definición del (sub)problema de dimensionado de buffers en una estación multilínea

El (sub)problema de dimensionado de buffers se centra en una estación multilínea, donde las unidades de los productos son transportadas de una estación a otra (de una en una) al cumplirse el tiempo de ciclo ct_h de la línea h, permitiendo únicamente la entrada y la salida de una unidad en cada puesto de trabajo. Además, todas las tareas de una unidad asignadas a un puesto de trabajo se deben de procesar en el mismo ciclo. Cada estación esta operada por un único operario y este operario solo podrá trabajar en una unidad en cada instante de tiempo, por lo que, si en un puesto de trabajo hay más de una unidad en un instante de tiempo, el excedente de unidades tendrá que almacenarse en el buffer. Cabe destacar que las unidades siguen un orden FIFO (*First In Fisrt Out*) para ser procesadas y para salir de las estaciones.

A continuación, se presentan las principales premisas asumidas en esta tesis para el (sub)problema de dimensionado de buffers en una estación multilínea:

- Se dispone de un sistema formado por 2 líneas en paralelo, entrando y saliendo de cada puesto de trabajo una unidad de producto cada tiempo de ciclo ct_h de la línea h.
- Las líneas trabajan con tiempos de ciclo ct_h diferentes.
- El conjunto de tareas asignadas a la estación multilínea es conocido.
- Todas las tareas de una unidad asignadas a un puesto de trabajo se deben de procesar en el mismo ciclo c_h. de la línea h.

- Todas las tareas de todas las unidades de ambas líneas deben de procesarse dentro del tiempo de ciclo común lcm de las líneas. El número de unidades q_h (donde $q_h = lcm/\ ct_h$) que se deben de realizar en una línea h, es igual al total de ciclos que transcurren en la línea h en un periodo lcm (o aquí llamado superciclo).
- En un ciclo c_h de la línea h, se pueden realizar 0, 1, 2 o más unidades de un producto (siempre que los tiempos de procesos lo permitan).
- Se dispone de un sistema adecuado para mover las unidades de la línea al buffer y viceversa, en unos tiempos de transporte negligibles.

2.3 Ejemplo ilustrativo

Para visualizar la problemática planteada, a continuación se presenta un ejemplo del problema PALB-B.

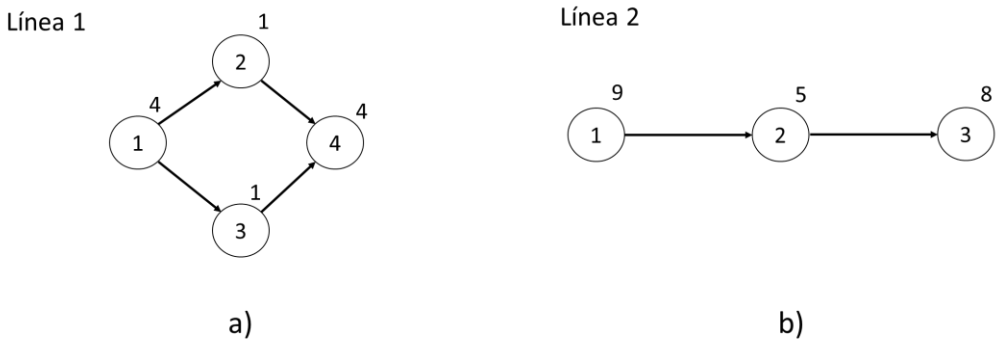

Figura 12. a) grafo de precedencias del producto A, b) grafo de precedencias del producto B.

La Figura 12 muestra los grafos de precedencias de dos productos A y B, asignados a las líneas L1 y L2, respectivamente. El número dentro del nodo hace referencia a la tarea y el número superior derecho a su tiempo de proceso.

En el presente ejemplo se asume que el tiempo de ciclo ct_h de la línea h ($h = 1, \dots, H$) es: $ct_1 = 4$ para línea 1 y $ct_2 = 10$ para línea 2.

Al utilizar el procedimiento LCM, se obtiene un tiempo común para las dos líneas $lcm = m.c.m.\{4, 10\} = 20$, con lotes $q_1 = \frac{20}{4} = 5$ unidades y $q_2 = \frac{20}{10} = 2$ unidades. Los tiempos de las tareas del producto asignado a la línea 1 se normalizan a: $t'_{1,1} = 4 \cdot 5 = 20$; $t'_{1,2} = 1 \cdot 5 = 5$; $t'_{1,3} = 1 \cdot 5 = 5$; $t'_{1,4} = 4 \cdot 5 = 20$. Los tiempos de las tareas del producto asignado a la línea 2 se normalizan a: $t'_{2,1} = 9 \cdot 2 = 18$; $t'_{2,2} = 5 \cdot 2 = 10$; $t'_{2,3} = 8 \cdot 2 = 16$.

En el tiempo de ciclo $lcm = 20$ (o superciclo), se deben procesar 5 unidades de producto A en línea 1, y 2 unidades de producto B en línea 2.

La Figura 13 muestra una solución óptima para PALBP, con 5 estaciones y operarios, de las cuales 4 son estaciones regulares y 1 es una estación multilínea (estación 3). El número dentro del puesto de trabajo hace referencia a la tarea(s) asignada.

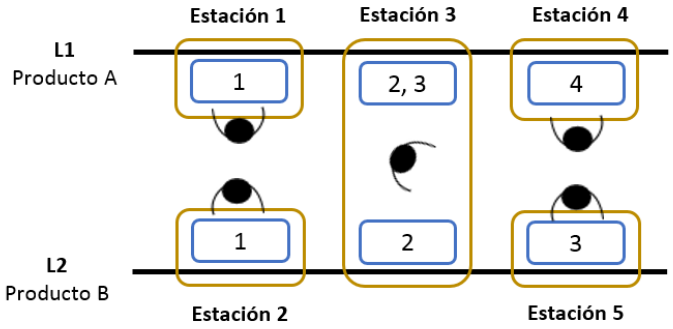

Figura 13. Representación de una solución óptima del ejemplo de la Figura 12 al utilizar el PALB con estaciones multilínea.

Como se puede observar, la estación multilínea 3 tiene asignadas la tarea 2 y 3 de línea 1 y la tarea 2 de línea 2, con tiempos de proceso $t_{1,2} = 1$, $t_{1,3} = 1$ y $t_{2,2} = 5$, respectivamente. Se recuerda que los tiempos de procesos normalizados t'_{hi} solo se utilizan para realizar el equilibrado de las líneas, para el (sub)problema de dimensionado de buffers se utilizan los tiempos de procesos estándar t_{hi}.

La Figura 14 muestra una propuesta de solución para dividir el tiempo de trabajo del operario entre ambas líneas. Se recuerda que en la figura se puede observar: cuando el operario está realizando las tareas de una unidad (recuadros grises) de la línea L1 y de la L2, así como el instante en que empieza y finaliza cada tarea; cuando una unidad aún no ha sido procesada (recuadros naranjas); y, finalmente, cuando ya ha sido procesada (recuadros verdes). Las unidades procesadas y no procesadas pueden permanecer en su línea (L1 o L2) o en el buffer de sus respectivas líneas (Buffer L1 o Buffer L2). Las unidades se representan con el símbolo u (o con el símbolo $u0$, para aquellas unidades que ya hay en los buffers al inicio del superciclo, es decir, instante 0 aquí representado) acompañadas de un número que hace referencia a la unidad del lote (valor entre 1 y q_h). Las tareas se representan con el símbolo T y acompañadas de un número (índice i) que hace referencia a la tarea en cuestión. Recuérdese que las unidades siguen un orden FIFO para ser procesadas y para salir de las estaciones.

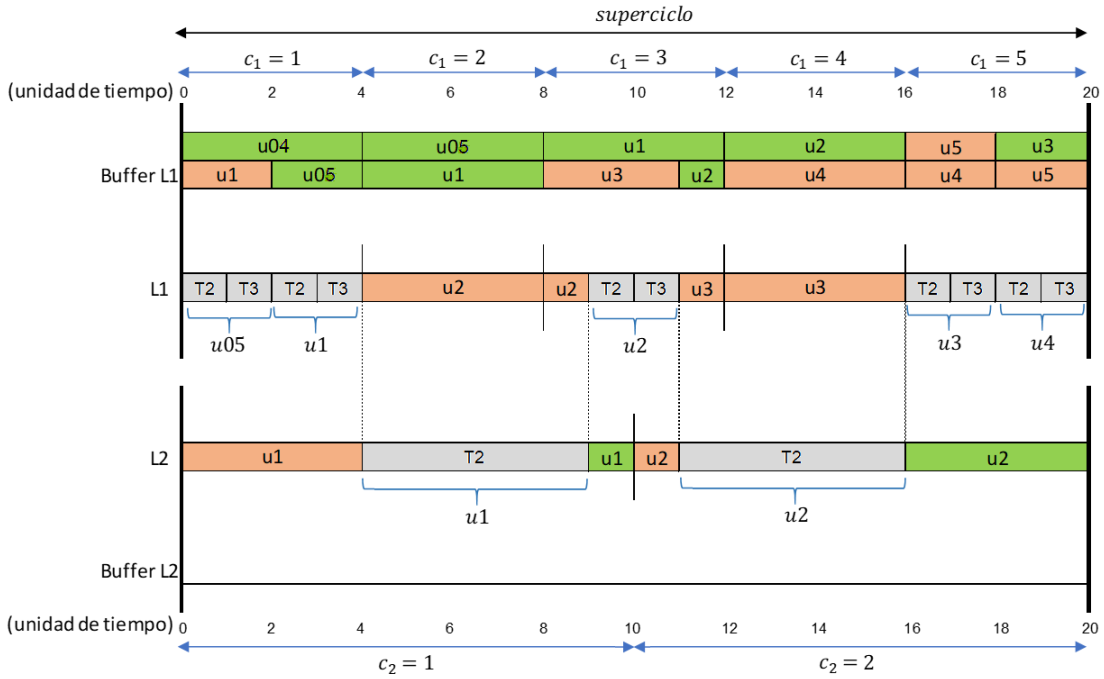

Figura 14. Solución para la estación multilínea 3 del ejemplo propuesto en la Figura 13.

A continuación se detalla la propuesta de solución representada en la Figura 14.

- Unidad $u04$: la unidad ya procesada (en el superciclo anterior) permanece en el Buffer L1 (en verde) desde el inicio del superciclo (instante 0) hasta el instante 4 que pasa a la estación siguiente de L1.
- Unidad $u05$: unidad no procesada (procedente del superciclo anterior y en la que, recuérdese, se han de procesar las tareas T2 y T3) permanece en línea L1 y es procesada desde el inicio del superciclo (instante 0) hasta el instante 2 que se guarda en Buffer L1 (en verde), hasta el instante 8 que pasa a la siguiente estación.
- Unidad $u1$: llega a la estación al inicio del superciclo (instante 0) y se envía directamente al Buffer L1 (en naranja); entre los instantes 2 y 4 (en gris) pasa a línea L1 para ser procesada; cuando termina de ser procesada (instante 4) se envía al Buffer L1 (en verde) hasta el instante 12 que pasa a la estación siguiente de L1.
- Unidad $u2$: llega a la estación en el instante 4 y permanece en la línea (en naranja) hasta que es procesada entre los instantes 9 y 11 (en gris). Una vez procesada, se envía al Buffer L1 (en verde) hasta el instante 16 que pasa a la estación siguiente de L1.
- Unidad $u3$: llega a la estación en el instante 8 y se guarda en el Buffer L1 (en naranja) hasta el instante 11 que pasa a línea y se espera en L1; entre los

instantes 16 y 18 es procesada (en verde); una vez procesada (instante 18) se guarda en Buffer L1 hasta el instante 20 que pasa a la estación siguiente de L1.

- Unidad $u4$: llega a la estación en el instante 12 y se guarda en el Buffer L1 (en naranja) hasta el instante 18 que pasa a línea y es procesada hasta el instante 20. Al terminar de ser procesada se guarda en el Buffer L1 (en verde) hasta el instante 4 (del siguiente superciclo), que pasa a la estación siguiente de L1. Se recuerda que la unidad $u4$ del actual superciclo es la unidad $u04$ ya procesada que hay en el Buffer L1 al inicio del siguiente superciclo.
- Unidad $u5$: llega a la estación en el instante 16 y se guarda en el Buffer L1 (en naranja) hasta el instante 0 (inicio del siguiente superciclo), que pasa a línea para ser procesada (en gris) hasta el instante 2 (del siguiente superciclo), que se guarda en el Buffer L1 (en verde) hasta el instante 8 (del siguiente superciclo) que pasa a la siguiente estación de L1. Se recuerda que la unidad $u5$ del actual superciclo es la unidad $u05$ no procesada que hay en la línea L1 al inicio del siguiente superciclo.

La secuencia en línea 2 es la siguiente:

- Unidad $u1$: llega a la estación al inicio del superciclo (instante 0) y se espera (en naranja) en línea L2 hasta que es procesada entre los instantes 4 y 9 (en gris), permanece procesada en L2 entre 9 y 10 (en verde), hasta el instante 10 que pasa a la estación siguiente de L2.
- Unidad $u2$: llega a la estación en el instante 10, y se espera (en naranja) en línea L2 hasta que es procesada entre los instantes 11 y 16 (en gris), permanece procesada en L2 entre 16 y 20 (en verde), hasta el instante 20 que pasa a la estación siguiente de L2.

De la solución de la Figura 14 se observa la necesidad de disponer de un buffer de dos unidades de capacidad en la línea 1.

Capítulo 3. Justificación y objetivos

3.1 Justificación

En el capítulo anterior se pone de manifiesto un problema existente en el PALBP con estaciones multilínea, al trabajar con líneas con tiempos de ciclo diferentes. Mediante un ejemplo se demuestra la necesidad de considerar (existencia y dimensionado) el uso de buffers en las estaciones multilínea, y la importancia de la secuenciación y programación de las tareas asignadas a dicha estación. Además, tras un análisis del estado del arte del PALBP, se constata que no hay ningún estudio, hasta nuestro conocimiento, que manifieste el uso de buffers al trabajar con líneas con tiempos de ciclo diferentes. No considerar el uso de buffers en las estaciones multilínea puede hacer, como ya se ha demostrado en el capítulo anterior, que soluciones académicas propuestas hasta ahora no puedan ser implementadas en un sistema industrial real.

A raíz de lo expuesto anteriormente, se justifica la razón de ser de la presente tesis doctoral.

3.2 Objetivos

El objetivo general de esta tesis es proponer y formalizar un nuevo problema inédito, aquí definido como PALBP-B, además de desarrollar procedimientos para resolverlo.

El problema de estudio se enfoca en el PALB de líneas rectas y modelo único, con líneas de montaje con tiempos de ciclo diferentes, donde las unidades de los productos son transportadas de una estación a otra al cumplirse el tiempo de ciclo, permitiendo únicamente la entrada y la salida de una unidad de producto en cada puesto de trabajo.

La resolución de este problema implica: 1) resolver el (sub)problema de equilibrado de líneas; y 2) resolver el (sub)problema de dimensionado de buffer en una estación multilínea.

Para alcanzar el objetivo principal, se marcan los siguientes subobjetivos específicos:

1. Analizar y clasificar los estudios realizados en la literatura PALBP. Como resultado se realiza una revisión del estado del arte, donde se exponen los principales aspectos que no han sido contemplados o que su consideración es escasa en la literatura.

2. Definir y caracterizar el PALBP-B: definir el (sub)problema equilibrado de líneas en paralelo con estaciones multilínea (PALBP) y el (sub)problema de dimensionado de buffers en una estación multilínea.

3. Formulación matemática del (sub)problema de dimensionado de buffers en una estación multilínea. Se desarrolla, por primera vez en la literatura, un modelo de programación matemática para la resolución óptima de ejemplares de tamaños pequeño y mediano del (sub)problema de dimensionado de buffers.

4. Diseñar e implementar, por primera vez en la literatura del PALBP, procedimientos heurísticos y metaheurístico para la resolución del (sub)problema de dimensionado de buffers en una estación multilínea. El desarrollo de procedimientos no exactos permite abordar ejemplares de tamaño realista.

5. Generación de ejemplares para el (sub)problema de dimensionado de buffers en una estación multilínea. Debido a que el (sub)problema de dimensionado de buffers no ha sido tratado en la literatura del PALBP, es necesario generar una serie ejemplares del problema.

6. Evaluar el rendimiento de los procedimientos presentados para el (sub)problema de dimensionado de buffers.

7. Diseñar e implementar métodos basados en heurísticas y metaheurísticas para la resolución del problema conjunto de equilibrado de líneas y dimensionado de buffers, PALBP-B. Debido a la naturaleza NP-difícil del PALBP, es recomendable el desarrollo de procedimientos no exactos para resolver ejemplares de tamaños reales del PALBP-B. Los procedimientos desarrollados resuelven el (sub)problema de equilibrado de líneas y a su vez ejecutan el/los procedimientos realizados para la resolución del (sub)problema de dimensionado de buffers en una estación multilínea.

8. Generación de ejemplares para el PALBP-B. Debido a que el problema de estudio de esta tesis no ha sido considerado en la literatura, es recomendable generar una serie ejemplares del problema. Los ejemplares para PALBP-B se basan en ejemplares realistas de la literatura.

9. Evaluar el rendimiento de los procedimientos presentados para el PALBP-B.

Capítulo 4. Métodos de resolución del (sub)problema de dimensionado de buffers en una estación multilínea

4.1 Introducción

Como se ha presentado en el estado del arte (Sección 1.3), la existencia de buffers no ha sido considerada en los sistemas PALB al trabajar con líneas con tiempos de ciclo diferentes. En la Sección 2.3 se ha demostrado la necesidad de considerar el uso de buffers en las estaciones multilínea cuando se trabaja con líneas con tiempos de ciclo diferentes. Además, se ha puesto de manifiesto la importancia de la secuenciación y programación de las tareas en relación con el tamaño de los buffers.

La novedad del (sub)problema de dimensionado de buffers en una estación multilínea, conlleva que se tenga que formalizar y desarrollar métodos para resolver el (sub)problema presentado y definido en la Subsección 2.2.2.

Los métodos aquí propuestos tienen como objetivo minimizar el tamaño total de los buffers (en ambas líneas) en una estación multilínea. De la resolución aportada por lo métodos se obtendrá la secuenciación y programación de las tareas asignadas a una estación multilínea, así como el tamaño del buffer de cada línea.

En este capítulo se presenta un modelo de programación matemática para la formalización y resolución óptima del (sub)problema de dimensionado de buffers (Subsección 4.2.1). El desarrollo del modelo tiene como objetivo observar si el problema puede ser resuelto de manera óptima en un tiempo límite viable. A partir del modelo matemático original se ha desarrollado una variante (Subsección 4.2.2) con el objetivo de incrementar el número de ejemplares que pueden ser solucionados de manera óptima respecto al modelo original, dentro en un tiempo límite. También, se ha realizado una modificación en un parámetro del software de resolución utilizado (Subsección 4.2.3) con el objetivo de reducir el tiempo de cálculo necesario para obtener una solución óptima respecto al modelo original, dentro de un tiempo límite.

Debido a la dificultad de obtener una solución en ejemplares que, por sus dimensiones, no pueden ser resueltos de manera exacta en un tiempo aceptable mediante los modelos matemáticos, en la Sección 4.3, 4.4 y 4.5, se presentan diferentes métodos no exactos para su resolución aproximada. Con el objetivo de solucionar ejemplares de un tamaño real, en la Sección 4.3 se han desarrollado dos heurísticas, y en la Sección 4.4. se ha diseñado un cóctel de heurísticas. Para mejorar la calidad de las soluciones, en

la Sección 4.5 se presenta un procedimiento basado en la metaheurística de recocido simulado.

Todos los métodos propuestos en este capítulo han sido evaluados y comparados entre sí vía un experimento computacional. Los resultados obtenidos y su análisis se reportan al final de cada Sección 4.2, 4.3, 4.4 y 4.5. Las conclusiones del capítulo se presentan en la Sección 4.6.

4.2 Modelos de programación matemática

El modelo de programación lineal entera mixta (PLEM) presentado esta sección, se ha desarrollado con el objetivo de formalizar y resolver de manera óptima el (sub)problema de dimensionado de buffer en una estación multilínea (definido en la Subsección 2.2.2).

Como se introdujo anteriormente, para un conjunto de tareas dado, el modelo de PLEM programa las tareas asignadas a una estación multilínea con el objetivo de minimizar el tamaño total de los buffers (en ambas líneas) en dicha estación. De la resolución del modelo de PLEM se obtiene el tamaño del buffer para cada línea de la estación, así como la secuenciación y programación de las tareas.

En las siguientes secciones se presenta el modelo de PLEM inicial, conocido como V1, una variante derivada de éste y una modificación en un parámetro del software de resolución del modelo V1.

El modelo V1 y su formalización se presentan en la Subsección 4.2.1. En la Subsección 4.2.2 se presenta la variante conocida como V2. La principal transformación en V2 consiste en la sustitución de unas variables binarias respecto al modelo original. En la Subsección 4.2.3 se presenta una modificación en un parámetro del software de resolución del modelo V1. Esta modificación se conocerá como V3.

En la Subsección 4.2.4 se realizan dos experimentos computacionales para evaluar el rendimiento de los modelos matemáticos presentados. En el primer experimento, los modelos tienen que resolver una serie de ejemplares dentro de un tiempo de cálculo límite fijado para cada ejemplar. En el segundo experimento, para un valor del tamaño del buffer predeterminado, los modelos tienen que resolver (encontrar una solución factible si existe) una serie de ejemplares dentro de un tiempo de cálculo límite fijado para cada ejemplar.

4.2.1 Modelo de PLEM V1

En este apartado se presenta el modelo de PLEM desarrollado para la formalización y resolución óptima del (sub)problema de dimensionado de buffers en una estación multilínea. El modelo de PLEM aquí desarrollado se conocerá como V1.

Cabe destacar que en la literatura no existía un modelo de programación matemática para el (sub)problema de dimensionado de buffers planteado en esta tesis, por lo que el modelo que se presenta a continuación es el primero en formalizar matemáticamente el problema de estudio.

Se recuerda que, dada una asignación de tareas a la estación multilínea, el siguiente modelo secuencia y programa las tareas con el objetivo de minimizar el tamaño total de los buffers (en ambas líneas) en una estación multilínea.

Datos:

H Conjunto de las líneas: $H = \{1,2\}$

ct_h Tiempo de ciclo de la línea h ; $h \in H$

lcm Tiempo de ciclo de todas las líneas: $lcm = m.c.m.\{ct_1, ct_2\}$

q_h Número de unidades a producir del producto de la línea h durante el tiempo lcm : $q_h = lcm / ct_h$; $h \in H$

T_h Conjunto de las tareas asignadas a la estación multilínea del producto de la línea h ; $h \in H$

t_{hi} Tiempo de proceso de la tarea i del producto de la línea h ; $h \in H$; $i \in T_h$

IP_{hi} Conjunto de las predecesoras inmediatas de la tarea i del producto de la línea h que están asignadas a la estación multilínea; $h \in H$; $i \in T_h$

IS_{hi} Conjunto de las sucesoras inmediatas de la tarea i del producto de la línea h que están asignadas a la estación multilínea; $h \in H$; $i \in T_h$

P_{hi} Conjunto de las predecesoras totales de la tarea i del producto de la línea h que están asignadas a la estación multilínea: $P_{hi} = IP_{hi} \cup \left(\bigcup_{j \in IP_{hi}} P_{hj} \right)$; $h \in H$; $i \in T_h$

$\bar{P}_h$ Conjunto de las parejas de tareas del producto de la línea h entre las que no hay relaciones de precedencia: $\bar{P}_h = \left\{ \left(i \in T_h, j \in T_h \right) | \left(i < j \right) \wedge \left(i \notin P_{hj} \right) \wedge \left(j \notin P_{hi} \right) \right\}$; $h \in H$

S_{hi} Conjunto de las sucesoras totales de la tarea i del producto de la línea h que están asignadas a la estación multilínea: $S_{hi} = \left\{ j \in T_h \mid i \in P_{hj} \right\}$; $h \in H$; $i \in T_h$

ec_{hu} Cota inferior del primer ciclo en que se podría tratar la u-ésima unidad del producto de la línea h: $ec_{hu} = \left\lceil \frac{u \cdot \sum_{i \in T_h} t_{hi}}{ct_h} \right\rceil$; $h \in H$; $u = 1,...,q_h$

lc_{hu} Cota superior del primer ciclo en que se podría tatar la u-ésima unidad del producto de la línea h: $lc_{hu} = q_h + 1 - \left\lceil \frac{\left(q_h - u + 1 \right) \cdot \sum_{i \in T_h} t_{hi}}{ct_h} \right\rceil$; $h \in H$; $u = 1,...,q_h$

U_{hc} Conjunto de las unidades del producto de la línea h que podrían producirse en el ciclo c: $U_{hc} = \left\{ u = 1,...,q_h \mid ec_{hu} \le c \le lc_{hu} \right\}$; $h \in H$; $c = 1,...,q_h$

est_{hui} Cota inferior del primer instante en que se podría comenzar a realizar la tarea i de la u-ésima unidad tratada del producto de la línea h: $est_{hui} = ct_h \cdot \left(ec_{hu} - 1 \right) + \sum_{j \in P_{hi}} t_{hj}$; $h \in H$; $u = 1,...,q_h$; $i \in T_h$

lst_{hui} Cota superior del primer instante en que se podría comenzar a realizar la tarea i de la u-ésima unidad tratada del producto de la línea h: $lst_{hui} = ct_h \cdot lc_{hu} - t_{hi} - \sum_{j \in S_{hi}} t_{hj}$; $h \in H$; $u = 1,...,q_h$; $i \in T_h$

Variables:

$x_{huc} \in \{0,1\}$ 1 si la u-ésima unidad del producto de la línea h es tratada en el ciclo c; $h \in H$; $u = 1,...,q_h$; $c = ec_{hu},...,lc_{hu}$

$y_{huij} \in \{0,1\}$ 1 si la tarea i se realiza antes que la tarea j en la u-ésima unidad tratada del producto de la línea h; $h \in H$; $u = 1,...,q_h$; $(i,\mathrm{j}) \in \bar{P}_h$

$w_{uivj} \in \{0,1\}$ 1 si la tarea i de la u -ésima unidad tratada del producto de la línea 1 se realiza antes que la tarea *j* de la v -ésima unidad tratada del producto de la línea 2; $u = 1,...,q_1$; $i \in T_1$; $v = 1,...,q_2$; $j \in T_2 \mid \left(est_{1ui} < lst_{2vj} + t_{2j}\right) \wedge \left(lst_{1ui} + t_{1i} > est_{2vj}\right)$

$st_{hui} \in \mathbb{R}^+$ Instante en que comienza a realizarse la tarea i de la u -ésima unidad tratada del producto de la línea h ; $h \in H$; $u = 1,...,q_h$; $i \in T_h$. Esta variable se puede acotar: $est_{hui} \le st_{hui} \le lst_{hui}$

$z_h \in \mathbb{Z}^+$ Número de unidades tratadas del producto de la línea h que hay inicialmente en el buffer; $h \in H$. Esta variable se puede acotar: $0 \le z_h \le q_h - 1$

$a_h \in \mathbb{Z}^+$ Número de unidades del producto de la línea h que hay en el buffer; $h \in H$. Esta variable se puede acotar: $0 \le a_h \le q_h - 1$

Función objetivo:

$$[\text{MIN}] \sum_{\text{h}\in H} a_h \tag{7}$$

Sujeto a:

$$a_h \ge z_h + \sum_{\tau=1}^{c} \left(\sum_{u \in U_{h\tau}} x_{hu\tau} - 1 \right) \qquad h \in H \ ; \ c = 1,...,q_h \tag{8}$$

$$z_h + \sum_{\tau=1}^{c} \left(\sum_{u \in U_{h\tau}} x_{hu\tau} - 1 \right) \ge 0 \qquad h \in H \ ; \ c = 1,...,q_h \tag{9}$$

$$st_{hui} \ge ct_h \cdot \sum_{c=ec_{hu}}^{lc_{hu}} \left(c-1\right) \cdot x_{huc} \qquad h \in H \ ; \ u = 1,...,q_h \ ; \ i \in T_h \mid IP_{hi} = \varnothing \tag{10}$$

$$st_{hui} + t_{hi} \le ct_h \cdot \sum_{c=ec_{hu}}^{lc_{hu}} c \cdot x_{huc} \qquad h \in H \ ; \ u = 1,...,q_h \ ; \ i \in T_h \mid IS_{hi} = \varnothing \tag{11}$$

$$st_{hui} \ge st_{huj} + t_{hj} \qquad h \in H \ ; \ u = 1,...,q_h \ ; \ i \in T_h \ ; \ j \in IP_{hi} \tag{12}$$

$$st_{hui} \ge st_{huj} + t_{hj} - lcm \cdot y_{huij} \qquad h \in H \ ; \ u = 1,...,q_h \ ; \ \left(i,\text{j}\right) \in \bar{P}_h \tag{13}$$

$$st_{huj} \ge st_{hui} + t_{hi} - lcm \cdot \left(1 - y_{huij}\right) \qquad h \in H \ ; \ u = 1,...,q_h \ ; \ \left(i,\text{j}\right) \in \bar{P}_h \tag{14}$$

$$st_{1ui} \geq st_{2vj} + t_{2j} - lcm \cdot w_{uivj} \qquad u = 1,...,q_1 \;;\; i \in T_1 \;;\; v = 1,...,q_2 \;;$$

$$j \in T_2 \,|\, \left(est_{1ui} < lst_{2vj} + t_{2j} \right) \wedge \left(lst_{1ui} + t_{1i} > est_{2vj} \right) \qquad (15)$$

$$st_{2vj} \geq st_{1ui} + t_{1i} - lcm \cdot \left(1 - w_{uivj} \right) \qquad u = 1,...,q_1 \;;\; i \in T_1 \;;\; v = 1,...,q_2 \;;$$

$$j \in T_2 \,|\, \left(est_{1ui} < lst_{2vj} + t_{2j} \right) \wedge \left(lst_{1ui} + t_{1i} > est_{2vj} \right) \qquad (16)$$

$$st_{huj} \geq st_{h,u-1,i} + t_{hi} \qquad h \in H \;;\; u = 2,...,q_h \;;\; i \in T_h \,|\, IS_{hi} = \varnothing \;;\; j \in T_h \,|\, IP_j = \varnothing \qquad (17)$$

- Ec. 7: Minimiza el tamaño total de los buffers.
- Ec. 8 y 9: Determinan el número de unidades de cada producto en el buffer.
- Ec. 10 y 11: Todas las tareas de una unidad se realizan durante el mismo tiempo de ciclo, y relacionan las variables x_{huc} y st_{hui}.
- Ec. 12: Se respetan las relaciones de precedencias.
- Ec. 13 y 14: Dos tareas de la misma unidad no se pueden realizar simultáneamente.
- Ec. 15 y 16: Dos tareas de productos diferentes no se pueden realizar simultáneamente.
- Ec. 17: Una unidad de un producto no puede comenzar a tratarse hasta que no haya acabado de tratarse la unidad anterior de ese producto.

4.2.2 Modelo de PLEM V2

En esta subsección se presenta una variante del modelo matemático inicial conocida como V2. La principal diferencia entre los modelos PLEM V1 y V2, radica que en V2 se han sustituido las variables binarias y_{huij} (que indican si la tarea *i* se realiza antes que la tarea *j* de la *u*-ésima unidad del producto de la línea h) por las variables y_{hij} (que indican si la tarea *i* se realiza antes que la tarea *j*, para toda unidad del producto de la línea h); de esta forma se fija la misma secuencia de tareas para todas las unidades que se realizan en la línea h en un ciclo lcm. Con esta variante se desea analizar si fijar la misma secuencia de tareas en todas las unidades de la línea h, facilita la resolución del programa matemático. Sin embargo, cabe destacar, que al restringir más el problema (fijar la misma secuencia de tareas en todas las unidades), las soluciones óptimas obtenidas por V2 no podrán ser mejores que las obtenidas por V1.

Al modificar la variable binaria, el modelo de PLEM V2 queda de la siguiente forma, donde las ecuaciones 13 y 14 (del modelo V1) son sustituidas por las ecuaciones 18 y 19, respectivamente.

$$st_{hui} \geq st_{huj} + t_{hj} - lcm \cdot y_{hij} \qquad h \in H\ ;\ u = 1,...,q_h\ ;\ (i,\mathrm{j}) \in \bar{P}_h \qquad (18)$$

$$st_{huj} \geq st_{hui} + t_{hi} - lcm \cdot (1 - y_{hij}) \qquad h \in H\ ;\ u = 1,...,q_h\ ;\ (i,\mathrm{j}) \in \bar{P}_h \qquad (19)$$

A continuación, se muestra un ejemplo presentado con el objetivo de desmentir que siempre existe una solución óptima con la misma secuencia para todas las unidades.

En la Figura 15, se muestran los grafos de precedencias del producto de la línea 1 (Figura 15a) y del producto de la línea 2 (Figura 15b). El número dentro del nodo hace referencia a la tarea y el número superior derecho su tiempo de proceso.

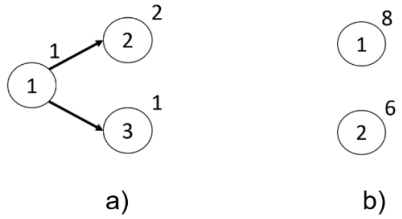

Figura 15. a) grafo de precedencias del producto de la línea 1, b) grafo de precedencias del producto de la línea 2.

Se asumen que los tiempos de ciclos para la línea 1 y 2 son, $ct_1 = 8$ y $ct_2 = 28$, respectivamente. El tiempo común para ambas líneas es $lcm = m.c.m.\{8, 28\} = 56$, con lotes $q_1 = 56/8 = 7$; $q_2 = 56/28 = 2$.

Los resultados óptimos obtenidos son los siguientes**:**

- Modelo matemático V1: 0 unidades de buffer.
  $a_1 = 0$; $a_2 = 0$;
- Modelo matemático V2: 1 unidad de buffer en L1.
  $a_1 = 1$; $a_2 = 0$;

En la Figura 16 se representa la solución óptima obtenida por el modelo V1 (posibilidad de secuencias de tareas diferentes para cada unidad).

Recuérdese que en dicha figura se puede observar cuándo el operario está realizando las tareas de una unidad (recuadros grises) de la línea L1 y de la L2, así como el instante en que empieza y finaliza cada tarea; y cuándo una unidad no ha sido procesada (recuadros naranjas) o ya ha sido procesada (recuadros verdes). Las unidades procesadas y no procesadas pueden permanecer en su línea (L1 o L2) o en el buffer de sus respectivas líneas (Buffer L1 o Buffer L2). Las unidades se representan con el símbolo u (o con el símbolo $u0$, para aquellas que ya hay en los buffers al inicio del superciclo, es decir, instante 0, proveniente del superciclo anterior) acompañadas de un

número que hace referencia a la unidad del lote (valor entre 1 y q_h). Las tareas se representan con el símbolo T y acompañadas de un número (índice i) que hace referencia a la tarea en cuestión.

Como se ha especificado, en cada puesto de trabajo y en cada tiempo de ciclo ct_h, entra una unidad sin procesar y sale una procesada de cada línea h; en L1 en los instantes 0, 8, 16, 24, 32, 40, 48 y 56 (instante 0 del siguiente superciclo), y en L2 en los instantes 0, 28 y 56. En la Figura 16, se puede observar cómo la secuencia de las tareas de la unidad 1 ($u1$) y 2 ($u2$) del producto de la línea 2 son diferentes. Por otra parte, la secuencia de las tareas de las unidades de línea L1 es la misma. Con esta secuencia de tareas se demuestra que no es necesario el uso de buffers ni en L1 ni en L2.

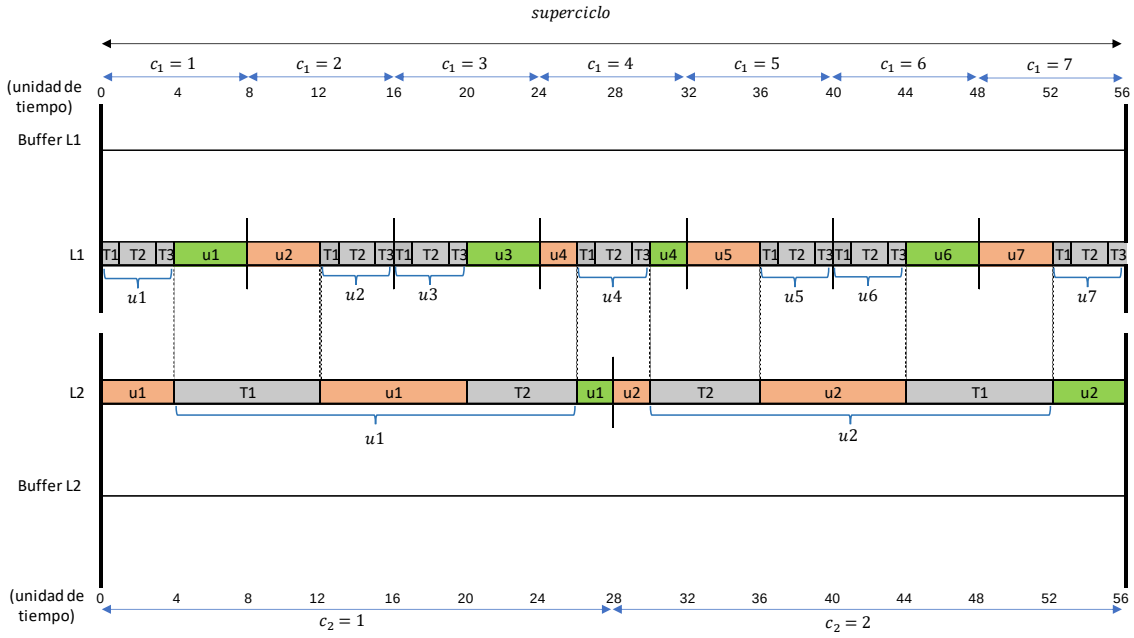

Figura 16. Representación de la solución obtenida por el modelo V1.

La Figura 17 muestra la solución óptima del modelo V2 (misma secuencia obligatoriamente para todas las unidades). De la solución obtenida por el modelo V2, se observa que la secuencia de las tareas (que no programación) de las unidades de línea L1, es la misma que la obtenida por el modelo V1 (Figura 16). Sin embargo, la secuencia de las tareas de las unidades de la línea L2 son diferentes para ambos modelos, V1 y V2. La solución del modelo V2 muestra que en línea L1 se debe de disponer de un buffer de tamaño 1 unidad ($a_1 = 1$). Además, habrá al menos una pieza terminada en el buffer de L1 en los ciclos 2 y 3, y en parte del ciclo 1; y habrá una unidad sin procesar ($u07$) en la línea L1 al inicio del superciclo, instante 0. La unidad $u07$ es la primera unidad en ser procesada, desde el instante 0 al instante 4. Se puede observar que la unidad $u7$

(no procesada) de L1 será la unidad no procesada $u07$ que hay en la línea de L1 al inicio del siguiente superciclo. Por otra parte, en la línea L2 no es necesario el uso de buffers.

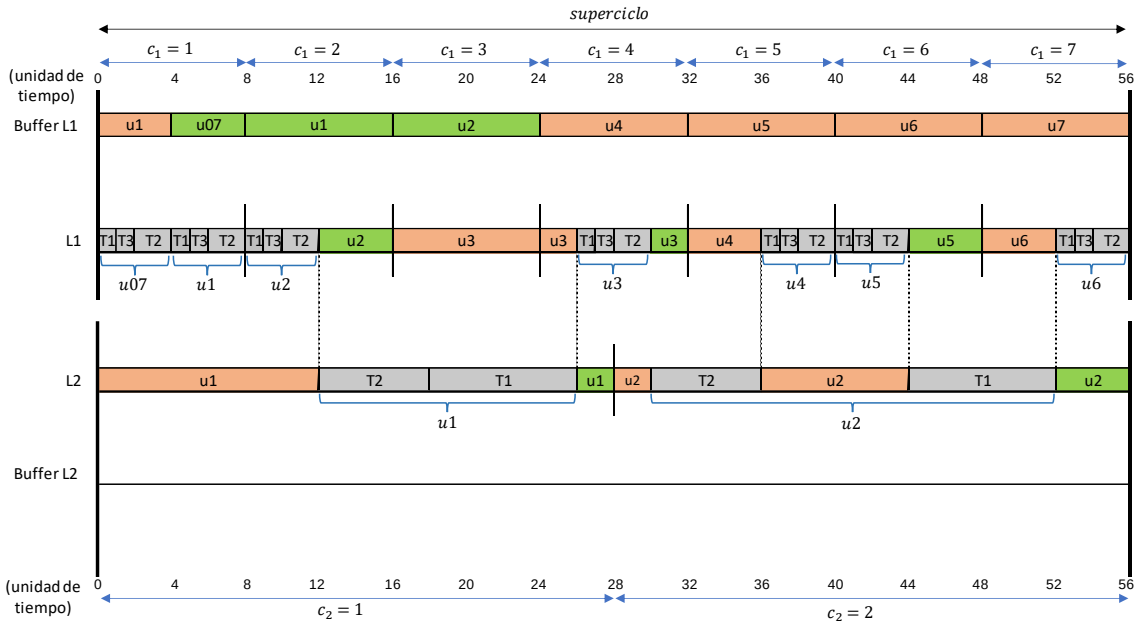

Figura 17. Representación de la solución obtenida por el modelo V2.

De los resultados obtenidos se puede observar que el óptimo del ejemplar a resolver solo se obtiene con secuencias diferentes para cada unidad (modelo V1); en cambio, con secuencias iguales para todas las unidades (modelo V2) se obtiene una solución no óptima. De esta forma se comprueba que no siempre existe una solución óptima con la misma secuencia para todas las unidades.

4.2.3 Modelo de PLEM V3

En este apartado se presenta una modificación en un parámetro del software de resolución respecto al modelo de PLEM V1, esta modificación se conocerá como V3. En V3 se ha modificado, en el software de optimización utilizado (IBM ILOG CPLEX Versión 12.9.0.0), el valor del parámetro *absolute MIP gap tolerance* (GAP), fijándolo a GAP = 0.9999. Anteriormente, en V1, este parámetro tenía un valor por defecto de GAP = 10^{-6}.

El objetivo de esta modificación es analizar si se reduce el tiempo de cálculo necesario para obtener una solución óptima demostrada del ejemplar a resolver. El parámetro GAP establece una tolerancia absoluta entre la diferencia de la mejor solución encontrada hasta el momento y el valor del mejor nodo restante por explorar; cuando esta diferencia es inferior al valor de GAP, el proceso de optimización finaliza. Dado que el valor de la función objetivo siempre es entero, el nuevo GAP utilizado continúa asegurando que la solución obtenida es óptima.

4.2.4 Experimento computacional

Para evaluar y comparar el rendimiento de los modelos de PLEM propuestos, en este apartado se realizan dos experimentos computacionales.

Debido a la novedad del (sub)problema de dimensionado de buffers en una estación multilínea, no existe ningún recopilatorio de ejemplares. De esta forma, se han generado 120 ejemplares como se muestra en la Subsección 4.2.4.1. Los ejemplares aquí generados también son utilizados en los experimentos computacionales de las siguientes secciones de este capítulo.

En el primer experimento (Subsección 4.2.4.2), los modelos tienen que resolver un conjunto de ejemplares dentro de un tiempo de cálculo límite fijado para cada ejemplar.

En el segundo experimento (Subsección 4.2.4.3), para un valor del tamaño del buffer a_h dado para cada línea h, los modelos tienen que resolver (encontrar si para los valores a_h dados existe una solución factible o no) una serie de ejemplares, dentro de un tiempo de cálculo límite fijado para cada ejemplar. El objetivo de este experimento es analizar si los modelos son capaces de encontrar en un breve periodo de tiempo una solución (factible u óptima) o no, para un valor del buffer predeterminado para cada ejemplar.

Los modelos de programación matemática han sido probados utilizando el software de optimización IBM ILOG CPLEX Versión 12.9.0.0. Todos los modelos han sido testados en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

4.2.4.1 Generación de ejemplares

Como se ha introducido anteriormente, para la realización de los experimentos computacionales del (sub)problema de dimensionado de buffers en una estación multilínea, se han generado un total de 120 ejemplares a partir de 8 grafos de precedencias mostrados en la Figura 18, Figura 19, Figura 20 y Figura 21.

Los grafos de precedencias utilizados tienen unas dimensiones comprendidas entre 5 y 20 tareas. El número del nodo hace referencia a la tarea y el número en la parte superior derecha se refiere a su tiempo de proceso.

Los 120 ejemplares se han generado de la misma forma que en Pastor et al. (2021) y Aguilar et al. (2021).

Los ejemplares generados se pueden clasificar en cuatro grupos (aquí llamados Conjuntos) de acuerdo con los grafos de precedencias que utilizan.

El Conjunto 1, está compuesto por 60 ejemplares creados a partir de los grafos de precedencias de la Figura 18. Para generar los diferentes ejemplares se ha sustraído un número de tareas de cada uno de los grafos asignados a cada una de las líneas. El número de tareas a sustraer en cada línea es un valor que se obtiene de manera aleatoria y está comprendido entre [0, *número total de tareas del grafo* - 1]. Por ejemplo, si para el grafo de precedencias de la línea 1 (Figura 18) se obtiene que el número de tareas a sustraer es 3, significa que el total de tareas del grafo resultante serán 17; entonces, del grafo se sustraen las 3 últimas tareas (es decir, tarea 18, 19 y 20), por lo que el grafo resultante estará formado por las tareas de la 1 a la 17. Los tiempos de proceso de las tareas de los grafos de precedencias de la Figura 18 han sido generados de manera aleatoria entre 1 y 10 unidades de tiempo (ut). Los tiempos de tarea son los mismos para todos los ejemplares del Conjunto 1. El tiempo de ciclo de cada línea para cada ejemplar se han generado de manera aleatoria entre 10 y 200 ut, considerando las siguientes premisas: el tiempo de ciclo de cada línea será superior a la carga de trabajo de su línea; los tiempos de ciclo de las líneas deben ser diferentes entre sí; no se generan lotes superiores a 80 unidades; y la carga de trabajo de la estación multilínea este comprendida entre el 80% y el 100%.

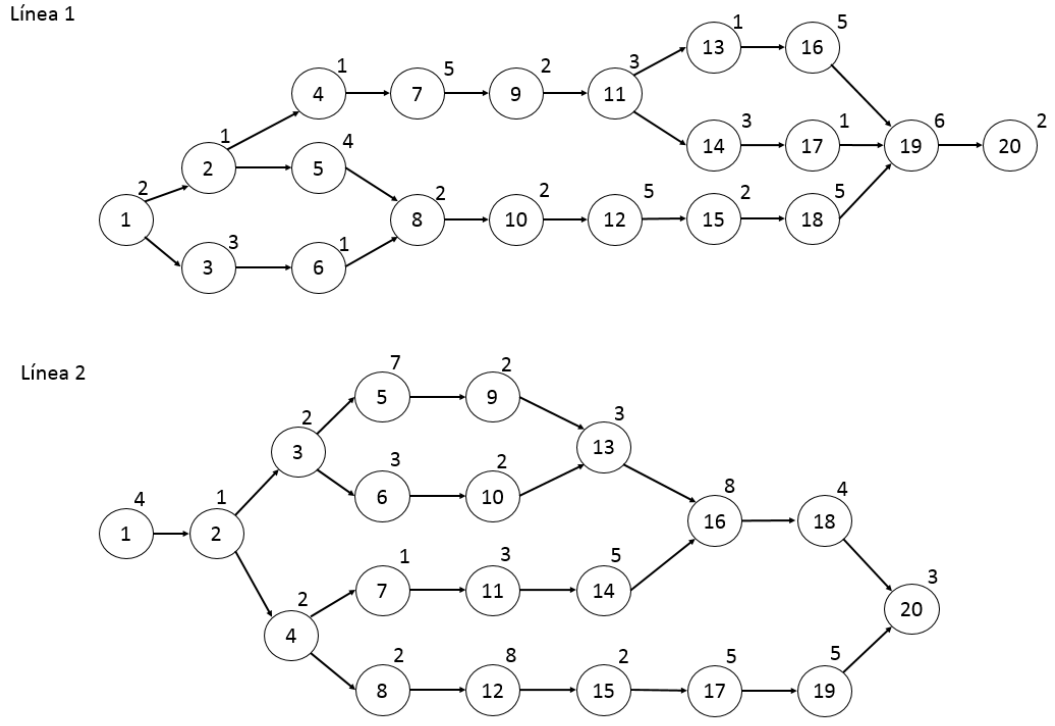

Figura 18. Grafo de precedencias de la línea 1 y 2, Conjunto 1.

El Conjunto 2 está compuesto por 20 ejemplares creados a partir de los grafos de precedencias de la Figura 19. Los ejemplares de este Conjunto se han generado de la

misma forma que en el Conjunto 1. El tiempo de las tareas de los grafos de precedencias de la Figura 19 se han generado de manera aleatoria entre 1 y 10 ut para el grafo de línea 1, y entre 10 y 50 ut para el grafo de la línea 2. La diferencia en el rango de tiempos de procesos entre línea 1 y 2, es para representar productos con tiempos de procesos diferentes. Los tiempos de tarea son los mismos para todos los ejemplares del Conjunto 2. El tiempo de ciclo de cada línea para cada ejemplar se ha generado de manera aleatoria entre 10 y 500 ut, considerando las mismas premisas que en el Conjunto 1.

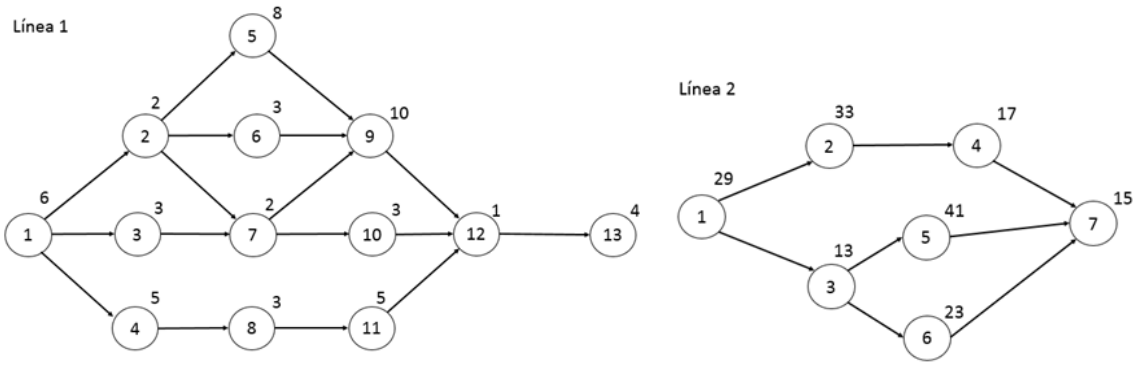

Figura 19. Grafo de precedencias de la línea 1 y 2, Conjunto 2.

El Conjunto 3 está compuesto por 20 ejemplares creados a partir de los grafos de precedencias de la Figura 20. Los ejemplares de este conjunto se han generado de la misma forma que en el Conjunto 1. El tiempo de las tareas de los grafos de precedencias de la Figura 20 se han generado de manera aleatoria entre 1 y 10 ut para el grafo de línea 1, y entre 10 y 100 ut para el grafo de la línea 2. La diferencia en el rango de tiempos de procesos entre línea 1 y 2, es para representar productos con tiempos de procesos bastante diferenciados. Los tiempos de tarea son los mismos para todos los ejemplares del Conjunto 3. El tiempo de ciclo de cada línea para cada ejemplar se ha generado de manera aleatoria entre 10 y 400 ut, considerando las mismas premisas que en el Conjunto 1.

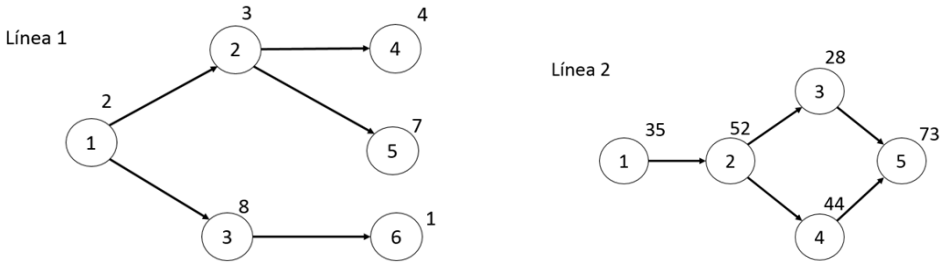

Figura 20. Grafo de precedencias de la línea 1 y 2, Conjunto 3.

El Conjunto 4 está compuesto por 20 ejemplares creados a partir de los grafos de precedencias de la Figura 21. Los ejemplares de este conjunto se han generado de la

misma forma que en el Conjunto 1. El tiempo de las tareas de los grafos de precedencias de la Figura 21 se han generado de manera aleatoria entre 1 y 100 ut para los grafos de ambas líneas. La amplitud del rango de tiempos de procesos en línea 1 y 2, se realiza para representar productos con gran variabilidad en sus tiempos de procesos. Los tiempos de tarea son los mismos para todos los ejemplares del Conjunto 4. El tiempo de ciclo de cada línea se ha generado de manera aleatoria entre 10 y 500 ut, considerando las mismas premisas que en el Conjunto 1.

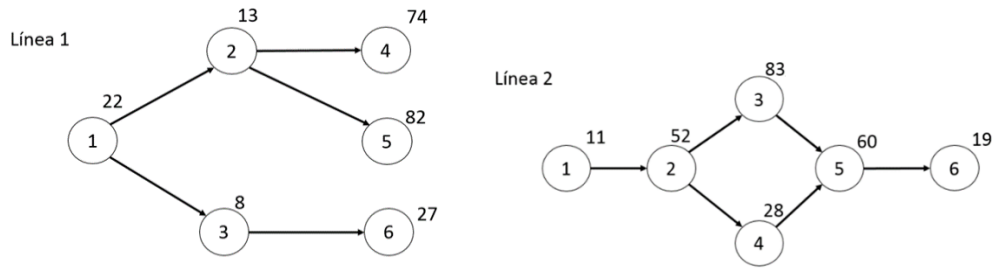

Figura 21. Grafo de precedencias de la línea 1 y 2, Conjunto 4.

4.2.4.2 Experimento computacional 1

En el primer experimento computacional el objetivo es analizar si los modelos desarrollados son capaces de resolver los ejemplares del problema en un tiempo límite fijado de 60 minutos (3600 segundos) para cada ejemplar, mismo tiempo que en Pastor et al. (2021) y Aguilar et al. (2021). Cabe recordar que en los modelos V1 y V2, el valor del parámetro GAP viene configurado por defecto a GAP=10^{-6}, y en V3 se ha configurado a GAP=0.9999.

Los resultados obtenidos por los modelos V1, V2 y V3 para el experimento computacional realizado se recogen en el Anexo B.1.1 (véase en el documento externo "Anexos").

A continuación, se presenta una tabla resumen de los resultados obtenidos por los modelos V1, V2 y V3. En la Tabla 1, se muestra el número de ejemplares para los cuales los diferentes modelos han conseguido una solución óptima, no óptima (pero factible) y no factible. Los resultados en la Tabla 1 se representan como "óptimo / no óptimo / no factible" para cada modelo y conjunto de ejemplares. La etiqueta "no factible" significa que no se ha encontrado ninguna solución factible en el tiempo máximo de cálculo fijado. La etiqueta "no óptimo" significa, como se ha introducido, que se ha encontrado una solución factible, pero que no se ha demostrado que sea óptima. La etiqueta "óptimo" significa que se ha encontrado una solución óptima demostrada dentro del tiempo límite fijado. En la cabecera de la tabla se muestra la etiqueta "Conjunto", que hace referencia al grupo de ejemplares resueltos y entre paréntesis el número de ejemplares que

componen el conjunto. Cabe destacar que, para que una solución de V2 sea considerada óptima, esta solución tiene que ser igual a la obtenida por V1 o V3 (en el caso que estas consigan la solución óptima), o ser factible y con un valor de la función objetivo igual a 0, con lo que se asegura el óptimo.

Tabla 1. Resumen de los resultados obtenidos por los procedimientos V1, V2 y V3.

Procedimiento	Conjunto 1 (60)	Conjunto 2 (20)	Conjunto 3 (20)	Conjunto 4 (20)	Total (120)
V1	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
V2	41 / 6 / 13	15 / 4 / 1	18 / 1 / 1	19 / 1 / 0	93 / 12 / 15
V3	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28

De los resultados de la Tabla 1, se puede observar que V2 es el modelo que encuentra más soluciones factibles, 105 (considerando las “óptimas” y las “no óptimas”). Por otra parte, V1 y V3 han obtenido los mismos resultados, 92 soluciones factibles. Sin embargo, se observa que los modelos V1, V2 y V3 no han podido encontrar una solución factible a 15 (para V2) y 28 (V1 y V3) ejemplares, dentro del tiempo de cálculo máximo fijado, aun siendo algunos de estos ejemplares de tan solo 6 tareas.

En la Tabla 2 se muestra la comparación de los resultados referentes al promedio del tamaño de los buffers y del tiempo de cálculo, obtenidos por los modelos V1, V2 y V3. Para la obtención de estos promedios se ha considerado el valor total del buffer (el de ambas líneas) de, únicamente, los ejemplares para los cuales se ha encontrado una solución factible u óptima por los 3 modelos. En este experimento han sido 92 ejemplares.

Tabla 2. Promedio del tamaño de los buffers y del tiempo de cálculo obtenidos por los procedimientos V1, V2 y V3.

	V1	V2	V3
Promedio tamaño buffers	1.750	1.293	1.750
Tiempo promedio (s)	258.66	210.3	260.17

Como se observa en la Tabla 2 el modelo V2 ha obtenido un mejor (menor) promedio del tamaño de los buffers que los modelos V1 y V3, reduciendo dicho valor en un 26%. Además, V2 también muestra un mejor rendimiento en tiempo de cálculo, donde ha reducido el tiempo promedio de cálculo de V1 y V3 en un 18% aproximadamente.

De los resultados anteriores se concluye que, para un tiempo de cálculo prefijado, el modelo V2, el cual considera la misma secuencia de tareas para todas las unidades, obtiene mejores resultados que los otros modelos presentados.

4.2.4.3 Experimento computacional 2

En este experimento computacional, para cada ejemplar se ha fijado el valor de la variable a_h (que indica el número de unidades tratadas y no tratadas que hay en el buffer, es decir el tamaño del buffer) de cada línea h a diferentes valores. El objetivo de este experimento es observar si los modelos son capaces de encontrar una solución (factible u óptima) o no, de una forma rápida (en tiempo), para un valor del buffer fijo predeterminado. De esta forma, si así fuese, se podría ir fijando a diferentes valores la variable a_h de cada línea, hasta encontrar la solución óptima (si existe una solución factible) para cada ejemplar.

Para este experimento se han resuelto un conjunto de 50 ejemplares basados en ejemplares del experimento 1 y generados como se detallada a continuación. Los 50 ejemplares generados corresponden a: 20 ejemplares basados en dos ejemplares extraídos del Conjunto 1; 10 ejemplares basados en un ejemplar extraído del Conjunto 2; 10 ejemplares basados en un ejemplar extraído del Conjunto 3; y 10 ejemplares basados en un ejemplar extraído del Conjunto 4. Para cada ejemplar extraído (de aquí en adelante conocido como ejemplar de referencia) de cada conjunto, se ha fijado el valor del buffer a_h de cada línea h a diez valores diferentes (de esta forma se generan 10 ejemplares por cada ejemplar de referencia del conjunto x). Estos valores a_h se han elegido a partir de los resultados obtenidos por los modelos V1, V2 y V3, para cada ejemplar del experimento computacional 1.

En el experimento computacional 2, el tiempo máximo de cálculo para cada ejemplar se ha fijado a 3600 segundos. Los resultados obtenidos por los modelos V1, V2 y V3, se recogen en el Anexo B.1.2 (véase en el documento externo “Anexos”).

En la Tabla 3 se muestra un resumen de los resultados obtenidos para cada uno de los modelos. En la columna “Total” se muestra el número de: “Solución”, si se ha encontrado una solución o se ha encontrado que ésta no existe, para el valor del buffer a_h fijado; “Sin solución”, si no se ha encontrado una solución para el valor del buffer a_h fijado dentro del tiempo de cálculo máximo permitido.

Tabla 3. Resumen de los resultados obtenidos por los procedimientos V1, V2 y V3 al fijar a diferentes valores la variable a_h.

Procedimiento	Total	
	Solución	Sin solución
V1	21	29
V2	32	18
V3	21	29

En la Tabla 3 se observa que, tanto los modelos V1 y V3, obtienen los mismos resultados: solución en 21 de 50 ejemplares. Por otra parte, V2 encuentra solución en 32 de los 50 ejemplares. Sin embargo, para los tres modelos, el número de ejemplares para los cuales no se encuentra una solución es elevado, siendo 29 ejemplares para V1 y V3, y 18 ejemplares para V2.

De los resultados obtenidos se puede observar que para algunos ejemplares de los cuales se conoce su valor a_h del buffer (ya sea factible u óptimo), fijando la variable a_h a este valor, el tiempo de cálculo necesario para obtener una solución es elevado, en especial para los modelos V1 y V3. Llegando estos dos modelos, en varios casos, a no encontrar una solución sí conocida por V2 dentro del tiempo límite de cálculo establecido.

Por otra parte, V2 consigue reducir en muchos ejemplares el tiempo de cálculo necesario por V1 y V3. Además, resuelve 14 ejemplares en un breve tiempo de cálculo (entre 0 y 1 segundos). Sin embargo, para otros ejemplares, el tiempo necesario de cálculo incrementa considerablemente en más de 50 segundos.

4.3 Heurísticas

Como se ha podido observar en la Subsección 4.2.4, el modelo de PLEM (V1, V2 y V3) en algunos ejemplares no es capaz de encontrar una solución factible dentro de un tiempo límite fijado y, para otros ejemplares, el tiempo necesario para encontrar una solución factible u óptima puede llegar a ser de hasta 3600 segundos.

Con el objetivo de obtener un mayor número de soluciones factibles, en esta sección se presentan dos procedimientos heurísticos: HB-1A y HB-1B.

Las heurísticas desarrolladas, secuencian y programan las tareas asignadas en una estación multilínea, cumpliendo con los supuestos y condiciones definidos en la Subsección 2.2.2.

4.3.1 Heurística HB-1A

El procedimiento heurístico HB-1A presentado en esta sección se ha diseñado para resolver el (sub)problema de dimensionado de buffer definido en la Subsección 2.2.2.

La solución proporcionada por la heurística HB-1A determina el tamaño del buffer, la secuenciación y la programación de las tareas. Adicionalmente a las premisas asumidas en la definición del (sub)problema de dimensionado de buffer (Subsección 2.2.2), en HB-1A se asumen las siguientes:

- El producto con el tiempo de ciclo más pequeño se asigna a la línea 1 (L1).
- En un ciclo c_h de la línea h, se pueden realizar 0, 1, 2 o más unidades de un producto, siempre que los tiempos de procesos lo permitan y que el número de unidades ya realizadas en la línea h no sea mayor al ciclo c_h.

A continuación se describe la notación utilizada en la heurística HB-1A.

Datos:

H Conjunto de líneas, $H = \{1,2\}$.

T_h Conjunto de las tareas asignadas a la estación multilínea del producto de la línea h; $h \in H$.

ct_h Tiempo de ciclo de la línea h; $h \in H$.

lcm Tiempo de ciclo de todas las líneas: $lcm = m.c.m.\{ct_1, ct_2\}$.

q_h Número de unidades a producir del producto de la línea h en un superciclo: $q_h = lcm/\ ct_h$; $h \in H$.

t_{hi} Tiempo de proceso de la tarea i del producto de la línea h; $h \in H; i \in T_h$.

tt_h Tiempo total de una unidad de la línea h: $tt_h = \sum_{i \in T_h} t_{hi}; h \in H.$

μ Numero $\mathbb{R}^+$ muy pequeño (p.ej. 10^{-6}).

Variables:

Ta Reloj del sistema. Inicialmente el tiempo de reloj será $Ta = 0$. El tiempo Ta, se va incrementando conforme se van realizando las tareas, tanto de la línea 1 como de la línea 2.

c_h Ciclo actual en la línea h: $c_h = \left\lceil \frac{Ta}{ct_h} + \mu \right\rceil$; $h \in H$.

x_2 1, si hay una unidad empezada en la línea $h = 2$ en el ciclo c_2; 0 de lo contrario.

a_h Máximo número de unidades tratadas y no tratadas de producto que hay en el buffer de la línea h en un superciclo; $h \in H$.

z_h Número de unidades tratadas de producto que hay inicialmente en el buffer de la línea h en un superciclo; $h \in H$.

nu_h Número de unidades ya realizadas en el puesto de trabajo de la línea h en el superciclo. El número de unidades nu_h tiene un valor inicial: $nu_h = 0$; $h \in H$.

tr_h Tiempo total de las tareas no realizadas de la unidad $nu_h + 1$ de la línea h. El tiempo tr_h tiene un valor inicial: $tr_h = tt_h$; $h \in H$. Este tiempo se va reduciendo conforme se van realizando las tareas de dicha unidad.

CTC_h Conjunto de tareas candidatas de la unidad $nu_h + 1$ de la línea $h \in H$. Una tarea es candidata cuando ésta aún no ha sido programada y sus predecesoras ya se han secuenciado.

MT $MT[h][x]$ proporciona el número de unidades que se han realizado en el ciclo x en la línea h. La matriz es de tamaño $[H][max\ \{q_1, q_2\}]$ y el valor inicial para todos sus elementos es 0.

A continuación, se muestra los diferentes pasos del procedimiento HB-1A. Estos pasos se deben realizar de forma iterativa mientras haya unidades por realizar en L1 y/o L2 y el tiempo de reloj Ta sea menor que el tiempo lcm. Cuando se han realizado todas las unidades tanto de la línea L1 como de la línea L2, el procedimiento finaliza y se obtiene el valor del buffer a_h de la línea h. En esta heurística se ha considerado la posibilidad de finalizar el procedimiento antes de que se hayan realizado todas las unidades, esto ocurrirá en el caso de que no haya tiempo suficiente para realizar las tareas de todas las unidades pendientes y, así, no se dispondrá de una solución factible.

Inicialización de variables: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0;\ c_1 = 1;\ c_2 = 1;\ x_2 = 0;\ tr_2 = tt_2;$ $MT[h][x] = 0$ ($h \in H;\ x = 1, \dots, [max\ \{q_1, q_2\}]$).

1. Se comprueba si se cumplen las dos siguientes condiciones:
   - Hay unidades por realizar en L1 o L2, es decir: $nu_1 < q_1$ o $nu_2 < q_2$.
   - La suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al tiempo lcm.

   Si se cumplen las dos condiciones, se irá al Paso 2; de lo contrario se irá al Paso 6.

2. Se comprueba si se cumplen las cuatro siguientes condiciones:
   - El número de unidades ya realizadas en L1 (nu_1) sea menor al número de unidades a realizar en L1 (q_1), es decir: $nu_1 < q_1$.
   - El número de unidades ya realizadas en L1 (nu_1) sea menor al ciclo actual de L1 (c_1), es decir: $nu_1 < c_1$.
   - La suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ de L1 quepa en el ciclo actual de L1, esto es: $Ta + tt_1 \leq ct_1 \cdot c_1$.
   - Solo en el caso de que haya una unidad empezada en L2 ($x_2 = 1$) en el ciclo c_2: después de secuenciar todas las tareas de la unidad $nu_1 + 1$ de L1, debe haber tiempo suficiente en el ciclo c_2 de L2 para terminar las tareas que faltan de la unidad $nu_2 + 1$ empezada en L2, esto es: $Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2$.

   De no cumplirse las cuatro condiciones, se irá al Paso 3; en caso contrario:

   2.1. Se secuencian y programan, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas

de la unidad $nu_1 + 1$ de L1. Se actualizan valores de L1: $Ta = Ta + tt_1$; $nu_1 = nu_1 + 1$; $MT[1][c_1] = MT[1][c_1] + 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.

2.2. Volver al Paso 2.

3. Se comprueba si se cumplen las tres siguientes condiciones:
   - El número de unidades ya realizadas en L2 (nu_2) sea menor al número de unidades a realizar en L2 (q_2), es decir: $nu_2 < q_2$.
   - El número de unidades ya realizadas en L2 (nu_2) sea menor al ciclo actual de L2 (c_2), es decir: $nu_2 < c_2$.
   - La suma del tiempo de las tareas que quedan por realizar de la unidad $nu_2 + 1$ de L2 quepa en el ciclo actual de L2 ($Ta + tr_2 \leq ct_2 \cdot c_2$).

   De no cumplirse las tres condiciones, se irá al Paso 5; en caso contrario:

   3.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $x_2 = 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

   3.2. Si se han realizado todas las tareas de la unidad en L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se irá al Paso 5.

   3.3. Si en L1 ya se han realizado todas las unidades ($nu_1 = q_1$), entonces se volverá al inicio del Paso 3.

   3.4. Si la duración de la tarea i secuenciada en L2 es mayor que el tiempo de un ciclo de L1 ($t_{2i} > ct_1$), entonces se irá al Paso 2; en caso contrario se irá al Paso 4.

4. Se comprueba si se cumplen las dos siguientes condiciones:
   - Hay una unidad empezada en L2 ($x_2 = 1$).
   - Hay unidades por realizar en L1 ($nu_1 < q_1$).

   Y también se comprueba si se cumple una de las dos condiciones siguientes:
   - En caso de no haber unidades por recuperar en L1 ($nu_1 = c_1$): se comprueba si hay tiempo en el siguiente ciclo de L1 ($c_1 + 1$) para realizar todas las tareas de la siguiente unidad de L1 ($nu_1 + 1$) y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.
   - En caso de haber unidades por recuperar en L1 ($nu_1 < c_1$): se comprueba si en el ciclo actual de L1 no hay tiempo para realizar todas las tareas de la unidad de la L1 (es decir: $ct_1 \cdot c_1 < Ta + tt_1$), y que hay tiempo en el siguiente ciclo de L1 ($c_1 + 1$) para realizar las tareas de la unidad pendiente en L1 y

realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.

De no cumplirse las tres condiciones, se irá al Paso 5; en caso contrario:

4.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias, y que hace que se cumpla una de las dos últimas condiciones anteriores. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

4.2. Si se han realizado todas las tareas de la unidad de L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables en el siguiente orden: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se va al Paso 5; en caso contrario se va al Paso 4.

5. Actualización del reloj Ta: cuando no hay unidades por recuperar en ninguna de las líneas, o cuando no hay tiempo para realizar todas las tareas de una unidad de la línea h en el ciclo c_h. Se actualizará cuando se cumpla uno de los siguientes Conjuntos de condiciones:

   Conjunto 1:

   - Hay unidades por realizar en una o ambas líneas, es decir: $nu_1 < q_1$ y/o $nu_2 < q_2$.
   - No hay unidades por recuperar en ninguna de las líneas, $nu_1 = c_1$ y $nu_2 = c_2$.

   Conjunto 2:

   - Hay unidades por recuperar en ambas líneas, $nu_1 < c_1$ y $nu_2 < c_2$.
   - La suma del tiempo de las tareas de la unidad de la L1 (tt_1) y de la unidad de la L2 (tr_2), no cabe en el tiempo del ciclo c_h de sus respectivas líneas, es decir: $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2$.

   Conjunto 3:

   - Hay unidades por recuperar solo en L1 ($nu_1 < c_1$ y $nu_2 = c_2$).
   - La suma del tiempo de todas las tareas de la unidad en L1 no cabe en el tiempo del ciclo c_1, es decir: $ct_1 \cdot c_1 < Ta + tt_1$.

   Conjunto 4:

   - Hay unidades por recuperar solo en L2 ($nu_1 = c_1$ y $nu_2 < c_2$).
   - La suma del tiempo de las tareas que faltan por procesar de la unidad $nu_2 + 1$ en L2 no cabe en el tiempo del ciclo c_2, es decir: $ct_2 \cdot c_2 < Ta + tr_2$.

Si no se cumple ninguno de los conjuntos de condiciones anteriores, ir al Paso 1.

La operativa para la actualización del reloj Ta consiste en:

5.1. Si se cumple uno de los Conjuntos y hay unidades por realizar solo en una de las líneas, es decir: $nu_1 < q_1$ o $nu_2 < q_2$; entonces, se actualizará el tiempo de reloj Ta al tiempo de ciclo de la línea h donde todavía quedan unidades por realizar: $Ta = ct_h \cdot c_h$; y se actualiza el ciclo c_h de la línea h.

5.2. Si se cumple uno de los Conjuntos y hay unidades por realizar en ambas líneas, es decir: $nu_1 < q_1$ y $nu_2 < q_2$; entonces, se actualizará el tiempo de reloj Ta al tiempo de ciclo de la línea h menor entre L1 y L2: $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\}$; y se actualizarán los ciclos c_h de las líneas.

5.3. Ir al Paso 1.

6. En este punto se termina el procedimiento heurístico, debido a uno de los siguientes casos:
   - Se han realizado todas las unidades en L1 y L2: $nu_1 = q_1$ y $nu_2 = q_2$. Se realiza el cálculo de los valores z_h y a_h (ver Figura 22 y Figura 23) y el procedimiento finaliza con una solución factible.
   - No hay tiempo suficiente para realizar las tareas que aún han de ejecutarse dentro del tiempo lcm. El procedimiento finaliza sin solución factible.

El cálculo del valor del buffer z_h y a_h se realizará cómo se indica en los pseudocódigos propuestos a continuación en la Figura 22 y Figura 23:

0. $z_h = 0$;
1. ***Para todo*** $(h = 1, \dots, 2)$
2. ***Para todo*** $(c = 1, \dots, q_h)$
3. ***Si*** $(\sum_{\tau=1}^{c}(MT[h][\tau] - 1) < z_h)$ *hacer*:
4. $z_h = \sum_{\tau=1}^{c}(MT[h][\tau] - 1)$;
5. ***Fin Para***
6. Devuelve el valor de $z_h = |z_h|$ para la línea h;
7. ***Fin Para***

Figura 22. Pseudocódigo del cálculo del valor z_h.

0. $a_h = 0$;
1. ***Para todo*** $(h = 1, \dots, 2)$
2. ***Para todo*** $(c = 1, \dots, q_h)$
3. ***Si*** $(z_h + \sum_{\tau=1}^{c}(MT[h][\tau] - 1) > a_h)$ *hacer*:
4. $a_h = z_h + \sum_{\tau=1}^{c}(MT[h][\tau] - 1)$;
5. ***Fin Para***
6. Devuelve el valor de a_h para la línea h;
7. ***Fin Para***

Figura 23. Pseudocódigo del cálculo del valor a_h.

A continuación en la Figura 24 se muestra el pseudocódigo de la heurística HB-1A:

Inicialización: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0\ ;\ c_1 = 1;\ c_2 = 1;\ x_2 = 0;\ tr_2 = tt_2;\ MT[h][x] = 0\ \left(h \in H;\ x = 1, \dots, \left[max\{q_1, q_2\}\right]\right)$.

1. ***Mientras*** $((nu_1 < q_1$ o $nu_2 < q_2)$ y $((x_2 = 0$ y $Ta + (q_1 - nu_1) \cdot tt_1 + (q_2 - nu_2) \cdot tt_2 \leq lcm)$ o $(x_2 = 1$ y $Ta + (q_1 - nu_1) \cdot tt_1 + (q_2 - nu_2) \cdot tt_2 - (tt_2 - tr_2) \leq lcm)))$ *hacer*
   2. ***Mientras*** $(nu_1 < q_1$ y $nu_1 < c_1$ y $Ta + tt_1 \leq ct_1 \cdot c_1$ y $((x_2 = 0)$ o $(x_2 = 1$ y $(Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2))$ *hacer*
      2.1. Secuenciar y programar, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas de la unidad $nu_1 + 1$ de L1 y actualizar: $Ta = Ta + tt_1;\ nu_1 = nu_1 + 1;\ MT[1][c_1] = MT[1][c_1] + 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;$ $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      Fin Mientras.
   3. ***Si*** $(nu_2 < q_2$ y $nu_2 < c_2$ y $Ta + tr_2 \leq ct_2 \cdot c_2)$ *entonces*
      3.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande, y actualizar: $Ta = Ta + t_{2i};\ tr_2 = tr_2 - t_{2i};\ x_2 = 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      3.2. ***Si*** $(tr_2 = 0)$ *entonces*
      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5;
      3.3. ***Si*** $(nu_1 = q_1)$ *entonces* ir al Paso 3;
      3.4. ***Si*** $(t_{2i} > ct_1)$ *entonces* ir al Paso 2;
      En caso contrario ir al Paso 4.
      En caso contrario ir al Paso 5.
   4. ***Mientras*** $(x_2 = 1$ y $nu_1 < q_1$ y $((nu_1 = c_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i})$ o $(nu_1 < c_1$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}))$ *hacer*
      4.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande y cumpla que $(nu_1 = c_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i})$ o que $(nu_1 < c_1$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i})$, entonces: $Ta = Ta + t_{2i};\ tr_2 = tr_2 - t_{2i};\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      4.2. ***Si*** $(tr_2 = 0)$ *entonces*
      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5.
      Fin Mientras.
   5. ***Si*** $(((nu_1 < q_1$ y/o $nu_2 < q_2)$ y $nu_1 = c_1$ y $nu_2 = c_2)$ o $(nu_1 < c_1$ y $nu_2 < c_2$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2)$ o $(nu_1 < c_1$ y $nu_2 = c_2$ y $ct_1 \cdot c_1 < Ta + tt_1)$ o $(nu_1 = c_1$ y $nu_2 < c_2$ y $ct_2 \cdot c_2 < Ta + tr_2))$
      5.1. ***Si*** $(nu_1 = q_1$ y $nu_2 < q_2)$ *entonces*
      $Ta = ct_2 \cdot c_2;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      Si $(nu_1 < q_1$ y $nu_2 = q_2)$ *entonces*
      $Ta = ct_1 \cdot c_1; c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      5.2. ***Si*** $(nu_1 < q_1$ y $nu_2 < q_2)$ *entonces*
      $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\};\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      5.3. Ir al Paso 1.
      En caso contrario ir al Paso 1.
   Fin Mientras.
6. ***Si*** $(nu_1 = q_1$ y $nu_2 = q_2)$ *entonces*
   Calcular los valores del buffer z_h y a_h (ver Figura 22 y Figura 23).
   En caso contrario el procedimiento finaliza sin solución factible.

Figura 24. Pseudocódigo de la heurística HB-1A.

Ejemplo ilustrativo 1 del procedimiento HB-1A:

A continuación se presenta la resolución de un ejemplo para ilustrar el funcionamiento del procedimiento HB-1A.

Datos:

$ct_1 = 10$; $ct_2 = 25$; $lcm = 50$;

$q_1 = 50/10 = 5$; $q_2 = 50/25 = 2$;

$T_1 = \{1,2\}$; $T_2 = \{1,2\}$;

$t_{11} = 2$; $t_{12} = 3$; $t_{21} = 10$; $t_{22} = 1$;

$tt_1 = (2 + 3) = 5$; $tt_2 = (10 + 1) = 11$;

$\mu = 10^{-6}$.

La relación de precedencia tanto para el conjunto de tareas de la línea 1 y de la línea 2 es la siguiente: la tarea 1 precede a la tarea 2.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_2 = 0$; $tr_2 = 11$; $MT[h][x] = 0$ ($h \in H$; $x = 1, \dots, [max\{5, 2\}]$).

1. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 0 < 5$; $nu_2 < q_2 \rightarrow 0 < 2$) y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($0 + (5 - 0) \cdot 5 + (2 - 0) \cdot 11 = 47 \leq 50$), se va al Paso 2.
2. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 0 < 5$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 0 < 1$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 0 + 5 \leq 10 \cdot 1$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 0 y 2, la tarea 2 entre los instantes 2 y 5; y se actualizan los valores: $Ta = 0 + 5 = 5$; $nu_1 = 0 + 1 = 1$; $MT[1][c_1] = MT[1][1] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{5}{10} + 10^{-6} \right\rceil = 1$; $c_2 = \left\lceil \frac{5}{25} + 10^{-6} \right\rceil = 1$; y se vuelve al Paso 2.
3. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 5$) pero el número de unidades ya realizadas en L1 no es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 1 \not< 1$), entonces se va al Paso 3.

4. Paso 3. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 2$); el número de unidades ya realizadas en L2 es menor que el ciclo actual de L2 ($nu_2 < c_2 \rightarrow 0 < 1$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (0+1=1) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 5 + 11 \leq 25 \cdot 1$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande que respete las precedencias, es decir la tarea 1: entre los instantes 5 y 15; y se actualizan los valores: $Ta = 5 + 10 = 15$; $tr_2 = 11 - 10 = 1;\ x_2 = 1;\ c_1 = \left\lceil \frac{15}{10} + 10^{-6} \right\rceil = 2$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
5. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 5$). Como $nu_1 < c_1 \rightarrow 1 < 2$, pero la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 10 \cdot 2 \not< 15 + 5$ no se cumple (hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), entonces, se va el Paso 5.
6. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 5$ y $nu_2 < q_2 \rightarrow 0 < 2$), pero hay unidades por recuperar en ambas líneas ($nu_1 = c_1 \rightarrow 1 \neq 2$ y $nu_2 = c_2 \rightarrow 0 \neq 1$); conjunto 2: hay unidades por recuperar en ambas líneas ($nu_1 < c_1 \rightarrow 1 < 2$ y $nu_2 < c_2 \rightarrow 0 < 1$), pero el tiempo de las tareas pendientes ($tt_1 = 5$ y $tr_2 = 1$) de L1 y L2 sí que caben en sus ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 10 \cdot 2 \not< 15 + 5$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 25 \cdot 1 \not< 15 + 1$); conjunto 3: hay unidades por recuperar en L1 ($nu_1 < c_1 \rightarrow 1 < 2$) pero también en L2 ($nu_2 = c_2 \rightarrow 0 \neq 1$); conjunto 4: hay unidades por recuperar en L2 ($nu_2 < c_2 \rightarrow 0 < 1$) pero también en L1 ($nu_1 = c_1 \rightarrow 1 \neq 2$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
7. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 5;\ nu_2 < q_2 \rightarrow 0 < 2$) y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($15 + (5 - 1) \cdot 5 + (2 - 0) \cdot 11 - (11 - 1) = 47 \leq 50$), se va al Paso 2.
8. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 5$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 1 < 2$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 15 + 5 \leq 10 \cdot 2$); y hay una unidad empezada en L2 ($x_2 = 1$) y el tiempo de las tareas que quedan por procesar de la unidad de L2 es menor al tiempo del

ciclo c_2 ($Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2 \rightarrow 15 + 5 + 1 \leq 25 \cdot 1$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 15 y 17, la tarea 2 entre los instantes 17 y 20; y se actualizan los valores: $Ta = 15 + 5 = 20$; $nu_1 = 1 + 1 = 2$; $MT[1][c_1] = MT[1][2] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{20}{10} + 10^{-6} \right\rceil = 3$; $c_2 = \left\lceil \frac{20}{25} + 10^{-6} \right\rceil = 1$; y se vuelve al Paso 2.

9. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 5$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 2 < 3$), la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 20 + 5 \leq 10 \cdot 3$); hay una unidad empezada en L2 ($x_2 = 1$), pero el tiempo de las tareas que quedan por procesar de la unidad de L2 es mayor al tiempo del ciclo c_2 ($Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2 \rightarrow 20 + 5 + 1 \nleq 25 \cdot 1$); entonces se va al Paso 3.
10. Paso 3. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 2$); el número de unidades ya realizadas en L2 es menor que el ciclo actual de L2 ($nu_2 < c_2 \rightarrow 0 < 1$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (0+1=1) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 20 + 1 \leq 25 \cdot 1$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 20 y 21; y se actualizan los valores: $Ta = 20 + 1 = 21$; $tr_2 = 1 - 1 = 0$; $x_2 = 1$; $c_1 = \left\lceil \frac{21}{10} + 10^{-6} \right\rceil = 3$. En el Paso 3.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (0+1=1) de L2, entonces, se actualizan los valores: $nu_2 = 0 + 1 = 1$; $tr_2 = 11$; $x_2 = 0$; $MT[2][c_2] = MT[2][1] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{21}{25} + 10^{-6} \right\rceil = 1$; y se va al Paso 5.
11. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 5$ y $nu_2 < q_2 \rightarrow 1 < 2$), pero hay unidades por recuperar en la línea L1 ($nu_1 = c_1 \rightarrow 2 \neq 3$ y $nu_2 = c_2 \rightarrow 1 = 1$); conjunto 2: no hay unidades por recuperar en ambas líneas ($nu_1 < c_1 \rightarrow 2 < 3$ y $nu_2 < c_2 \rightarrow 1 \not< 1$); conjunto 3: hay unidades por recuperar solo en L1 ($nu_1 < c_1 \rightarrow 2 < 3$; $nu_2 = c_2 \rightarrow 1 = 1$), pero la suma del tiempo de toda las tareas de la unidad en L1 caben en el tiempo del ciclo c_1 ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 10 \cdot 3 \not< 21 + 5$); conjunto 4: no hay unidades por recuperar en L2 ($nu_2 < c_2 \rightarrow 1 \not< 1$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.

12. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 5;\ nu_2 < q_2 \rightarrow 1 < 2$) y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($21 + (5-2)\cdot 5 + (2-1)\cdot 11 = 47 \leq 50$), se va al Paso 2.
13. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 5$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 2 < 3$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 21 + 5 \leq 10 \cdot 3$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En el Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 21 y 23, la tarea 2 entre los instantes 23 y 26; y se actualizan los valores: $Ta = 21 + 5 = 26$; $nu_1 = 2 + 1 = 3$; $MT[1][c_1] = MT[1][3] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{26}{10} + 10^{-6} \right\rceil = 3$; $c_2 = \left\lceil \frac{26}{25} + 10^{-6} \right\rceil = 2$; y se vuelve al Paso 2.
14. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 3 < 5$) pero el número de unidades ya realizadas en L1 no es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 3 \not< 3$); entonces se va al Paso 3.
15. Paso 3. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 1 < 2$); el número de unidades ya realizadas en L2 es menor que el ciclo actual de L2 ($nu_2 < c_2 \rightarrow 1 < 2$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (1+1=2) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 26 + 11 \leq 25 \cdot 2$), entonces, se va al Paso 3.1. En el Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 1: entre los instantes 26 y 36; y se actualizan los valores: $Ta = 26 + 10 = 36$; $tr_2 = 11 - 10 = 1;\ x_2 = 1;\ c_1 = \left\lceil \frac{36}{10} + 10^{-6} \right\rceil = 4$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
16. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 3 < 5$). Como $nu_1 < c_1 \rightarrow 3 < 4$, la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 10 \cdot 4 < 36 + 5$ sí se cumple (no hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), y hay tiempo en el siguiente ciclo ($c_1 + 1 = 4 + 1 = 5$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $CTC_2 = \{2\}$ de la unidad $nu_2 + 1$ (1+1=2) de L2 ($ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i} \rightarrow 10 \cdot (4 + 1) \geq 36 + 5 + 1$), entonces, se va al Paso 4.1.

En el Paso 4.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 36 y 37; y se actualizan los valores: $Ta = 36 + 1 = 37$; $tr_2 = 1 - 1 = 0$; $c_1 = \left\lceil \frac{37}{10} + 10^{-6} \right\rceil = 4$. En el Paso 4.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (1+1=2) de L2, entonces, se actualizan los valores: $nu_2 = 1 + 1 = 2$; $tr_2 = 11$; $x_2 = 0$; $MT[2][c_2] = MT[2][2] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{37}{25} + 10^{-6} \right\rceil = 2$; y se va al Paso 5.

17. Paso 5. Se actualiza el tiempo de reloj Ta, ya que se cumple uno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en una de las líneas ($nu_1 < q_1 \rightarrow 3 < 5$ y $nu_2 = q_2 \rightarrow 2 = 2$), pero hay unidades por recuperar en la línea L1 ($nu_1 = c_1 \rightarrow 3 \neq 4$ y $nu_2 = c_2 \rightarrow 2 = 2$); conjunto 2: no hay unidades por recuperar en ambas líneas ($nu_1 < c_1 \rightarrow 3 < 4$ y $nu_2 < c_2 \rightarrow 2 \not< 2$); conjunto 3: hay unidades por recuperar solo en L1 ($nu_1 < c_1 \rightarrow 3 < 4$; $nu_2 = c_2 \rightarrow 2 = 2$), y la suma del tiempo de toda las tareas de la unidad en L1 no caben en el tiempo del ciclo c_1 ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 10 \cdot 4 < 37 + 5$); como sí se cumple el conjunto 3, entonces se va al Paso 5.1. Como hay unidades por realizar solo en una de las líneas ($nu_1 < q_1 \rightarrow 3 < 5$ y $nu_2 = q_2 \rightarrow 2 = 2$), en L1, entonces, se actualiza el tiempo $Ta = ct_1 \cdot c_1 = 10 \cdot 4 = 40$; y el ciclo $c_1 = \left\lceil \frac{40}{10} + 10^{-6} \right\rceil = 5$. En el Paso 5.3, se vuelve al Paso 1.
18. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en una de las dos líneas ($nu_1 < q_1 \rightarrow 3 < 5$; $nu_2 = q_2 \rightarrow 2 = 2$) y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($40 + (5 - 3) \cdot 5 + (2 - 2) \cdot 11 = 50 \leq 50$), se va al Paso 2.
19. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 3 < 5$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 3 < 5$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (3+1=4) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 40 + 5 \leq 10 \cdot 5$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En el Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (3+1=4) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 40 y 42, la tarea 2 entre los instantes 42 y 45; y se actualizan los valores: $Ta = 40 + 5 = 45$; $nu_1 = 3 + 1 = 4$; $MT[1][c_1] = MT[1][5] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{45}{10} + 10^{-6} \right\rceil = 5$; $c_2 = \left\lceil \frac{45}{25} + 10^{-6} \right\rceil = 2$; y se vuelve al Paso 2.
20. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 4 < 5$); el número de unidades ya realizadas en L1

es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 4 < 5$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (4+1=5) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 45 + 5 \leq 10 \cdot 5$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En el Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (4+1=5) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 45 y 47, la tarea 2 entre los instantes 47 y 50; y se actualizan los valores: $Ta = 45 + 5 = 50$; $nu_1 = 4 + 1 = 5$; $MT[1][c_1] = MT[1][5] = 1 + 1 = 2$; $c_1 = \left\lceil \frac{50}{10} + 10^{-6} \right\rceil = 6$; $c_2 = \left\lceil \frac{50}{25} + 10^{-6} \right\rceil = 3$; y se vuelve al Paso 2.

21. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como no hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 5 \not< 5$); entonces, se va al Paso 3.
22. Paso 3. Se comprueba si se cumplen las tres condiciones. Como no hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 2 \not< 2$); entonces, se va al Paso 5.
23. Paso 5. No se actualiza el tiempo de reloj Ta, ya que se han realizado todas las unidades tanto en L1 como en L2. Se va al Paso 1.
24. Paso 1. Como ya no hay unidades por realizado en ninguna de las líneas ($nu_1 < q_1 \rightarrow 5 \not< 5$ y $nu_2 < q_2 \rightarrow 2 \not< 2$), entonces se va al Paso 6.
25. Paso 6. Como se han realizado todas las unidades en ambas líneas, se calculan los valores a_h y z_h y el procedimiento finaliza con una solución factible.

Al resolver el ejemplo con el procedimiento propuesto, se obtiene el siguiente resultado $a_h = \{1,0\}$ y $z_h = \{1,0\}$, es decir se necesita un buffer en la línea L1 con una capacidad de 1 unidad.

En la Figura 25, se representa la solución obtenida, donde se puede observar cuándo el operario está realizando las tareas de una unidad (recuadros grises, considerando que el número de dentro indica la tarea) de la línea L1 y de la L2, así como el instante en que empieza y finaliza cada tarea; y cuándo una unidad no ha sido procesada (recuadros naranjas) o ya ha sido procesada (recuadros verdes). Las unidades procesadas y no procesadas pueden permanecer en su línea (L1 o L2) o en el buffer de sus respectivas líneas (Buffer L1 o Buffer L2). Las unidades se representan con el símbolo u (o con el símbolo $u0$, para aquellas que ya hay en los buffers al inicio del superciclo, es decir, instante 0, proveniente del superciclo anterior) acompañadas de un número que hace referencia a la unidad del lote (valor entre 1 y q_h). Las tareas se representan con el símbolo T y acompañadas de un número (índice i) que hace referencia a la tarea en cuestión. Se recuerda que las unidades siguen un orden FIFO para ser procesadas y para salir de las estaciones.

Como se ha especificado, en cada puesto de trabajo y en cada tiempo de ciclo ct_h, entra una unidad sin procesar y sale una procesada de cada línea h; en L1 en los instantes 0, 10, 20, 30, 40 y 50 (instante 0 del siguiente superciclo), y en L2 en los instantes 0, 25 y 50. En la Figura 25, se puede observar que habrá una unidad finalizada en el buffer de L1 en los ciclos 1, 2, 3, 4 y en la mitad del ciclo 5; y habrá 1 unidad ($u05$) terminada ($z_1 = 1$) al comienzo del superciclo, en el instante 0. La unidad $u05$ es la unidad $u5$, del superciclo anterior, ya que como se puede observar, las unidades $u4$ y $u5$ están finalizadas en L1 en el instante 50 (que a su vez es el instante 0 del siguiente superciclo); en ese instante la unidad $u4$ pasa a la siguiente estación y $u5$ pasa al buffer de L1.

En la solución obtenida se pueden observar un momento de inactividad del operario en L1, entre los instantes 37 y 40.

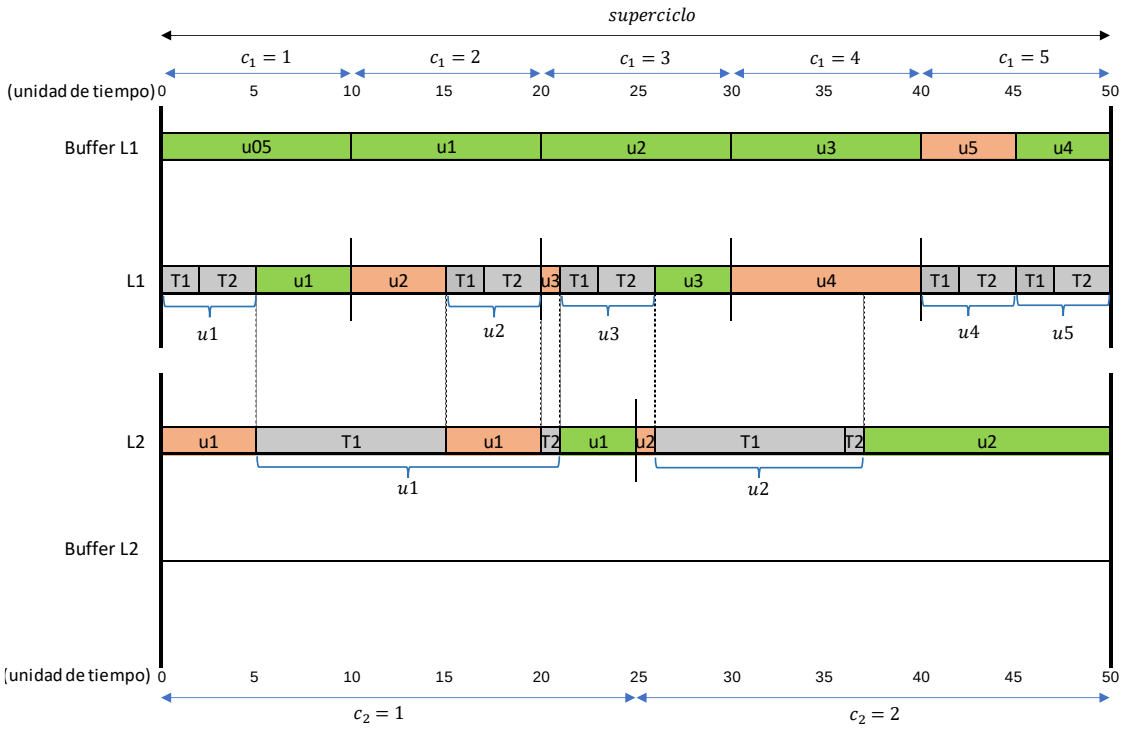

Figura 25. Representación de la solución obtenida para el ejemplo 1 al utilizar la heurística HB-1A.

Al resolver el ejemplo con el modelo matemático V1, se obtiene un resultado óptimo $a_h = \{1,0\}$ y $z_h = \{1,0\}$. Se puede observar que es el mismo resultado que el obtenido con en el procedimiento HB-1A, por lo que dicha solución es óptima.

Ejemplo ilustrativo 2 del procedimiento HB-1A:

A continuación, se muestra un ejemplo donde el procedimiento propuesto HB-1A no encuentra una solución factible; este ejemplo sirve para mostrar como el procedimiento HB-1B (presentado en la Subsección 4.3.2) puede facilitar la obtención de soluciones factibles.

Datos:

$ct_1 = 20$; $ct_2 = 35$; $lcm = 140$;

$q_1 = 140/20 = 7$; $q_2 = 140/35 = 4$;

$T_1 = \{1,2\}$; $T_2 = \{1,2\}$;

$t_{11} = 6$; $t_{12} = 4$; $t_{21} = 15$; $t_{22} = 1$;

$tt_1 = (6 + 4) = 10$; $tt_2 = (15 + 1) = 16$;

$\mu = 10^{-6}$.

La relación de precedencia tanto para el conjunto de tareas de la línea 1 y de la línea 2 es la siguiente: la tarea 1 precede a la tarea 2.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_2 = 0$; $tr_2 = 16$; $MT[h][x] = 0$ ($h \in H$; $x = 1, \dots, [max\{7,4\}]$).

Procedimiento HB-1A:

1. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 0 < 7$; $nu_2 < q_2 \rightarrow 0 < 4$) y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($0 + (7 - 0) \cdot 10 + (4 - 0) \cdot 16 = 134 \leq 140$), se va al Paso 2.
2. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 0 < 7$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 0 < 1$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 0 + 10 \leq 20 \cdot 1$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 0 y 6, la tarea 2 entre los instantes 6 y 10; y se actualizan los valores: $Ta = 0 + 10 = 10$; $nu_1 = 0 + 1 = 1$; $MT[1][c_1] = MT[1][1] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{10}{20} + 10^{-6} \right\rceil = 1$; $c_2 = \left\lceil \frac{10}{35} + 10^{-6} \right\rceil = 1$; y se vuelve al Paso 2.
3. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$) pero el número de unidades ya realizadas en L1 no es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 1 \not< 1$), entonces se va al Paso 3.
4. Paso 3. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 4$); el número de unidades ya realizadas en L2 es menor que el ciclo actual de L2 ($nu_2 < c_2 \rightarrow 0 < 1$); y la suma del tiempo de las

tareas de la unidad $nu_2 + 1$ (0+1=1) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 10 + 16 \leq 35 \cdot 1$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 1: entre los instantes 10 y 25; y se actualizan los valores: $Ta = 10 + 15 = 25$; $tr_2 = 16 - 15 = 1$; $x_2 = 1$; $c_1 = \left\lceil \frac{25}{20} + 10^{-6} \right\rceil = 2$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.

5. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$). Como $nu_1 < c_1 \rightarrow 1 < 2$, pero la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 2 \not< 25 + 10$ no se cumple (hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), entonces, se va al Paso 5.
6. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$ y $nu_2 < q_2 \rightarrow 0 < 4$), pero hay unidades por recuperar en la L1 y L2 ($nu_1 = c_1 \rightarrow 1 \neq 2$ y $nu_2 = c_2 \rightarrow 0 \neq 1$); conjunto 2: hay unidades por recuperar en ambas líneas ($nu_1 < c_1 \rightarrow 1 < 2$ y $nu_2 < c_2 \rightarrow 0 < 1$), pero el tiempo de las tareas pendientes ($tt_1 = 10$ y $tr_2 = 1$) de L1 y L2 sí que caben en sus ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 2 \not< 25 + 10$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 35 \cdot 1 \not< 25 + 1$); conjunto 3: hay unidades por recuperar en L1 ($nu_1 < c_1 \rightarrow 1 < 2$), pero también en L2 ($nu_2 = c_2 \rightarrow 0 \neq 1$); conjunto 4: hay unidades por recuperar en L2 ($nu_2 < c_2 \rightarrow 0 < 1$) pero hay unidades por recuperar en L1 ($nu_1 = c_1 \rightarrow 1 \neq 2$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
7. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$; $nu_2 < q_2 \rightarrow 0 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($25 + (7 - 1) \cdot 10 + (4 - 0) \cdot 16 - (16 - 1) = 134 \leq 140$), se va al Paso 2.
8. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 1 < 2$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 25 + 10 \leq 20 \cdot 2$); y hay una unidad empezada en L2 ($x_2 = 1$), pero el tiempo de las tareas que quedan por procesar de la unidad de L2 es mayor al tiempo del ciclo c_2 ($Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2 \rightarrow 25 + 10 + 1 \not\leq 35 \cdot 1$); entonces se va al Paso 3.

9. Paso 3. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 4$); el número de unidades ya realizadas en L2 es menor que el ciclo actual de L2 ($nu_2 < c_2 \rightarrow 0 < 1$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (0+1=1) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 25 + 1 \leq 35 \cdot 1$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 25 y 26; y se actualizan los valores: $Ta = 25 + 1 = 26$; $tr_2 = 1 - 1 = 0$; $x_2 = 1$; $c_1 = \left\lceil \frac{26}{20} + 10^{-6} \right\rceil = 2$. En el Paso 3.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (0+1=1) de L2, entonces, se actualizan los valores: $nu_2 = 0 + 1 = 1$; $tr_2 = 16$; $x_2 = 0$; $MT[2][c_2] = MT[2][1] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{26}{35} + 10^{-6} \right\rceil = 1$; y se va al Paso 5.
10. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$ y $nu_2 < q_2 \rightarrow 1 < 4$), pero hay unidades por recuperar en la línea L1 ($nu_1 = c_1 \rightarrow 1 \neq 2$ y $nu_2 = c_2 \rightarrow 1 = 1$); conjunto 2: no hay unidades por recuperar en ambas líneas ($nu_1 < c_1 \rightarrow 1 < 2$ y $nu_2 < c_2 \rightarrow 1 \not< 1$); conjunto 3: hay unidades por recuperar en L1 ($nu_1 < c_1 \rightarrow 1 < 2$), pero la suma del tiempo de toda las tareas de la unidad en L1 caben en el tiempo del ciclo c_1 ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 2 \not< 26 + 10$); conjunto 4: no hay unidades por recuperar en L2 ($nu_2 < c_2 \rightarrow 1 \not< 1$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
11. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$; $nu_2 < q_2 \rightarrow 1 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($26 + (7 - 1) \cdot 10 + (4 - 1) \cdot 16 = 134 \leq 140$), se va al Paso 2.
12. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$); el número de unidades ya realizadas en L1 es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 1 < 2$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 26 + 10 \leq 20 \cdot 2$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 26 y 32, la tarea 2 entre los instantes 32 y 36; y se actualizan los valores: $Ta = 26 + 10 = 36$; $nu_1 = 1 + 1 = 2$; $MT[1][c_1] =$

$MT[1][2] = 0 + 1 = 1;\ c_1 = \left\lceil \frac{36}{20} + 10^{-6} \right\rceil = 2;\ c_2 = \left\lceil \frac{36}{35} + 10^{-6} \right\rceil = 2;$ y se vuelve al Paso 2.

13. Paso 2. Se comprueba si se cumplen las cuatro condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$) pero el número de unidades ya realizadas en L1 no es menor que el ciclo actual de L1 ($nu_1 < c_1 \rightarrow 2 \not< 2$), entonces se va al Paso 3.
14. Paso 3. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 1 < 4$); el número de unidades ya realizadas en L2 es menor que el ciclo actual de L2 ($nu_2 < c_2 \rightarrow 1 < 2$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (1+1=2) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 36 + 16 \leq 35 \cdot 2$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande que respete las precedencias, es decir la tarea 1: entre los instantes 36 y 51; y se actualizan los valores: $Ta = 36 + 15 = 51;$ $tr_2 = 16 - 15 = 1;\ x_2 = 1;\ c_1 = \left\lceil \frac{51}{20} + 10^{-6} \right\rceil = 3.$ Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
15. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$). Como $nu_1 < c_1 \rightarrow 2 < 3$, la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 3 < 51 + 10$ sí se cumple (no hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), y hay tiempo en el siguiente ciclo ($c_1 + 1 = 3 + 1 = 4$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $CTC_2 = \{2\}$ de la unidad $nu_2 + 1$ (1+1=2) de L2 ($ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i} \rightarrow 20 \cdot (3 + 1) \geq 51 + 10 + 1$), entonces, se va el Paso 4.1. En el Paso 4.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 51 y 52; y se actualizan los valores: $Ta = 51 + 1 = 52;\ tr_2 = 1 - 1 = 0;\ c_1 = \left\lceil \frac{52}{20} + 10^{-6} \right\rceil = 3.$ En el Paso 4.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (1+1=2) de L2, entonces, se actualizan los valores: $nu_2 = 1 + 1 = 2;\ tr_2 = 16;$ $x_2 = 0;\ MT[2][c_2] = MT[2][2] = 0 + 1 = 1;\ c_2 = \left\lceil \frac{52}{35} + 10^{-6} \right\rceil = 2;$ y se va al Paso 5.
16. Paso 5. Se actualiza el tiempo de reloj Ta, ya que se cumple uno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 2 < 4$), pero hay unidades por recuperar solo en la línea L1 ($nu_1 = c_1 \rightarrow 2 \neq 3$ y $nu_2 = c_2 \rightarrow 2 = 2$); conjunto 2: no hay unidades por recuperar en ambas líneas ($nu_1 < c_1 \rightarrow 2 < 3$ y $nu_2 < c_2 \rightarrow 2 \not< 2$); conjunto 3: hay unidades por recuperar solo en L1 ($nu_1 < c_1 \rightarrow 2 < 3$; $nu_2 = c_2 \rightarrow 2 = 2$), y la suma del tiempo de toda las tareas de la unidad en L1 no caben en el tiempo del ciclo c_1

($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 3 < 52 + 10$); como sí se cumple el conjunto 3, entonces se va al Paso 5.1. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 2 < 4$), entonces, se actualiza el tiempo $Ta = min\{ct_1 \cdot c_1,\ ct_2 \cdot c_2\} = min\{20 \cdot 3, 35 \cdot 2\} = 60$; y los ciclos $c_1 = \left\lceil \frac{60}{20} + 10^{-6} \right\rceil = 4$ y $c_2 = \left\lceil \frac{60}{35} + 10^{-6} \right\rceil = 2$. En el Paso 5.3, se vuelve al Paso 1.

17. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7;\ nu_2 < q_2 \rightarrow 2 < 4$), pero la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas es superior al lcm ($60 + (7-2) \cdot 10 + (4-2) \cdot 16 = 142 \nleq 140$), se va al Paso 6.
18. Paso 6. Como no se han realizado todas las unidades en ninguna de las líneas ($nu_1 < q_1 \rightarrow 2 < 7;\ nu_2 < q_2 \rightarrow 2 < 4$), el procedimiento finaliza sin una solución factible.

No se puede resolver el ejemplo con el procedimiento HB-1A porque se obtiene un momento de inactividad, entre los instantes 52 y 60, que hace que la solución deje de ser factible, ya que en esta estación multilínea el tiempo muerto es de solo 6 unidades de tiempo. En la Figura 26 se representa la solución que se obtendría si no se considerase como condición de parada el tiempo lcm. Se puede observar que la programación de las tareas terminaría en el instante 156, por lo que dicha solución excedería el tiempo lcm en 16 unidades de tiempo.

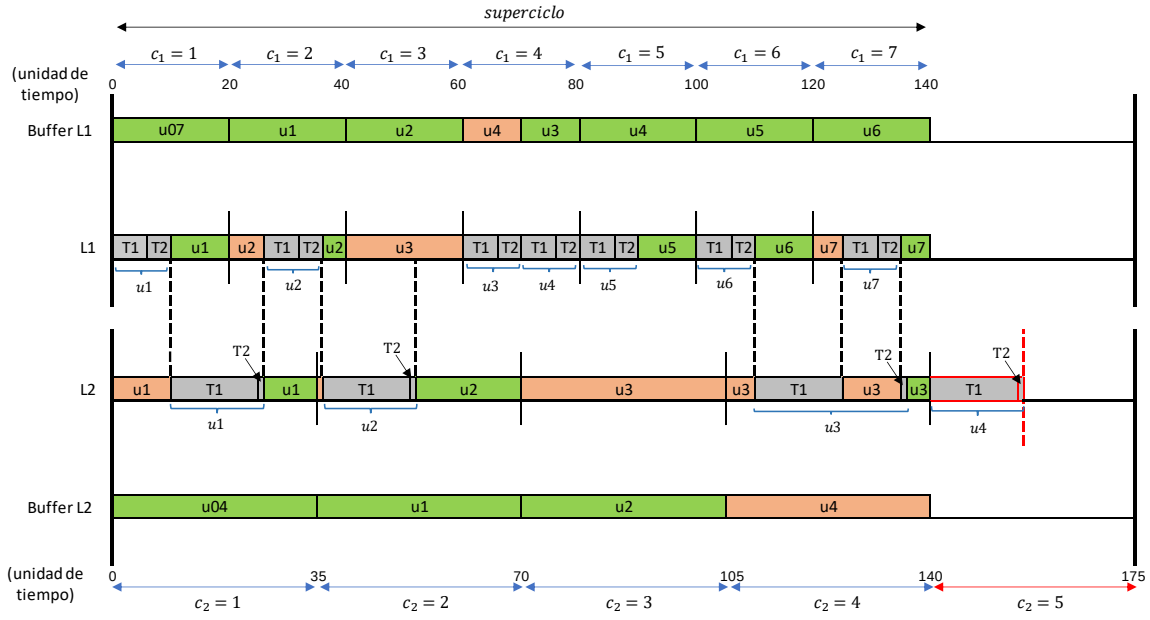

Figura 26. Solución obtenida para el ejemplo 2, al utilizar la heurística HB-1A sin el tiempo lcm como condición de parada.

Cabe destacar que resolviendo el ejemplo anterior con el modelo matemático V1, se obtiene un resultado óptimo con $a_h = \{1,1\}$ y $z_h = \{1,0\}$.

4.3.2 Heurística HB-1B

El procedimiento heurístico HB-1B propuesto en esta sección es similar al procedimiento HB-1A (Subsección 4.3.1). Pero, a diferencia del anterior, en esta heurística cabe la posibilidad de realizar más unidades que el ciclo actual, siempre que los tiempos de proceso lo permitan.

La solución proporcionada por la heurística HB-1B también determina el tamaño de los buffer de ambas líneas, la secuenciación y la programación de las tareas.

Adicionalmente a las premisas asumidas en la definición del (sub)problema de dimensionado de buffer (Subsección 2.2.2), en la presente heurística HB-1B se asumen las siguientes:

- El producto con el tiempo de ciclo más pequeño se asigna a la línea 1 (L1).
- El número de unidades ya realizadas en la línea h puede ser mayor al valor del ciclo c_h. De esta forma, en un ciclo c_h, se pueden realizar 0, 1, 2 o más unidades de un producto, siempre que los tiempos de procesos lo permitan.

La notación utilizada en la heurística HB-1B es la misma que la utilizada en HB-1A (Subsección 4.3.1), añadiendo una variable adicional:

Variable:

Id Tiempo muerto de la estación multilínea.

A continuación, se muestra los diferentes pasos del procedimiento HB-1B. Estos pasos se deben realizar de forma iterativa mientras haya unidades por realizar en L1 y/o L2 y el tiempo de reloj Ta sea menor que el tiempo lcm. Cuando se han realizado todas las unidades tanto de la línea L1 como de la línea L2, el procedimiento finaliza y se obtiene el valor del buffer a_h de la línea h. En esta heurística se ha considerado la posibilidad de finalizar el procedimiento antes de que se hayan realizado todas las unidades, esto ocurrirá en el caso de que no haya tiempo suficiente para realizar las tareas de todas las unidades pendientes y, así, no se dispondrá de una solución factible.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_2 = 0$; $tr_2 = tt_2$; $MT[h][x] = 0$ ($h \in H$; $x = 1, \dots, [max\{q_1, q_2\}]$); $Id = lcm - (tt_1 \cdot q_1 + tt_2 \cdot q_2)$.

1. Se comprueba si se cumplen las dos siguientes condiciones:
    - Hay unidades por realizar en L1 o L2, es decir: $nu_1 < q_1$ o $nu_2 < q_2$.

- La suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al tiempo lcm.

Si se cumplen las dos condiciones, se irá al Paso 2; de lo contrario se irá al Paso 6.

2. Se comprueba si se cumple las tres condiciones siguientes:
   - El número de unidades ya realizadas en L1 (nu_1) sea menor al número de unidades a realizar en L1 (q_1), es decir: $nu_1 < q_1$.
   - La suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ de L1 quepa en el ciclo actual de L1, esto es: $Ta + tt_1 \leq c_1 \cdot ct_1$.
   - Solo en el caso de que haya una unidad empezada en L2 ($x_2 = 1$) en el ciclo c_2: después de secuenciar todas las tareas de la unidad $nu_1 + 1$ de L1, debe haber tiempo suficiente en el ciclo c_2 de L2 para terminar las tareas que faltan de la unidad $nu_2 + 1$ empezada en L2, esto es: $Ta + tt_1 + tr_2 \leq c_2 \cdot ct_2$.

   De no cumplirse las tres condiciones anteriores, se irá al Paso 3; en caso contrario:

   2.1. Se secuencian y programan, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas de la unidad $nu_1 + 1$ de L1. Se actualizan valores de L1: $Ta = Ta + tt_1$; $nu_1 = nu_1 + 1$; $MT[1][c_1] = MT[1][c_1] + 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.

   2.2. Si no hay unidades por recuperar en L1 ($nu_1 \geq c_1$) y: 1) hay unidades por realizar en L2 ($nu_2 < q_2$) y el tiempo muerto Id es mayor a 0; o 2) ya se han realizado todas las unidades en L2 ($nu_2 = q_2$) y el tiempo muerto Id es mayor a $ct_1 \cdot c_1 - Ta$; entonces se va al Paso 3. De lo contrario, se vuelve al Paso 2.

3. Se comprueba si se cumple las dos condiciones siguientes:
   - El número de unidades ya realizadas en L2 (nu_2) sea menor al número de unidades a realizar en L2 (q_2), es decir: $nu_2 < q_2$.
   - La suma del tiempo de las tareas que quedan por realizar de la unidad $nu_2 + 1$ de L2 quepa en el ciclo actual de L2 ($Ta + tr_2 \leq ct_2 \cdot c_2$).

   De no cumplirse las dos condiciones anteriores, se irá al Paso 5; en caso contrario:

   3.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $x_2 = 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

   3.2. Si se han realizado todas las tareas de la unidad en L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se irá al Paso 5.

3.3. Si en L1 ya se han realizado todas las unidades ($nu_1 = q_1$), entonces se volverá al inicio del Paso 3.

3.4. Si la duración de la tarea i secuenciada en L2 es mayor que el tiempo de un ciclo de L1 ($t_{2i} > ct_1$), entonces se irá al Paso 2; en caso contrario se irá al Paso 4.

4. Se comprueba si se cumplen las dos siguientes condiciones:

   - Hay una unidad empezada en L2 ($x_2 = 1$).
   - Hay unidades por realizar en L1 ($nu_1 < q_1$).

   Y también se comprueba si se cumple una de las dos condiciones siguientes:

   - En caso de no haber unidades por recuperar en L1 ($nu_1 \geq c_1$): se comprueba si hay tiempo en el siguiente ciclo de L1 ($c_1 + 1$) para realizar todas las tareas de la siguiente unidad de L1 ($nu_1 + 1$) y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.
   - En caso de haber unidades por recuperar en L1 ($nu_1 < c_1$): se comprueba si en el ciclo actual de L1 no hay tiempo para realizar todas las tareas de la unidad de la L1 (es decir: $ct_1 \cdot c_1 < Ta + tt_1$), y que hay tiempo en el siguiente ciclo de L1 ($c_1 + 1$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.

   De no cumplirse las tres condiciones, se irá al Paso 5; en caso contrario:

   4.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias, y que hace que se cumpla una de las dos últimas condiciones anteriores. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

   4.2. Si se han realizado todas las tareas de la unidad de L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables en el siguiente orden: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se va al Paso 5; en caso contrario se va al Paso 4.

5. Actualización del reloj Ta: cuando no hay unidades por recuperar en ninguna de las líneas, o cuando no hay tiempo para realizar todas las tareas de una unidad de la línea h en el ciclo c_h. Se actualizará cuando se cumpla uno de los siguientes Conjuntos de condiciones:

   Conjunto 1:

   - Hay unidades por realizar en una o ambas líneas, es decir: $nu_1 < q_1$ y/o $nu_2 < q_2$.

- No hay unidades por recuperar en ninguna de las líneas, $nu_1 \geq c_1$ y $nu_2 \geq c_2$.

Conjunto 2:

- Hay unidades por realizar en ambas líneas, $nu_1 < q_1$ y $nu_2 < q_2$.
- La suma del tiempo de las tareas de la unidad de la L1 (tt_1) y de la unidad de la L2 (tr_2), no cabe en el tiempo del ciclo c_h de sus respectivas líneas, es decir: $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2$.

Conjunto 3:

- Hay unidades por realizar solo en L1 ($nu_1 < q_1$ y $nu_2 = q_2$).
- La suma del tiempo de todas las tareas de la unidad en L1 no cabe en el tiempo del ciclo c_1, es decir: $ct_1 \cdot c_1 < Ta + tt_1$.

Conjunto 4:

- Hay unidades por realizar solo en L2 ($nu_1 = q_1$ y $nu_2 < q_2$).
- La suma del tiempo de las tareas que faltan por procesar de la unidad $nu_2 + 1$ en L2 no caben en el tiempo del ciclo c_2, es decir: $ct_2 \cdot c_2 < Ta + tr_2$.

Si no se cumple ninguno de los conjuntos de condiciones anteriores, ir al Paso 1.

La operativa para la actualización del reloj Ta consiste en:

5.1. Si se cumple uno de los Conjuntos y hay unidades por realizar solo en una de las líneas, es decir $nu_1 < q_1$ o $nu_2 < q_2$, entonces, se actualizará el tiempo muerto Id, y el tiempo de reloj Ta al tiempo de ciclo de la línea h donde todavía quedan unidades por realizar: $Id = Id - (ct_h \cdot c_h - Ta)$; $Ta = ct_h \cdot c_h$; y se actualiza el ciclo c_h de la línea h.

5.2. Si se cumple uno de los Conjuntos y hay unidades por realizar en ambas líneas, es decir $nu_1 < q_1$ y $nu_2 < q_2$, entonces, se actualizará el tiempo muerto Id, y el tiempo de reloj Ta al tiempo de ciclo de la línea h menor entre L1 y L2: $Id = Id - (min\{ct_1 \cdot c_1, ct_2 \cdot c_2\} - Ta)$; $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\}$; y se actualizarán los ciclos c_h de las líneas.

5.3. Ir al Paso 1.

6. En este punto se termina el procedimiento heurístico, debido a uno de los siguientes casos:
   - Se han realizado todas las unidades en L1 y L2: $nu_1 = q_1$ y $nu_2 = q_2$. Se realiza el cálculo de los valores z_h y a_h (ver Figura 22 y Figura 23) y el procedimiento finaliza con una solución factible.
   - No hay tiempo suficiente para realizar las tareas que aún han de ejecutarse dentro del tiempo lcm. El procedimiento finaliza sin solución factible.

A continuación en la Figura 27 se muestra el pseudocódigo de la heurística:

Inicialización: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0\ ;\ c_1 = 1;\ c_2 = 1;\ x_2 = 0;\ tr_2 = tt_2;\quad MT[h][x] = 0\quad (h \in H;\ x = 1, \dots, [max\{q_1, q_2\}]);\ Id = lcm - (tt_1 \cdot q_1 + tt_2 \cdot q_2)$.

1. ***Mientras*** $((nu_1 < q_1$ o $nu_2 < q_2)$ y $((x_2 = 0$ y $Ta + (q_1 - nu_1) \cdot tt_1 + (q_2 - nu_2) \cdot tt_2 \le lcm)$ o $(x_2 = 1$ y $Ta + (q_1 - nu_1) \cdot tt_1 + (q_2 - nu_2) \cdot tt_2 - (tt_2 - tr_2) \le lcm)))$ *hacer*
   2. ***Mientras*** $(nu_1 < q_1$ y $Ta + tt_1 \le ct_1 \cdot c_1$ y $((x_2 = 0)$ o $(x_2 = 1$ y $(Ta + tt_1 + tr_2 \le ct_2 \cdot c_2))$ *hacer*
      2.1. Secuenciar y programar, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas de la unidad $nu_1 + 1$ de L1 y actualizar: $Ta = Ta + tt_1;\ nu_1 = nu_1 + 1;\ MT[1][c_1] = MT[1][c_1] + 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.

      2.2 ***Si*** $(nu_1 \ge c_1$ y $((nu_2 < q_2$ y $Id > 0)$ o $(nu_2 = q_2$ y $Id > ct_1 \cdot c_1 - Ta)))$ *entonces*
      Ir al Paso 3.

      Fin Mientras.
   3. ***Si*** $(nu_2 < q_2$ y $Ta + tr_2 \le ct_2 \cdot c_2)$ *entonces*
      3.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande, y actualizar: $Ta = Ta + t_{2i};\ tr_2 = tr_2 - t_{2i};\ x_2 = 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

      3.2. ***Si*** $(tr_2 = 0)$ *entonces*

      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5;

      3.3. ***Si*** $(nu_1 = q_1)$ *entonces* ir al Paso 3;

      3.4. ***Si*** $(t_{2i} > ct_1)$ *entonces* ir al Paso 2;
      En caso contrario, ir al Paso 4.

      En caso contrario, ir al Paso 5.
   4. ***Mientras*** $(x_2 = 1$ y $nu_1 < q_1$ y $((nu_1 \ge c_1$ y $ct_1 \cdot (c_1 + 1) \ge Ta + tt_1 + t_{2i})$ o $(nu_1 < c_1$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \ge Ta + tt_1 + t_{2i})))$ *hacer*
      4.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande y cumpla que $(nu_1 \ge c_1$ y $ct_1 \cdot (c_1 + 1) \ge Ta + tt_1 + t_{2i})$ o que $(nu_1 < c_1$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \ge Ta + tt_1 + t_{2i})$, entonces: $Ta = Ta + t_{2i};\ tr_2 = tr_2 - t_{2i};\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

      4.2. ***Si*** $(tr_2 = 0)$ *entonces*

      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5.

      Fin Mientras.
   5. ***Si*** $(((nu_1 < q_1$ y/o $nu_2 < q_2)$ y $nu_1 \ge c_1$ y $nu_2 \ge c_2)$ o $(nu_1 < q_1$ y $nu_2 < q_2$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2)$ o $(nu_1 < q_1$ y $nu_2 = q_2$ y $ct_1 \cdot c_1 < Ta + tt_1)$ o $(nu_1 = q_1$ y $nu_2 < q_2$ y $ct_2 \cdot c_2 < Ta + tr_2))$ *entonces*
      5.1. ***Si*** $(nu_1 = q_1$ y $nu_2 < q_2)$ *entonces*

      $Id = Id - (ct_2 \cdot c_2 - Ta);\ Ta = ct_2 \cdot c_2;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.

      Si $(nu_1 < q_1$ y $nu_2 = q_2)$ *entonces*

      $Id = Id - (ct_1 \cdot c_1 - Ta);\ Ta = ct_1 \cdot c_1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

      5.2. ***Si*** $(nu_1 < q_1$ y $nu_2 < q_2)$ *entonces*

      $Id = Id - (min\{ct_1 \cdot c_1, ct_2 \cdot c_2\} - Ta);\ Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\};$
      $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.

      5.3. Ir al Paso 1.

      En caso contrario, ir al Paso 1.

   Fin Mientras.
6. ***Si*** $(nu_1 = q_1$ y $nu_2 = q_2)$ *entonces*
   Calcular los valores del buffer z_h y a_h (ver Figura 22 y Figura 23).

   En caso contrario, el procedimiento finaliza sin solución factible.

Figura 27. Pseudocódigo de la heurística HB-1B.

Ejemplo ilustrativo del procedimiento HB-1B:

A continuación se presenta la resolución de un ejemplo para ilustrar el funcionamiento del procedimiento HB-1B. El ejemplo resuelto es el mismo que el segundo ejemplo presentado en la Subsección 4.3.1.

Datos:
$ct_1 = 20$; $ct_2 = 35$; $lcm = 140$;
$q_1 = 140/20 = 7$; $q_2 = 140/35 = 4$;
$T_1 = \{1,2\}$; $T_2 = \{1,2\}$;
$t_{11} = 6$; $t_{12} = 4$; $t_{21} = 15$; $t_{22} = 1$;
$tt_1 = (6 + 4) = 10$; $tt_2 = (15 + 1) = 16$;
$\mu = 10^{-6}$.

La relación de precedencia tanto para el conjunto de tareas de la línea 1 y de la línea 2 es la siguiente: la tarea 1 precede a la tarea 2.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_2 = 0$; $tr_2 = 16$; $MT[h][x] = 0$ $((h \in H;\ x = 1, \dots, [max\{7,4\}])$; $Id = 140 - (7 \cdot 10 + 4 \cdot 16) = 140 - 134 = 6$.

Procedimiento HB-1B:

1. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 0 < 7;\ nu_2 < q_2 \rightarrow 0 < 4$) y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($0 + (7 - 0) \cdot 10 + (4 - 0) \cdot 16 = 134 \leq 140$), se va al Paso 2.
2. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 0 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 0 + 10 \leq 20 \cdot 1$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 0 y 6, la tarea 2 entre los instantes 6 y 10; y se actualizan los valores: $Ta = 0 + 10 = 10$; $nu_1 = 0 + 1 = 1$; $MT[1][c_1] = MT[1][1] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{10}{20} + 10^{-6} \right\rceil = 1$; $c_2 = \left\lceil \frac{10}{35} + 10^{-6} \right\rceil = 1$. En Paso 2.2, como no hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 1 \geq 1$); hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 4$); y se cumple la condición $Id > 0 \rightarrow 6 > 0$; entonces se va al Paso 3.

3. Paso 3. Se comprueba si se cumplen las dos condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 4$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (0+1=1) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 10 + 16 \leq 35 \cdot 1$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 1: entre los instantes 10 y 25; y se actualizan los valores: $Ta = 10 + 15 = 25$; $tr_2 = 16 - 15 = 1$; $x_2 = 1$; $c_1 = \left\lceil \frac{25}{20} + 10^{-6} \right\rceil = 2$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
4. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$). Como $nu_1 < c_1 \rightarrow 1 < 2$, pero la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 2 \not< 25 + 10$ no se cumple (hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), entonces, se va al Paso 5.
5. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$ y $nu_2 < q_2 \rightarrow 0 < 4$), pero hay unidades por recuperar en la línea 1 y 2 ($nu_1 \geq c_1 \rightarrow 1 \not\geq 2$ y $nu_2 \geq c_2 \rightarrow 0 \not\geq 1$); conjunto 2: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$ y $nu_2 < q_2 \rightarrow 0 < 4$), pero el tiempo de las tareas pendientes ($tt_1 = 10$ y $tr_2 = 1$) de L1 y L2 sí que caben en sus ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 2 \not< 25 + 10$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 35 \cdot 1 \not< 25 + 1$); conjunto 3: hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$), pero también en L2 ($nu_2 = q_2 \rightarrow 0 \neq 4$); conjunto 4: hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 4$), pero también en L1 ($nu_1 = q_1 \rightarrow 1 \neq 7$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
6. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$; $nu_2 < q_2 \rightarrow 0 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($25 + (7 - 1) \cdot 10 + (4 - 0) \cdot 16 - (16 - 1) = 134 \leq 140$), se va al Paso 2.
7. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 25 + 10 \leq 20 \cdot 2$); y hay una unidad empezada en L2 ($x_2 = 1$), pero el tiempo de las tareas que quedan por procesar de la unidad de L2 es mayor al tiempo del ciclo c_2 ($Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2 \rightarrow 25 + 10 + 1 \not\leq 35 \cdot 1$); entonces se va al Paso 3.

8. Paso 3. Se comprueba si se cumplen las dos condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 0 < 4$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (0+1=1) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 25 + 1 \leq 35 \cdot 1$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 25 y 26; y se actualizan los valores: $Ta = 25 + 1 = 26$; $tr_2 = 1 - 1 = 0$; $x_2 = 1$; $c_1 = \left\lceil \frac{26}{20} + 10^{-6} \right\rceil = 2$. En el Paso 3.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (0+1=1) de L2, entonces, se actualizan los valores: $nu_2 = 0 + 1 = 1$; $tr_2 = 16$; $x_2 = 0$; $MT[2][c_2] = MT[2][1] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{26}{35} + 10^{-6} \right\rceil = 1$; y se va al Paso 5.
9. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$ y $nu_2 < q_2 \rightarrow 1 < 4$), pero hay unidades por recuperar en la línea 1 ($nu_1 \geq c_1 \rightarrow 1 \ngeq 2$ y $nu_2 \geq c_2 \rightarrow 1 \geq 1$); conjunto 2: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$ y $nu_2 < q_2 \rightarrow 1 < 4$), pero el tiempo de las tareas pendientes ($tt_1 = 10$ y $tr_2 = 16$) de L1 sí que caben en su ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 2 \nless 26 + 10$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 35 \cdot 1 < 26 + 16$); conjunto 3: hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$), pero también en L2 ($nu_2 = q_2 \rightarrow 1 \neq 4$); conjunto 4: hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 1 < 4$), pero también en L1 ($nu_1 = q_1 \rightarrow 1 \neq 7$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
10. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 7$; $nu_2 < q_2 \rightarrow 1 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($26 + (7 - 1) \cdot 10 + (4 - 1) \cdot 16 = 134 \leq 140$), se va al Paso 2.
11. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 1 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 26 + 10 \leq 20 \cdot 2$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (1+1=2) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 26 y 32, la tarea 2 entre los instantes 32 y 36; y se actualizan los valores: $Ta = 26 + 10 = 36$; $nu_1 = 1 + 1 = 2$; $MT[1][c_1] = MT[1][2] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{36}{20} + 10^{-6} \right\rceil = 2$; $c_2 = \left\lceil \frac{36}{35} + 10^{-6} \right\rceil = 2$. En Paso 2.2, como no hay unidades por

recuperar en L1 ($nu_1 \geq c_1 \rightarrow 2 \geq 2$), hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 1 < 4$), y se cumple la condición $Id > 0 \rightarrow 6 > 0$, entonces se va al Paso 3.

12. Paso 3. Se comprueba si se cumplen las dos condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 1 < 4$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (1+1=2) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 36 + 16 \leq 35 \cdot 2$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 1: entre los instantes 36 y 51; y se actualizan los valores: $Ta = 36 + 15 = 51$; $tr_2 = 16 - 15 = 1$; $x_2 = 1$; $c_1 = \left\lceil \frac{51}{20} + 10^{-6} \right\rceil = 3$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
13. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$). Como $nu_1 < c_1 \rightarrow 2 < 3$, la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 3 < 51 + 10$ sí se cumple (no hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), y hay tiempo en el siguiente ciclo ($c_1 + 1 = 3 + 1 = 4$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $CTC_2 = \{2\}$ de la unidad $nu_2 + 1$ (1+1=2) de L2 ($ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i} \rightarrow 20 \cdot (3 + 1) \geq 51 + 10 + 1$), entonces, se va el Paso 4.1. En Paso 4.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 51 y 52; y se actualizan los valores: $Ta = 51 + 1 = 52$; $tr_2 = 1 - 1 = 0$; $c_1 = \left\lceil \frac{52}{20} + 10^{-6} \right\rceil = 3$. En el Paso 4.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (1+1=2) de L2, entonces, se actualizan los valores: $nu_2 = 1 + 1 = 2$; $tr_2 = 16$; $x_2 = 0$; $MT[2][c_2] = MT[2][2] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{52}{35} + 10^{-6} \right\rceil = 2$; y se va al Paso 5.
14. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 2 < 4$), pero hay unidades por recuperar en la línea L1 ($nu_1 \geq c_1 \rightarrow 2 \ngeq 3$ y $nu_2 \geq c_2 \rightarrow 2 \geq 2$); conjunto 2: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 2 < 4$), pero el tiempo de las tareas pendientes ($tt_1 = 10$ y $tr_2 = 16$) de L2 sí que caben en su ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 3 < 52 + 10$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 35 \cdot 2 \nless 52 + 16$); conjunto 3: hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$), pero también en L2 ($nu_2 = q_2 \rightarrow 2 \neq 4$); conjunto 4: hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 2 < 4$), pero también en L1 ($nu_1 = q_1 \rightarrow 2 \neq 7$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.

15. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$; $nu_2 < q_2 \rightarrow 2 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm ($52 + (7-2) \cdot 10 + (4-2) \cdot 16 = 134 \leq 140$), se va al Paso 2.
16. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$); pero la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1 no cabe en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 52 + 10 \not\leq 20 \cdot 3$); entonces se va al Paso 3.
17. Paso 3. Se comprueba si se cumplen las dos condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 2 < 4$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (2+1=3) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 52 + 16 \leq 35 \cdot 2$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 1: entre los instantes 52 y 67; y se actualizan los valores: $Ta = 52 + 15 = 67$; $tr_2 = 16 - 15 = 1$; $x_2 = 1$; $c_1 = \left\lceil \frac{67}{20} + 10^{-6} \right\rceil = 4$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
18. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$). Como $nu_1 < c_1 \rightarrow 2 < 4$, pero la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 4 \not< 67 + 10$ no se cumple (hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), entonces, se va al Paso 5.
19. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 2 < 4$), pero hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 2 \not\geq 4$ y $nu_2 \geq c_2 \rightarrow 2 \geq 2$); conjunto 2: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 2 < 4$), pero el tiempo de las tareas pendientes ($tt_1 = 10$ y $tr_2 = 1$) de L1 y L2 sí que caben en su ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 4 \not< 67 + 10$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 35 \cdot 2 \not< 67 + 1$); conjunto 3: hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$), pero también en L2 ($nu_2 = q_2 \rightarrow 2 \neq 4$); conjunto 4: hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 2 < 4$), pero también en L1 ($nu_1 = q_1 \rightarrow 2 \neq 7$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
20. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$; $nu_2 < q_2 \rightarrow 2 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades

que quedan por realizar en ambas líneas no es superior al lcm $(67 + (7-2)\cdot 10 + (4-2)\cdot 16 - (16-1) = 134 \leq 140)$, se va al Paso 2.

21. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 67 + 10 \leq 20 \cdot 4$); y hay una unidad empezada en L2 ($x_2 = 1$), pero el tiempo de las tareas que quedan por procesar de la unidad de L2 es mayor al tiempo del ciclo c_2 ($Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2 \rightarrow 67 + 10 + 1 \nleq 35 \cdot 2$); entonces se va al Paso 3.
22. Paso 3. Se comprueba si se cumplen las dos condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 2 < 4$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (2+1=3) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 67 + 1 \leq 35 \cdot 2$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 67 y 68; y se actualizan los valores: $Ta = 67 + 1 = 68$; $tr_2 = 1 - 1 = 0$; $x_2 = 1$; $c_1 = \left\lceil \frac{68}{20} + 10^{-6} \right\rceil = 4$. En el Paso 3.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (2+1=3) de L2, entonces, se actualizan los valores: $nu_2 = 2 + 1 = 3$; $tr_2 = 16$; $x_2 = 0$; $MT[2][c_2] = MT[2][2] = 1 + 1 = 2$; $c_2 = \left\lceil \frac{68}{35} + 10^{-6} \right\rceil = 2$; y se va al Paso 5.
23. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 3 < 4$), pero hay unidades por recuperar en la línea L1 ($nu_1 \geq c_1 \rightarrow 2 \ngeq 4$ y $nu_2 \geq c_2 \rightarrow 3 \geq 2$); conjunto 2: hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 3 < 4$), pero el tiempo de las tareas pendientes ($tt_1 = 10$ y $tr_2 = 16$) de L1 sí que caben en su ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 4 \nless 68 + 10$ y $ct_2 \cdot c_2 < Ta + tr_2 \rightarrow 35 \cdot 2 < 68 + 16$); conjunto 3: hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$), pero también en L2 ($nu_2 = q_2 \rightarrow 3 \neq 4$); conjunto 4: hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 3 < 4$), pero también en L1 ($nu_1 = q_1 \rightarrow 2 \neq 7$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
24. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7;\ nu_2 < q_2 \rightarrow 3 < 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en ambas líneas no es superior al lcm $(68 + (7-2)\cdot 10 + (4-3)\cdot 16 = 134 \leq 140)$, se va al Paso 2.

25. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 2 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 68 + 10 \leq 20 \cdot 4$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (2+1=3) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 68 y 74, la tarea 2 entre los instantes 74 y 78; y se actualizan los valores: $Ta = 68 + 10 = 78$; $nu_1 = 2 + 1 = 3$; $MT[1][c_1] = MT[1][4] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{78}{20} + 10^{-6} \right\rceil = 4$; $c_2 = \left\lceil \frac{78}{35} + 10^{-6} \right\rceil = 3$. En Paso 2.2, como hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 3 \not\geq 4$), entonces se vuelve al Paso 2.
26. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 3 < 7$); pero la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (3+1=4) de L1 no cabe en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 78 + 10 \leq 20 \cdot 4$); entonces se va al Paso 3.
27. Paso 3. Se comprueba si se cumplen las dos condiciones. Como hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 3 < 4$); y la suma del tiempo de las tareas de la unidad $nu_2 + 1$ (3+1=4) de L2 caben en el ciclo c_2 ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 78 + 16 \leq 35 \cdot 3$), entonces, se va al Paso 3.1. En Paso 3.1 se secuencia la tarea de $CTC_2 = \{1\}$ con el tiempo más grande, es decir la tarea 1: entre los instantes 78 y 93; y se actualizan los valores: $Ta = 78 + 15 = 93$; $tr_2 = 16 - 15 = 1$; $x_2 = 1$; $c_1 = \left\lceil \frac{93}{20} + 10^{-6} \right\rceil = 5$. Los Pasos 3.2, 3.3 y 3.4 no se ejecutan ya que no se cumplen sus condiciones, entonces, se va al Paso 4.
28. Paso 4. Se comprueba que hay una unidad empezada en L2 ($x_2 = 1$) y que todavía hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 3 < 7$). Como $nu_1 < c_1 \rightarrow 3 < 5$, la condición $ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 5 < 93 + 10$ sí se cumple (no hay tiempo en el ciclo actual de L1 para secuenciar una unidad pendiente en L1), y hay tiempo en el siguiente ciclo ($c_1 + 1 = 5 + 1 = 6$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $CTC_2 = \{2\}$ de la unidad $nu_2 + 1$ (3+1=4) de L2 ($ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i} \rightarrow 20 \cdot (5 + 1) \geq 93 + 10 + 1$), entonces, se va el Paso 4.1. En el Paso 4.1 se secuencia la tarea de $CTC_2 = \{2\}$ con el tiempo más grande, que respete las precedencias, es decir la tarea 2: entre los instantes 93 y 94; y se actualizan los valores: $Ta = 93 + 1 = 94$; $tr_2 = 1 - 1 = 0$; $c_1 = \left\lceil \frac{94}{20} + 10^{-6} \right\rceil = 5$. En el Paso 4.2 como ya se han realizado todas las tareas ($tr_2 = 0$) de la unidad $nu_2 + 1$ (3+1=4) de L2, entonces, se actualizan los valores: $nu_2 = 3 + 1 = 4$; $tr_2 = 16$; $x_2 = 0$; $MT[2][c_2] = MT[2][3] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{94}{35} + 10^{-6} \right\rceil = 3$; y se va al Paso 5.

29. Paso 5. Se actualiza el tiempo de reloj Ta, ya que se cumple uno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en línea 1 ($nu_1 < q_1 \rightarrow 3 < 7$ y $nu_2 = q_2 \rightarrow 4 = 4$), pero hay unidades por recuperar en la línea 1 ($nu_1 \geq c_1 \rightarrow 3 \ngeq 5$ y $nu_2 \geq c_2 \rightarrow 4 \geq 3$); conjunto 2: no hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 2 < 7$ y $nu_2 < q_2 \rightarrow 4 \nless 4$); conjunto 3: hay unidades por realizar solo en L1 ($nu_1 < q_1 \rightarrow 3 < 7$ y $nu_2 = q_2 \rightarrow 4 = 4$), y el tiempo de las tareas pendientes de L1 ($tt_1 = 10$) no caben en su ciclo c_h ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 20 \cdot 5 < 94 + 10$); como sí se cumple el conjunto 3, entonces se va al Paso 5.1. Como hay unidades por realizar solo en la L1 ($nu_1 < q_1 \rightarrow 3 < 7$ y $nu_2 = q_2 \rightarrow 4 = 4$), entonces, se actualiza el tiempo $Id = Id - (ct_h \cdot c_h - Ta) = 6 - (20 \cdot 5 - 94) = 0$; $Ta = ct_1 \cdot c_1 = 20 \cdot 5 = 100$; y el ciclo $c_1 = \left\lceil \frac{100}{20} + 10^{-6} \right\rceil = 6$. En el Paso 5.3, se vuelve al Paso 1.
30. Paso 1. Se comprueba si el procedimiento cumple las dos condiciones. Como hay unidades por realizar en la línea 1 ($nu_1 < q_1 \rightarrow 3 < 7$; $nu_2 = q_2 \rightarrow 4 = 4$), y la suma del tiempo de reloj Ta y el tiempo total de las tareas de todas las unidades que quedan por realizar en la línea 1 no es superior al lcm ($100 + (7 - 3) \cdot 10 + (4 - 4) \cdot 16 = 140 \leq 140$), se va al Paso 2.
31. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 3 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (3+1=4) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 100 + 10 \leq 20 \cdot 6$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (3+1=4) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 100 y 106, la tarea 2 entre los instantes 106 y 110; y se actualizan los valores: $Ta = 100 + 10 = 110$; $nu_1 = 3 + 1 = 4$; $MT[1][c_1] = MT[1][6] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{110}{20} + 10^{-6} \right\rceil = 6$; $c_2 = \left\lceil \frac{110}{35} + 10^{-6} \right\rceil = 4$. Paso 2.2, como hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 4 \ngeq 6$), entonces se vuelve al Paso 2.
32. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 4 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (4+1=5) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 110 + 10 \leq 20 \cdot 6$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (4+1=5) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 110 y 116, la tarea 2 entre los instantes 116 y 120; y se actualizan los valores: $Ta = 110 + 10 = 120$; $nu_1 = 4 + 1 = 5$; $MT[1][c_1] = MT[1][6] = 1 +$

$1 = 2$; $c_1 = \left\lceil \frac{120}{20} + 10^{-6} \right\rceil = 7$; $c_2 = \left\lceil \frac{120}{35} + 10^{-6} \right\rceil = 4$. Paso 2.2, como hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 5 \ngeq 7$), entonces se vuelve al Paso 2.

33. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 5 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (5+1=6) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 120 + 10 \leq 20 \cdot 7$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (5+1=6) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 120 y 126, la tarea 2 entre los instantes 126 y 130; y se actualizan los valores: $Ta = 120 + 10 = 130$; $nu_1 = 5 + 1 = 6$; $MT[1][c_1] = MT[1][7] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{130}{20} + 10^{-6} \right\rceil = 7$; $c_2 = \left\lceil \frac{130}{35} + 10^{-6} \right\rceil = 4$. Paso 2.2, como hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 6 \ngeq 7$), entonces se vuelve al Paso 2.
34. Paso 2. Se comprueba si se cumplen las tres condiciones. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 6 < 7$); la suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ (6+1=7) de L1 caben en el ciclo actual ($Ta + tt_1 \leq ct_1 \cdot c_1 \rightarrow 130 + 10 \leq 20 \cdot 7$); y no hay una unidad empezada en L2 ($x_2 = 0$); entonces se va al Paso 2.1. En Paso 2.1 se secuencian todas las tareas de la unidad $nu_1 + 1$ (6+1=7) de L1, de mayor a menor tiempo de proceso y respetando las precedencias: la tarea 1 entre los instantes 130 y 136, la tarea 2 entre los instantes 136 y 140; y se actualizan los valores: $Ta = 130 + 10 = 140$; $nu_1 = 6 + 1 = 7$; $MT[1][c_1] = MT[1][7] = 1 + 1 = 2$; $c_1 = \left\lceil \frac{140}{20} + 10^{-6} \right\rceil = 8$; $c_2 = \left\lceil \frac{140}{35} + 10^{-6} \right\rceil = 5$. Paso 2.2, como hay unidades por recuperar en L1 ($nu_1 \geq c_1 \rightarrow 7 \ngeq 8$), entonces se vuelve al Paso 2.
35. Paso 2. Se comprueba si se cumplen las tres condiciones. Como no hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 7 \nless 7$), entonces se va al Paso 3.
36. Paso 3. Se comprueba si se cumple las dos condiciones. Como no hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 4 \nless 4$), entonces se va al Paso 5.
37. Paso 5. No se actualiza el tiempo de reloj Ta, ya que no se cumple ninguna condición, porque se han realizado todas las unidades en L1 y L2, entonces se va al Paso 1.
38. Paso 1. Como ya no hay unidades por realizar en las líneas L1 y L2 ($nu_1 < q_1 \rightarrow 7 \nless 7$ y $nu_2 < q_2 \rightarrow 4 \nless 4$), entonces se va al Paso 6.
39. Paso 6. Como se han realizado todas las unidades en ambas líneas, se calculan los valores a_h y z_h y el procedimiento finaliza con una solución factible.

De los resultados obtenidos $a_h = \{2,1\}$ y $z_h = \{2,0\}$, en la línea L1 se debe de disponer de un buffer de tamaño 2 unidades ($a_1 = 2$). Como se puede observar en la Figura 28,

habrá dos piezas terminadas en el buffer de L1 en los ciclos 1, 2, 3 y en parte de los ciclos 4 y 7. Se puede observar que la unidad $u6$ y $u7$ de L1 serán las unidades procesadas $u06$ y $u07$, respectivamente, del siguiente superciclo. Por otra parte, en la línea L2 se necesita un buffer de tamaño 1 unidad ($a_2 = 1$), en la cual habrá una unidad sin procesar ($u04$) al inicio del superciclo, instante 0. La unidad $u04$ de L2 es la primera unidad en ser procesada, desde el instante 10 al instante 26. Se puede observar que la unidad $u4$ de L2 será la unidad sin procesar $u04$ del siguiente superciclo. En la Figura 28, también se puede observar un momento de inactividad del operario, entre los instantes 94 y 100.

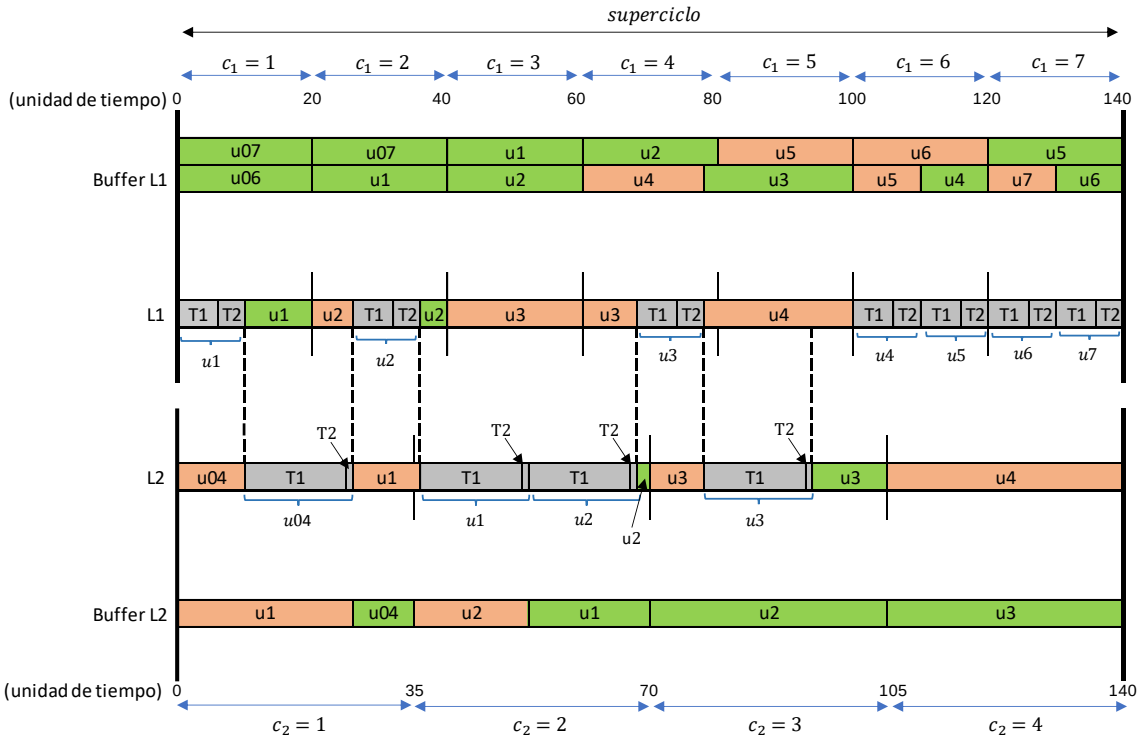

Figura 28. Representación de la solución obtenida para el ejemplo al utilizar la heurística HB-1B.

Cabe destacar que resolviendo el ejemplo anterior con el modelo matemático V1, se obtiene un resultado óptimo con $a_h = \{1,1\}$ y $z_h = \{1,0\}$. Se puede observar que la solución obtenida con en el procedimiento HB-1B no es óptima, pero sí factible.

4.3.3 Experimento computacional

Con el fin de evaluar el rendimiento de los procedimientos heurísticos propuestos en la Subsecciones 4.3.1 y 4.3.2, se ha realizado el mismo experimento computacional 1 que en la Subsección 4.2.4.2.

Los procedimientos desarrollados han sido codificados y ejecutados en Java SE8, y testeados en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

Los resultados obtenidos por las heurísticas HB-1A y HB-1B para el experimento computacional realizado se recogen en el Anexo B.2.1 (documento externo “Anexos”).

En la Tabla 4, que muestra la misma información que la Tabla 1, se puede observar que en general, los procedimientos heurísticos encuentran un mayor número de soluciones factibles (considerando las soluciones óptimas y no óptimas) que los modelos matemáticos. Cabe destacar, que las heurísticas HB-1A y HB-1B, han encontrado una solución factible en todos los ejemplares del Conjunto 1 y 4, a diferencia de los modelos matemáticos. Solo V2 ha encontrado una solución en todos los ejemplares del Conjunto 4. Además, HB-1B iguala el número de soluciones factibles obtenidas por V2 para el Conjunto 2, aunque V2 obtiene 9 soluciones óptimas más.

En términos globales, la heurística HB-1B, es el procedimiento que más soluciones factibles ha conseguido, 114 (considerando las soluciones óptimas y no óptimas).

Tabla 4. Resumen de los resultados obtenidos por los procedimientos V1, V2, V3, HB-1A y HB-1B.

Procedimiento	Conjunto 1 (60)	Conjunto 2 (20)	Conjunto 3 (20)	Conjunto 4 (20)	Total (120)
V1	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
V2	41 / 6 / 13	15 / 4 / 1	18 / 1 / 1	19 / 1 / 0	93 / 12 / 15
V3	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
HB-1A	58 / 2 / 0	9 / 9 / 2	3 / 11 / 6	14 / 6 / 0	84 / 28 / 8
HB-1B	33 / 27 / 0	6 / 13 / 1	3 / 12 / 5	10 / 10 / 0	52 / 62 / 6

En la Tabla 5 se muestra los mismos resultados que la Tabla 2, pero en este caso, solo 84 ejemplares han sido resueltos por todos los procedimientos.

De los resultados se observa que el modelo V2 es el que obtiene un mejor promedio del tamaño de los buffers, mejorando en un 40% y en un 68% los resultados obtenidos por HB-1A y HB-1B, respectivamente. Sin embargo, en términos de tiempo de cálculo computacional, V2 está muy lejos de los tiempos logrados por HB-1A y HB-1B. Además, las heurísticas HB-1A y HB-1B, como se puede observar en la Tabla 4, obtienen un mayor número de soluciones factibles que V2 (112 y 114 soluciones respectivamente, respecto a 105 de V2).

Tabla 5. Promedio del tamaño de los buffers y del tiempo de cálculo obtenidos por los procedimientos V1, V2, V3, HB-1A y HB-1B.

	V1	V2	V3	HB-1A	HB-1B
Promedio tamaño buffers	1.060	0.786	1.060	1.321	2.488
Tiempo promedio (s)	195.14	144.4	196.73	0.0004	0.0004

De los resultados se concluye que las heurísticas desarrolladas ayudan a encontrar un mayor número de soluciones factibles en un tiempo de cálculo muy pequeño. Sin embargo, el valor promedio del tamaño de los buffers de HB-1A y HB-1B son peores que los obtenidos por los modelos matemáticos. Cabe destacar, que en algunos ejemplares del Conjunto 1 y 4, los cuales solo han sido resueltos por los procedimientos V2, HB-1A y HB-1B (y por tanto no han sido considerados en la comparativa de la Tabla 5), V2 obtiene unos valores de a_h elevados.

4.4 Cóctel de heurísticas

4.4.1 Diseño del cóctel de heurísticas

Las heurísticas propuestas HB-1A y HB-1B en las subsecciones anteriores (Subsección 4.3.1 y Subsección 4.3.2, respectivamente), presentan un mejor rendimiento en términos de resultados (número de soluciones factibles y tiempo de cálculo) respecto a los modelos matemáticos. En especial, el tiempo de cálculo empleado por las heurísticas HB-1A y HB-1B para resolver los ejemplares es muy pequeño, con un tiempo promedio de cálculo para ambos procedimientos de 0.4 milisegundos.

Dada la breve duración de tiempo requerido por HB-1A y HB-1B para resolver un ejemplar, en esta sección se presenta un nuevo procedimiento llamado HB-1AB, el cual se basa en el concepto de cóctel de heurísticas. El cóctel de heurísticas consiste en hacer uso de todas las heurísticas rápidas disponibles (en nuestro caso HB-1A y HB-1B) para resolver un problema, obteniendo tantas soluciones como heurísticas rápidas disponibles, y de entre todas las soluciones, retener la mejor (García-Villoria et al., 2012).

El funcionamiento del procedimiento HB-1AB es el siguiente: para cada ejemplar se ejecutará, en el siguiente orden, la heurística HB-1A y HB-1B, y se retendrá la mejor solución. En el caso de que la primera heurística que se ejecute (i.e. HB-1A) obtenga una solución factible y un valor de la función objetivo igual a 0, con lo que se asegura el óptimo, el procedimiento HB-1AB finaliza y se devuelve la solución encontrada.

4.4.2 Experimento computacional

Con el fin de evaluar el rendimiento de HB-1AB, se ha realizado el mismo experimento computacional que en la Subsección 4.3.3.

El procedimiento desarrollado ha sido codificado y ejecutado en Java SE8, y testeado en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

Los resultados obtenidos por la heurística HB-1AB para el experimento computacional realizado se recogen en el Anexo B.2.1 (véase en el documento externo “Anexos”).

En la Tabla 6, que muestra la misma información que la Tabla 1, se puede observar que HB-1AB ha encontrado una solución factible (considerando las soluciones óptimas y no óptimas) en todos los ejemplares del Conjunto 1 y 4. Además, iguala el número de soluciones factibles obtenidas por V2 para el Conjunto 2, aunque V2 obtiene 6 soluciones óptimas más. En términos globales, HB-1AB es el procedimiento que, junto con HB-1B, más soluciones factibles ha conseguido (114, considerando soluciones óptimas y no óptimas), 22 más que V1 y V3, y 9 más que V2. Además, respecto a la heurística HB-1A, ha encontrado 2 soluciones factibles más, y respecto a HB-1B ha encontrado 23 soluciones óptimas más.

Tabla 6. Resumen de los resultados obtenidos por los procedimientos V1, V2, V3, HB-1A, HB-1B y HB-1AB.

Procedimiento	Conjunto 1 (60)	Conjunto 2 (20)	Conjunto 3 (20)	Conjunto 4 (20)	Total (120)
V1	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
V2	41 / 6 / 13	15 / 4 / 1	18 / 1 / 1	19 / 1 / 0	93 / 12 / 15
V3	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
HB-1A	58 / 2 / 0	9 / 9 / 2	3 / 11 / 6	14 / 6 / 0	84 / 28 / 8
HB-1B	33 / 27 / 0	6 / 13 / 1	3 / 12 / 5	10 / 10 / 0	52 / 62 / 6
HB-1AB	58 / 2 / 0	9 / 10 / 1	4 / 11 / 5	14 / 6 / 0	85 / 29 / 6

En la Tabla 7, que muestra los mismos resultados que la Tabla 2, se comparan los resultados de los 84 ejemplares que han sido resueltos por todos los procedimientos.

Tabla 7. Promedio del tamaño de los buffers y del tiempo de cálculo obtenidos por los procedimientos V1, V2, V3, HB-1A, HB-1B y HB-1AB.

	V1	V2	V3	HB-1A	HB-1B	HB-1AB
Promedio tamaño buffers	1.060	0.786	1.060	1.321	2.488	1.321
Tiempo promedio (s)	195.14	144.4	196.73	0.0004	0.0004	0.0007

De los resultados de la Tabla 7 se observa que HB-1AB obtiene un mejor (menor) promedio del tamaño del buffer que HB-1B e igual que HB-1A, y está cerca de V1 y V3. Sin embargo, V2 mejora estos resultados en un 40%. No obstante, en términos de tiempo de cálculo computacional, el modelo V2 está muy lejos de los tiempos logrados por HB-1A, HB-1B y HB-1AB. Además, la heurística HB-1AB, como se ha podido observar en la Tabla 6, obtiene un mayor número de soluciones factibles (114,

considerando soluciones óptimas y no óptimas) que V2 (105, considerando soluciones óptimas y no óptimas).

4.5 Metaheurística basada en recocido simulado

En este apartado se presenta un procedimiento basado en la metaheurística de recocido simulado (*Simulated Annealing* (SA)) para la resolución del (sub)problema de dimensionado de buffers, el cual se conocerá como HB-SA.

En la Subsección 4.5.1 se expone una introducción a la metaheurística SA. En la Subsección 4.5.2 se muestra el procedimiento HB-SA desarrollado para la secuenciación y programación de las tareas en una estación multilínea. La Subsección 4.5.3 presenta el experimento computacional realizado.

4.5.1 Introducción a la metaheurística SA

El SA es un algoritmo ampliamente utilizado para resolver problemas de optimización combinatoria, obteniendo exitosos resultados en sus aplicaciones (Gendreau and Potvin, 2010).

El SA es una técnica que se inspira en el proceso físico de enfriamiento de los metales propuesto por Metropolis et al. (1953). Posteriormente Kirkpatrick et al. (1983) introducen el SA en los problemas de optimización combinatoria.

El SA es un procedimiento de búsqueda local, el cual permite aceptar peores soluciones que la solución en curso, con el objetivo de evitar quedar atrapado en óptimos locales.

La metaheurística SA comienza con una solución inicial S_0, la cual se define inicialmente como la solución en curso S_C y a su vez como la mejor solución conocida hasta el momento S_B. Posteriormente, en cada iteración, una nueva solución S_N perteneciente al vecindario de la solución en curso $N(S_C)$ es generada. Si la solución vecina S_N no es peor que la solución en curso, entonces la solución vecina se convierte en la solución en curso (y si la solución en curso es, a su vez, mejor que la solución S_B, entonces la solución en curso pasa a ser la mejor solución conocida). En caso contrario, la solución vecina se convertirá en la solución en curso con una cierta probabilidad que dependerá de: 1) el valor de la solución en curso y de la vecina; y 2) el valor de un parámetro llamado temperatura t.

En el SA, la temperatura t influye en la probabilidad de aceptar soluciones vecinas peores que la solución en curso. A mayor temperatura, es más probable aceptar una peor solución vecina; en cambio a una menor temperatura es menos probable aceptar

una peor solución vecina. La temperatura t se va reduciendo gradualmente. El número de iteraciones para la cual la temperatura t permanece constante antes de ser reducida es itt.

En la Figura 29 se muestra un esquema general del SA, con una función objetivo a ser minimizada.

Sea $E(S)$ el valor de la función objetivo a ser minimizado de la solución en curso S.
Sea $N(S)$ el vecindario de la solución S.
Sea $A(t)$ el nuevo valor de la temperatura calculado a partir de la temperatura t.
0. Inicializar los parámetros:
    t_0 temperatura inicial
    itt número de iteraciones donde la temperatura permanece igual
1. $t \coloneqq t_0$
2. $S_C \coloneqq$ Obtener una solución inicial (S_0); $S_B \coloneqq S_C$;
3. ***Mientras*** el criterio de parada no se cumple *hacer*
4. $iter \coloneqq 0$
5. ***Mientras*** ($iter < itt$) *hacer*
6. Generar la solución vecina S_N de $N(S_C)$.
7. Sea $\Delta = E(S_N) - E(S_C)$
8. ***Si*** ($\Delta \leq 0$) *entonces*
9. $S_C \coloneqq S_N$
10. ***Si*** ($E(S_C) < E(S_B)$) *entonces*
11. $S_B \coloneqq S_C$
12. ***En caso contrario***
13. ***Si*** ($aleatorio[0,1) < e^{-\Delta/t}$) *entonces*
14. $S_C \coloneqq S_N$
15. $iter \coloneqq iter + 1$
16. ***Fin Mientras***
17. $t \coloneqq A(t)$
18. ***Fin Mientras***
19. Devuelve la solución S_B.

Figura 29. Pseudocódigo de la metaheurística SA.

4.5.2 Metaheurística HB-SA

El procedimiento HB-SA propuesto (ver Figura 30) se divide en dos fases.

En la primera fase se obtiene una solución inicial S_0, al utilizar un procedimiento basado en el cóctel de heurísticas (procedimiento HB-1AB). Si la solución aportada por el cóctel de heurística es factible y óptima, se finaliza el procedimiento HB-SA y se devuelve la solución obtenida. En caso contrario (solución no factible o factible pero no óptima) se pasa a la segunda fase.

La segunda fase (fase de búsqueda) empieza en el punto 4 del algoritmo de la Figura 30. En el SA propuesto, se permite partir de soluciones iniciales no factibles, por lo que

es posible que al llegar al criterio de parada se finalice el procedimiento sin una solución factible.

Sea $E(S)$ el valor de la función objetivo de la solución S.
Sea $N(S)$ el vecindario de la solución S.
Sea t la temperatura del sistema.
Sea max_time_{Bf} el tiempo máximo disponible de cálculo.
0. Inicializar los parámetros:
    t_0 temperatura inicial
    t_{min} temperatura mínima
    itt número de iteraciones donde la temperatura permanece igual
    α factor de enfriamiento
    GN número conceptualmente grande
    $t \coloneqq t_0$
1. Generación de una solución inicial (S_0)
2. ***Si*** ($E(S_0) = 0$ y $S_0 = factible$) *entonces*
3. $S_B \coloneqq S_0$; $E(S_B) \coloneqq E(S_0)$; ir al Paso 28;
4. $S_C \coloneqq S_0$; $E(S_C) \coloneqq E(S_0)$; $S_B \coloneqq S_C$; $E(S_B) \coloneqq E(S_C)$;
5. ***Si*** ($S_C = no\ factible$) *entonces*
6. $E(S_C) \coloneqq E(S_C) + incumplimiento \cdot GN$; $E(S_B) \coloneqq E(S_C)$;
7. Construcción de la lista de prioridad PL.
8. ***Mientras*** ($t \geq t_{min}$ y no se cumpla el tiempo max_time_{Bf}) *hacer*
9. $iter \coloneqq 0$
10. ***Mientras*** ($iter < itt$) *hacer*
11. Generar una solución vecina S_N de $N(S_C)$.
12. ***Si*** ($S_N = no\ factible$) *entonces*
13. $E(S_N) \coloneqq E(S_N) + incumplimiento \cdot GN$;
14. Sea $\Delta \coloneqq E(S_N) - E(S_c)$;
15. ***Si*** ($\Delta \leq 0$) *entonces*
16. $S_C \coloneqq S_N$; $E(S_C) \coloneqq E(S_N)$;
17. ***Si*** ($E(S_C) < E(S_B)$) *entonces*
18. $S_B \coloneqq S_C$; $E(S_B) \coloneqq E(S_C)$;
19. ***Si*** ($E(S_B) = 0$ y $S_B = factible$) *entonces*
20. ir al Paso 28;
21. ***En caso contrario***
22. ***Si*** ($aleatorio[0,1) < e^{-\Delta/t}$) *entonces*
23. $S_C \coloneqq S_N$; $E(S_C) \coloneqq E(S_N)$;
24. $iter \coloneqq iter + 1$
25. ***Fin Mientras***
26. $t \coloneqq \alpha \cdot t$
27. ***Fin Mientras***
28. ***Si*** ($S_B = factible$) *entonces*
29. Devuelve la solución S_B y el valor $E(S_B)$.
30. ***En caso contrario*** el procedimiento finaliza sin solución factible.

Figura 30. Pseudocódigo de la metaheurística HB-SA.

El SA propuesto utiliza un procedimiento de codificación de la solución basado en asignar prioridades a las tareas (Gen and Cheng, 2000; Hwang et al., 2008). Este procedimiento de codificación utiliza una lista de prioridad (PL) donde se consideran

todas las tareas de todas las unidades a procesar en un superciclo, y donde a cada tarea se le asigna un valor de prioridad.

El valor de prioridad que se le asigna (sin repetir) a cada tarea de la PL puede ser un número aleatorio o puede partir de una solución obtenida previamente. Al ordenar las tareas considerando su valor de prioridad (por ejemplo, de mayor a menor), se obtiene una secuencia de tareas que se utiliza para construir una solución. La variación de los valores de prioridad de una solución dada, proporciona, de esta forma, una solución vecina (punto 11 del algoritmo de la Figura 30).

El procedimiento propuesto de codificación de tareas ha sido utilizado en diferentes procedimientos presentados en la literatura del ALBP y PALBP (Gen and Cheng, 2000; Hwang et al., 2008; Özcan et al., 2009; Özcan et al., 2010; Hamzadayi and Yildiz, 2013; Özcan, 2019; Chutima and Jirachai, 2020; entre otros).

A continuación se especifican los elementos de la metaheurística HB-SA propuesta.

Generación de una solución inicial

Para la generación de una solución inicial S_0 se ha desarrollado un nuevo cóctel de heurísticas conocido como HB-2AB. La mejor solución aportada por HB-2AB será la solución inicial S_0.

El cóctel de heurísticas HB-2AB utiliza unas variaciones de los procedimientos HB-1A y HB-1B (Subsección 4.3.1 y 4.3.2, respectivamente), que se conocerán como HB-2A y HB-2B, respectivamente.

La principal diferencia entre los procedimientos HB-1A/1B y HB-2A/2B radica que, en las versiones HB-2A/2B, no se considera el tiempo lcm como una condición de parada. Por lo tanto, se programan todas las tareas asignadas a la estación multilínea aunque se realicen en un ciclo c_h superior al tamaño del lote q_h; obteniendo, en este caso, una solución no factible.

Seguidamente se muestra el funcionamiento de los procedimiento HB-2A y HB-2B.

Heurística HB-2A:

Adicionalmente a las premisas asumidas en la definición del (sub)problema de dimensionado de buffer (Subsección 2.2.2), en la presente heurística HB-2A se asumen las siguientes:

- El producto con el tiempo de ciclo más pequeño se asigna a la línea 1 (L1).

- El tiempo requerido para procesar todas las unidades de ambas líneas puede superar el tiempo lcm del sistema.
- En un ciclo c_h, se pueden realizar 0, 1, 2 o más unidades de un producto, siempre que los tiempos de procesos lo permitan y que el número de unidades ya realizadas en la línea h no sea mayor al ciclo c_h.

La notación utilizada en la heurística HB-2A es la misma que la utilizada en HB-1A (Subsección 4.3.1).

A continuación, se muestra los diferentes pasos del procedimiento HB-2A. Estos pasos se deben realizar de forma iterativa mientras haya unidades por realizar en L1 y/o L2. Cuando se han realizado todas las unidades tanto de la línea L1 como de la línea L2, el procedimiento finaliza y se obtiene la duración de la programación de las tareas y el valor del buffer a_h de la línea h.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_2 = 0$; $tr_2 = tt_2$; $MT[h][x] = 0$ ($h \in H$; $x = 1, \dots, [max\{q_1, q_2\}]$);

1. Se comprueba si se cumple la siguiente condición:
   - Hay unidades por realizar en L1 o L2, es decir: $nu_1 < q_1$ o $nu_2 < q_2$.

   Si se cumple la condición, se irá al Paso 2; de lo contrario se irá al Paso 6.

2. Se comprueba si se cumplen las cuatro siguientes condiciones:
   - El número de unidades ya realizadas en L1 (nu_1) sea menor al número de unidades a realizar en L1 (q_1), es decir: $nu_1 < q_1$.
   - El número de unidades ya realizadas en L1 (nu_1) sea menor al mínimo entre el ciclo (c_1) y el lote (q_1) de L1, es decir: $nu_1 < min\{c_1, q_1\}$.
   - La suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ de L1 quepa en el ciclo actual de L1, esto es: $Ta + tt_1 \leq ct_1 \cdot c_1$.
   - Solo en el caso de que haya una unidad empezada en L2 ($x_2 = 1$) en el ciclo c_2, después de secuenciar todas las tareas de la unidad $nu_1 + 1$ de la L1, debe haber tiempo suficiente en el ciclo c_2 de L2 para terminar las tareas que faltan de la unidad $nu_2 + 1$ empezada en L2, esto es: $Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2$.

   De no cumplirse las cuatro condiciones, se irá al Paso 3; en caso contrario:

   2.1. Se secuencian y programan, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas

de la unidad $nu_1 + 1$ de L1. Se actualizan valores de L1: $Ta = Ta + tt_1$; $nu_1 = nu_1 + 1$; $MT[1][c_1] = MT[1][c_1] + 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$

2.2. Volver al Paso 2.

3. Se comprueba si se cumplen las tres siguientes condiciones:
   - El número de unidades ya realizadas en L2 (nu_2) sea menor al número de unidades a realizar en L2 (q_2), es decir: $nu_2 < q_2$.
   - El número de unidades ya realizadas en L2 (nu_2) sea menor al mínimo entre el ciclo (c_2) y el lote (q_2) de L2, es decir: $nu_2 < min\{c_2, q_2\}$.
   - La suma del tiempo de las tareas que quedan por realizar de la unidad $nu_2 + 1$ de L2 quepa en el ciclo actual de L2 ($Ta + tr_2 \leq ct_2 \cdot c_2$).

   De no cumplirse las tres condiciones, se irá al Paso 5; en caso contrario:

   3.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $x_2 = 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

   3.2. Si se han realizado todas las tareas de la unidad en L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se irá al Paso 5.

   3.3. Si en L1 ya se han realizado todas las unidades ($nu_1 = q_1$), entonces se volverá al inicio del Paso 3.

   3.4. Si la duración de la tarea i secuenciada en L2 es mayor que el tiempo de un ciclo de L1 ($t_{2i} > ct_1$), entonces se irá al Paso 2; en caso contrario se irá al Paso 4.

4. Se comprueba si se cumplen las dos siguientes condiciones:
   - Hay una unidad empezada en L2 ($x_2 = 1$).
   - Hay unidades por realizar en L1 ($nu_1 < q_1$).

   Y también se comprueba si se cumple una de las dos condiciones siguientes:
   - En caso de no haber unidades por recuperar en L1 ($nu_1 = min\{c_1, q_1\}$): se comprueba si hay tiempo en este o el siguiente ciclo de L1 ($c_1 + 1$) para realizar todas las tareas de la siguiente unidad de L1 ($nu_1 + 1$) y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.
   - En caso de haber unidades por recuperar en L1 ($nu_1 < min\{c_1, q_1\}$): se comprueba si en el ciclo actual de L1 no hay tiempo para realizar todas las tareas de la unidad de L1 (es decir: $ct_1 \cdot c_1 < Ta + tt_1$), y que hay tiempo en

el siguiente ciclo de L1 ($c_1 + 1$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.

De no cumplirse las tres condiciones, se irá al Paso 5; en caso contrario:

4.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias, y que hace que se cumpla una de las dos últimas condiciones anteriores. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

4.2. Si se han realizado todas las tareas de la unidad de L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables en el siguiente orden: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se va al Paso 5; en caso contrario se va al Paso 4.

5. Actualización del reloj Ta: cuando no hay unidades por recuperar en ninguna de las líneas, o cuando no hay tiempo para realizar todas las tareas de una unidad de la línea h en el ciclo c_h. Se actualizará cuando se cumpla uno de los siguientes Conjuntos de condiciones:

   Conjunto 1:

   - Hay unidades por realizar en una o ambas líneas, es decir: $nu_1 < q_1$ y/o $nu_2 < q_2$.
   - No hay unidades por recuperar en ninguna de las líneas, $nu_1 = min\{c_1, q_1\}$ y $nu_2 = min\{c_2, q_2\}$.

   Conjunto 2:

   - Hay unidades por recuperar en ambas líneas, $nu_1 < min\{c_1, q_1\}$ y $nu_2 < min\{c_2, q_2\}$.
   - La suma del tiempo de las tareas de la unidad de L1 (tt_1) y de la unidad de L2 (tr_2), no cabe en el tiempo del ciclo c_h de sus respectivas líneas, es decir: $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2$.

   Conjunto 3:

   - Hay unidades por recuperar solo en L1 ($nu_1 < min\{c_1, q_1\}$ y $nu_2 = min\{c_2, q_2\}$).
   - La suma del tiempo de todas las tareas de la unidad en L1 no cabe en el tiempo del ciclo c_1, es decir: $ct_1 \cdot c_1 < Ta + tt_1$.

   Conjunto 4:

   - Hay unidades por recuperar solo en L2 ($nu_1 = min\{c_1, q_1\}$ y $nu_2 < min\{c_2, q_2\}$).

- La suma del tiempo de las tareas que faltan por procesar de la unidad $nu_2 + 1$ en L2 no cabe en el tiempo del ciclo c_2, es decir: $ct_2 \cdot c_2 < Ta + tr_2$.

Si no se cumple ninguno de los conjuntos de condiciones anteriores, ir al Paso 1.

La operativa para la actualización del reloj Ta consiste en:

5.1. Si se cumple uno de los Conjuntos y hay unidades por realizar solo en una de las líneas, es decir: $nu_1 < q_1$ o $nu_2 < q_2$, entonces, se actualizará el tiempo de reloj Ta al tiempo de ciclo de la línea h donde todavía quedan unidades por realizar: $Ta = ct_h \cdot c_h$; y se actualiza el ciclo c_h de la línea h.

5.2. Si se cumple uno de los Conjuntos y hay unidades por realizar en ambas líneas, es decir: $nu_1 < q_1$ y $nu_2 < q_2$, entonces, se actualizará el tiempo de reloj Ta al tiempo de ciclo de la línea h más cercano entre L1 y L2: $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\}$; y se actualizarán los ciclos c_h.

5.3. Ir al Paso 1.

6. En este punto se termina el procedimiento heurístico, debido al siguiente caso:
   - Se han realizado todas las unidades en L1 y L2: $nu_1 = q_1$ y $nu_2 = q_2$. Se realiza el cálculo de los valores z_h y a_h (ver Figura 22 y Figura 23) y el procedimiento finaliza con una solución factible (si el tiempo de la programación no supera el tiempo lcm) o no factible (si el tiempo de la programación supera el tiempo lcm).

A continuación en la Figura 31 se muestra el pseudocódigo de la heurística HB-2A:

Inicialización: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0\ ;\ c_1 = 1;\ c_2 = 1;\ x_2 = 0;\ tr_2 = tt_2;\ MT[h][x] = 0\ \left(h \in H;\ x = 1, \dots, \left\lceil max\{q_1, q_2\}\right\rceil\right)$.

1. ***Mientras*** ($nu_1 < q_1$ o $nu_2 < q_2$) *hacer*
   2. ***Mientras*** ($nu_1 < q_1$ y $nu_1 < min\{c_1, q_1\}$ y $Ta + tt_1 \leq ct_1 \cdot c_1$ y (($x_2 = 0$) o ($x_2 = 1$ y ($Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2$)) *hacer*
      2.1. Secuenciar y programar, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas de la unidad $nu_1 + 1$ de L1 y actualizar: $Ta = Ta + tt_1;\ nu_1 = nu_1 + 1;\ MT[1][c_1] = MT[1][c_1] + 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;$ $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      Fin Mientras
   3. ***Si*** ($nu_2 < q_2$ y $nu_2 < min\{c_2, q_2\}$ y $Ta + tr_2 \leq ct_2 \cdot c_2$) *entonces*
      3.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande, y actualizar: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $x_2 = 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      3.2. ***Si*** ($tr_2 = 0$) *entonces*
      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5;
      3.3. ***Si*** ($nu_1 = q_1$) *entonces* ir al Paso 3;
      3.4. ***Si*** ($t_{2i} > ct_1$) *entonces* ir al Paso 2;
      En caso contrario, ir al Paso 4.
      En caso contrario, ir al Paso 5.
   4. ***Mientras*** ($x_2 = 1$ y $nu_1 < q_1$ y (($nu_1 = min\{c_1, q_1\}$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$) o ($nu_1 < min\{c_1, q_1\}$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$))) *hacer*
      4.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande y cumpla que ($nu_1 = c_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$) o que ($nu_1 < c_1$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$), entonces: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      4.2. ***Si*** ($tr_2 = 0$) *entonces*
      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5.
      Fin Mientras
   5. ***Si*** ((($nu_1 < q_1$ y/o $nu_2 < q_2$) y $nu_1 = min\{c_1, q_1\}$ y $nu_2 = min\{c_2, q_2\}$) o ($nu_1 < min\{c_1, q_1\}$ y $nu_2 < min\{c_2, q_2\}$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2$) o ($nu_1 < min\{c_1, q_1\}$ y $nu_2 = min\{c_2, q_2\}$ and $ct_1 \cdot c_1 < Ta + tt_1$) o ($nu_1 = min\{c_1, q_1\}$ y $nu_2 < min\{c_2, q_2\}$ y $ct_2 \cdot c_2 < Ta + tr_2$)) *entonces*
      5.1. ***Si*** ($nu_1 = q_1$ y $nu_2 < q_2$) *entonces*
      $Ta = ct_2 \cdot c_2$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      Si ($nu_1 < q_1$ y $nu_2 = q_2$) *entonces*
      $Ta = ct_1 \cdot c_1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      5.2. ***Si*** ($nu_1 < q_1$ y $nu_2 < q_2$) *entonces*
      $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\}$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      5.3. Ir al Paso 1.
      En caso contrario ir al Paso 1.
   Fin Mientras
6. Calcular los valores del buffer z_h y a_h (ver Figura 22 y Figura 23).
   Si ($Ta \leq lcm$) *entonces*
   El procedimiento finaliza con una solución factible.
   En caso contrario, el procedimiento finaliza sin solución factible.

Figura 31. Pseudocódigo de la heurística HB-2A.

Heurística HB-2B:

Adicionalmente a las premisas asumidas en la definición del (sub)problema de dimensionado de buffer (Subsección 2.2.2), en la presente heurística HB-2B se asumen las siguientes:

- El producto con el tiempo de ciclo más pequeño se asigna a la línea 1 (L1).
- El tiempo requerido para procesar todas las unidades de ambas líneas, puede superar el tiempo lcm del sistema.
- El número de unidades ya realizadas en la línea h puede ser mayor al valor del ciclo c_h. De esta forma, en un ciclo c_h, se pueden realizar 0, 1, 2 o más unidades de un producto, siempre que los tiempos de procesos lo permitan.

Las variables y los parámetros utilizados en la heurística HB-2B son los mismos que los utilizados en la heurística HB-1B (Subsección 4.3.2).

A continuación, se muestra los diferentes pasos del procedimiento HB-2B. Estos pasos se deben realizar de forma iterativa mientras haya unidades por realizar en L1 y/o L2. Cuando se han realizado todas las unidades tanto de la línea L1 como de la línea L2, el procedimiento finaliza y se obtiene la duración de la programación de las tareas y el valor del buffer a_h de la línea h.

Inicialización de variables: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0;\ c_1 = 1;\ c_2 = 1;\ x_2 = 0;\ tr_2 = tt_2;$ $MT[h][x] = 0$ $(h \in H;\ x = 1, \dots, [max\ \{q_1, q_2\}]);\ Id = lcm - (tt_1 \cdot q_1 + tt_2 \cdot q_2)$.

1. Se comprueba si se cumple la siguiente condición:
   - Hay unidades por realizar en L1 o L2, es decir: $nu_1 < q_1$ o $nu_2 < q_2$.

   Si se cumple la condición, se irá al Paso 2; de lo contrario se irá al Paso 6.

2. Se comprueba si se cumplen las tres siguientes condiciones:
   - El número de unidades ya realizadas en L1 (nu_1) sea menor al número de unidades a realizar en L1 (q_1), es decir: $nu_1 < q_1$.
   - La suma del tiempo de todas las tareas de la unidad $nu_1 + 1$ de L1 quepa en el ciclo actual de L1, esto es: $Ta + tt_1 \leq ct_1 \cdot c_1$.
   - Solo en el caso de que haya una unidad empezada en L2 ($x_2 = 1$) en el ciclo c_2, después de secuenciar todas las tareas de la unidad $nu_1 + 1$ de la L1, debe haber tiempo suficiente en el ciclo c_2 de L2 para terminar las tareas que faltan de la unidad $nu_2 + 1$ empezada en L2, esto es: $Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2$.

   De no cumplirse las tres condiciones, se irá al Paso 3; en caso contrario:

2.1. Se secuencian y programan, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas de la unidad $nu_1 + 1$ de L1. Se actualizan valores de L1: $Ta = Ta + tt_1$; $nu_1 = nu_1 + 1$; $MT[1][c_1] = MT[1][c_1] + 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.

2.2. Si no hay unidades por recuperar en L1 ($nu_1 \geq min\{c_1, q_1\}$) y: 1) hay unidades por realizar en L2 ($nu_2 < q_2$) y el tiempo muerto Id es mayor a 0; o 2) ya se han realizado todas las unidades en L2 ($nu_2 = q_2$) y el tiempo muerto Id es mayor a: $ct_1 \cdot c_1 - Ta$; entonces se va al Paso 3. De lo contrario, se vuelve al Paso 2.

3. Se comprueba si se cumplen las dos siguientes condiciones:
   - El número de unidades ya realizadas en L2 (nu_2) sea menor al número de unidades a realizar en L2 (q_2), es decir: $nu_2 < q_2$.
   - La suma del tiempo de las tareas que quedan por realizar de la unidad $nu_2 + 1$ de L2 quepa en el ciclo actual de L2 ($Ta + tr_2 \leq ct_2 \cdot c_2$).

   De no cumplirse las dos condiciones, se irá al Paso 5; en caso contrario:

   3.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $x_2 = 1$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

   3.2. Si se han realizado todas las tareas de la unidad en L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y ser irá al Paso 5.

   3.3. Si en L1 ya se han realizado todas las unidades ($nu_1 = q_1$), entonces se volverá al inicio del Paso 3.

   3.4. Si la duración de la tarea i secuenciada en L2 es mayor que el tiempo de un ciclo de L1 ($t_{2i} > ct_1$), entonces se irá al Paso 2; en caso contrario se irá al Paso 4.

4. Se comprueba si se cumplen las dos siguientes condiciones:
   - Hay una unidad empezada en L2 ($x_2 = 1$).
   - Hay unidades por realizar en L1 ($nu_1 < q_1$).

   Y también se comprueba si se cumple una de las dos condiciones siguientes:
   - En caso de no haber unidades por recuperar en L1 ($nu_1 \geq min\{c_1, q_1\}$): se comprueba si hay tiempo en este o el siguiente ciclo de L1 ($c_1 + 1$) para realizar todas las tareas de la siguiente unidad de L1 ($nu_1 + 1$) y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.

- En caso de haber unidades por recuperar en L1 ($nu_1 < min\{c_1, q_1\}$): se comprueba si en el ciclo actual de L1 no hay tiempo para realizar todas las tareas de la unidad de la L1 (es decir: $ct_1 \cdot c_1 < Ta + tt_1$), y que hay tiempo en el siguiente ciclo de L1 ($c_1 + 1$) para realizar las tareas de la unidad pendiente en L1 y realizar alguna tarea $i \in CTC_2$ de la unidad actual de L2, es decir: $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$.

De no cumplirse las tres condiciones, se irá al Paso 5; en caso contrario:

4.1. Se programa en el instante Ta, la tarea $i \in CTC_2$ con el tiempo de proceso más grande que respete las precedencias, y que hace que se cumpla una de las dos últimas condiciones anteriores. Se actualizarán las siguientes variables: $Ta = Ta + t_{2i}$; $tr_2 = tr_2 - t_{2i}$; $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.

4.2. Si se han realizado todas las tareas de la unidad de L2 ($tr_2 = 0$), entonces se actualizarán las siguientes variables en el siguiente orden: $nu_2 = nu_2 + 1$; $tr_2 = tt_2$; $x_2 = 0$; $MT[2][c_2] = MT[2][c_2] + 1$; $c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; y se va al Paso 5; en caso contrario se va al Paso 4.

5. Actualización del reloj Ta: cuando no hay unidades por recuperar en ninguna de las líneas, o cuando no hay tiempo para realizar todas las tareas de una unidad de la línea h en el ciclo c_h. Se actualizará cuando se cumpla uno de los siguientes Conjuntos de condiciones:

   Conjunto 1:

   - Hay unidades por realizar en una o ambas líneas, es decir: $nu_1 < q_1$ y/o $nu_2 < q_2$.
   - No hay unidades por recuperar en ninguna de las líneas, $nu_1 \geq min\{c_1, q_1\}$ y $nu_2 \geq min\{c_2, q_2\}$.

   Conjunto 2:

   - Hay unidades por realizar en ambas líneas, $nu_1 < q_1$ y $nu_2 < q_2$.
   - La suma del tiempo de las tareas de la unidad de L1 (tt_1) y de la unidad de L2 (tr_2), no cabe en el tiempo del ciclo c_h de sus respectivas líneas, es decir: $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2$.

   Conjunto 3:

   - Hay unidades por realizar solo en L1 ($nu_1 < q_1$ y $nu_2 = q_2$).
   - La suma del tiempo de todas las tareas de la unidad en L1 no cabe en el tiempo del ciclo c_1, es decir: $ct_1 \cdot c_1 < Ta + tt_1$.

   Conjunto 4:

   - Hay unidades por realizar solo en L2 ($nu_1 = q_1$ y $nu_2 < q_2$).

- La suma del tiempo de las tareas que faltan por procesar de la unidad $nu_2 + 1$ en L2 no cabe en el tiempo del ciclo c_2, es decir: $ct_2 \cdot c_2 < Ta + tr_2$.

Si no se cumple ninguno de los conjuntos de condiciones anteriores, ir al Paso 1.

La operativa para la actualización del reloj Ta consiste en:

5.1. Si se cumple uno de los Conjuntos y hay unidades por realizar solo en una de las líneas, es decir: $nu_1 < q_1$ o $nu_2 < q_2$, entonces, se actualizará el tiempo muerto Id, y el tiempo de reloj Ta al tiempo de ciclo de la línea h donde todavía quedan unidades por realizar: $Id = Id - (ct_h \cdot c_h - Ta)$; $Ta = ct_h \cdot c_h$; y se actualiza el ciclo c_h de la línea h.

5.2. Si se cumple uno de los Conjuntos y hay unidades por realizar en ambas líneas, es decir: $nu_1 < q_1$ y $nu_2 < q_2$, entonces, se actualizará el tiempo muerto Id, y el tiempo de reloj Ta al tiempo de ciclo de la línea h más cercano entre L1 y L2: $Id = Id - (min\{ct_1 \cdot c_1, ct_2 \cdot c_2\} - Ta)$; $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\}$; y se actualizarán los ciclos c_h de las líneas.

5.3. Ir al Paso 1.

6. En este punto se termina el procedimiento heurístico, debido al siguiente caso:
   - Se han realizado todas las unidades en L1 y L2: $nu_1 = q_1$ y $nu_2 = q_2$. Se realiza el cálculo de los valores z_h y a_h (ver Figura 22 y Figura 23) y el procedimiento finaliza con una solución factible (si el tiempo de la programación no supera el tiempo lcm) o no factible (si el tiempo de la programación supera el tiempo lcm).

En la Figura 32 se muestra el pseudocódigo de la heurística HB-2B.

Si el valor de la solución inicial $E(S_0)$ es igual a 0 (es decir, no se necesita ningún buffer, $E(S_0) = \sum_{h \in H} a_h = 0$) y factible (se respeta el tiempo lcm), entonces la solución obtenida es óptima y finaliza el procedimiento. En caso contrario, se inicia la segunda fase del procedimiento.

Inicialización: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0\ ;\ c_1 = 1;\ c_2 = 1;\ x_2 = 0;\ tr_2 = tt_2;\quad MT[h][x] = 0\quad (h \in H;\ x = 1, \dots, [max\{q_1, q_2\}]);\ Id = lcm - (tt_1 \cdot q_1 + tt_2 \cdot q_2)$.

1. ***Mientras*** ($nu_1 < q_1$ o $nu_2 < q_2$) *hacer*
   2. ***Mientras*** ($nu_1 < q_1$ y $Ta + tt_1 \leq ct_1 \cdot c_1$ y (($x_2 = 0$) o ($x_2 = 1$ y $(Ta + tt_1 + tr_2 \leq ct_2 \cdot c_2)$)) *hacer*
      2.1. Secuenciar y programar, una detrás de otra, en orden decreciente de tiempo de proceso y respetando las relaciones de precedencia, todas las tareas de la unidad $nu_1 + 1$ de L1 y actualizar: $Ta = Ta + tt_1;\ nu_1 = nu_1 + 1;\ MT[1][c_1] = MT[1][c_1] + 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      2.2 ***Si*** ($nu_1 \geq min\ \{c_1, q_1\}$ y (($nu_2 < q_2$ y $Id > 0$) o ($nu_2 = q_2$ y $Id > ct_1 \cdot c_1 - Ta$))) *entonces* Ir al Paso 3.
      Fin Mientras
   3. ***Si*** ($nu_2 < q_2$ y $Ta + tr_2 \leq ct_2 \cdot c_2$) *entonces*
      3.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande, y actualizar: $Ta = Ta + t_{2i};\ tr_2 = tr_2 - t_{2i};\ x_2 = 1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      3.2. ***Si*** ($tr_2 = 0$) *entonces*
      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5;
      3.3. ***Si*** ($nu_1 = q_1$) *entonces* ir al Paso 3;
      3.4. ***Si*** ($t_{2i} > ct_1$) *entonces* ir al Paso 2;
      En caso contrario, ir al Paso 4.
      En caso contrario, ir al Paso 5.
   4. ***Mientras*** ($x_2 = 1$ y $nu_1 < q_1$ y (($nu_1 \geq min\ \{c_1, q_1\}$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$) o ($nu_1 < min\ \{c_1, q_1\}$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$))) *hacer*
      4.1. Secuenciar y programar la tarea $i \in CTC_2$ con el tiempo de proceso más grande y cumpla que ($nu_1 \geq c_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$) o que ($nu_1 < c_1$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_1 \cdot (c_1 + 1) \geq Ta + tt_1 + t_{2i}$), entonces: $Ta = Ta + t_{2i};\ tr_2 = tr_2 - t_{2i};\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      4.2. ***Si*** ($tr_2 = 0$) *entonces*
      $nu_2 = nu_2 + 1;\ tr_2 = tt_2;\ x_2 = 0;\ MT[2][c_2] = MT[2][c_2] + 1;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$; ir al Paso 5.
      Fin Mientras
   5. ***Si*** ((($nu_1 < q_1$ y/o $nu_2 < q_2$) y $nu_1 \geq min\ \{c_1, q_1\}$ y $nu_2 \geq min\ \{c_2, q_2\}$) o ($nu_1 < q_1$ y $nu_2 < q_2$ y $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tr_2$) o ($nu_1 < q_1$ y $nu_2 = q_2$ y $ct_1 \cdot c_1 < Ta + tt_1$) o ($nu_1 = q_1$ y $nu_2 < q_2$ y $ct_2 \cdot c_2 < Ta + tr_2$)) *entonces*
      5.1. ***Si*** ($nu_1 = q_1$ y $nu_2 < q_2$) *entonces*
      $Id = Id - (ct_2 \cdot c_2 - Ta);\ Ta = ct_2 \cdot c_2;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      Si ($nu_1 < q_1$ y $nu_2 = q_2$) *entonces*
      $Id = Id - (ct_1 \cdot c_1 - Ta);\ Ta = ct_1 \cdot c_1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      5.2. ***Si*** ($nu_1 < q_1$ y $nu_2 < q_2$) *entonces*
      $Id = Id - (min\{ct_1 \cdot c_1, ct_2 \cdot c_2\} - Ta);\ Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\};$
      $c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      5.3. Ir al Paso 1.
      En caso contrario, ir al Paso 1.
   Fin Mientras.
6. Calcular los valores del buffer z_h y a_h (ver Figura 22 y Figura 23).
   Si ($Ta \leq lcm$) *entonces*
   El procedimiento finaliza con una solución factible.
   En caso contrario, el procedimiento finaliza sin solución factible.

Figura 32. Pseudocódigo de la heurística HB-2B.

Evaluación de una solución no factible

Como se ha mencionado, es posible partir de soluciones no factibles. Si la solución en curso S_C no es factible, entonces esta se evalúa de la siguiente forma (punto 6 y 13 de la Figura 30): $E(S_C) = E(S_C) + incumplimiento \cdot GN$; donde $incumplimiento = tiempo\ final\ del\ proceso\ de\ secuencición\ de\ las\ unidades - lcm$ (es decir, el tiempo que la programación de tareas excede a lcm) y $GN \in \mathbb{Z}^+$ es un número conceptualmente grande (a fijar/calibrar posteriormente).

Construcción de la lista de prioridad (PL) de las tareas

La PL (punto 7 del algoritmo de la Figura 30) es una lista que incluye todas las tareas de todas las unidades de todas las líneas, y donde cada tarea tiene asignado un valor de prioridad φ (ver Figura 33). El valor de prioridad φ se utiliza para ordenar las tareas del conjunto de tareas candidatas (CTC) de manera descendente a su valor φ y, posteriormente, seleccionar la primera de la lista para ser programada.

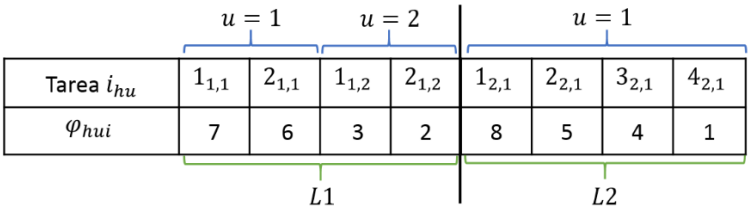

Figura 33. Ejemplo de una lista de prioridad (PL).

La PL es de tamaño N, donde $N = q_1 \cdot card(T_1) + q_2 \cdot card(T_2)$, siendo T_h el conjunto de tareas asignadas a la estación multilínea del producto de la línea h ($h \in H$) y q_h el número de unidades a procesar en la línea h ($h \in H$) en el tiempo lcm (o superciclo). Cada posición en la PL representa una tarea $i_{hu} \in T_h$ de una determinada unidad u ($u = 1, \dots, q_h$) perteneciente a la línea h ($h \in H$), y el valor de la posición de la lista representa el valor de prioridad φ_{hui} de dicha tarea i_{hu}. Siendo el rango del valor de prioridad $\varphi = N, \dots, 1$.

Para la construcción inicial de la PL, a cada tarea se le asignará un valor de prioridad φ siguiendo la secuencia de las tareas de L1 y L2 obtenida en la construcción de la solución S_0. La asignación de los valores de prioridad se realiza de la siguiente manera: la primera tarea en ser secuenciada (ya sea de L1 o L2) se le asigna el valor de prioridad más grande $\varphi_{hui} = N$, a la segunda tarea en ser secuenciada se le asigna el valor de prioridad $\varphi_{hui} = N - 1$, y así sucesivamente, hasta que la última tarea en ser secuenciada tiene un valor de prioridad $\varphi_{hui} = 1$.

En la Figura 33 se muestra un ejemplo de la estructura de una PL. Dicha PL está formada por 2 unidades en L1 ($q_1 = 2$) y 1 unidad en L2 ($q_2 = 1$). Las unidades de L1

están formadas por 2 tareas ($card(T_1) = 2$) y no existen precedencias entre ellas, mientras que la unidad de L2 está formada por 4 tareas ($card(T_2) = 4$) y tampoco existen precedencias entre ellas; por lo tanto, la longitud de la lista N será: $N = q_1 \cdot card(T_1) + q_2 \cdot card(T_2) = 2 \cdot 2 + 1 \cdot 4 = 8$; y el valor de prioridad irá de $\varphi = 8, \dots, 1$. Como se ha mencionado anteriormente, cada posición de la PL representa una tarea; en este ejemplo, las tareas pertenecientes a las unidades de la línea L1 ocupan las posiciones: 1 hasta $q_1 \cdot card(T_1) = 2 \cdot 2 = 4$, y las tareas pertenecientes a las unidades de la línea L2 ocupan las posiciones desde $q_1 \cdot card(T_1) + 1 = 2 \cdot 2 + 1 = 5$ hasta $q_1 \cdot card(T_1) + q_2 \cdot card(T_2) = 2 \cdot 2 + 1 \cdot 4 = 8$. Por ejemplo, la posición 6 en la PL es ocupada por la tarea 2 de la unidad 1 de la línea 2 ($i_{hu} \rightarrow 2_{2,1}$), con un valor de prioridad $\varphi_{212} = 5$.

A continuación, se muestra el procedimiento (HB-S) desarrollado para la construcción de una solución, donde dada una lista PL, las tareas se secuencian de acuerdo con su valor de prioridad φ en dicha lista. A diferencia de las heurísticas HB-1A/1B y HB-2A/2B, donde siempre se empieza por secuenciar primero las tareas de la unidad de la línea 1; en HB-S, es posible empezar secuenciando las tareas pertenecientes a la unidad(es) de línea 2.

Adicionalmente a las premisas asumidas en la definición del (sub)problema de dimensionado de buffer (Subsección 2.2.2), en la heurística HB-S se asume la siguiente:

- El tiempo requerido para procesar todas las unidades de ambas líneas, puede superar el tiempo lcm del sistema.

En la presente heurística, las variables y los parámetros utilizados son los mismos que los utilizados en la heurística HB-1A (Subsección 4.3.1), pero, en este caso, se modifican las siguientes variables:

Variables:

x_h 1, si hay una unidad empezada en la línea h en el ciclo c_h ($h \in H$); de lo contrario 0.

CTC Conjunto de tareas candidatas que puede estar formado por tareas de la unidad $nu_h + 1$ y $nu_j + 1$, pertenecientes a las líneas $(h, j) \in H$.

A continuación, se muestra los diferentes pasos del procedimiento HB-S. Estos pasos se deben realizar de forma iterativa mientras haya unidades por realizar en L1 y/o L2. Cuando se han realizado todas las unidades de las líneas L1 y L2 el procedimiento

finaliza, y se obtiene la duración de la programación de las tareas, una solución S_N y el valor del buffer a_h de la línea h.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_1 = 0$; $x_2 = 0$; $tr_1 = tt_1$; $tr_2 = tt_2$; $MT[h][x] = 0$ ($h \in H$; $x = 1, \dots, [max\{q_1, q_2\}]$).

1. Se comprueba si se cumple la siguiente condición:
   - Hay unidades por realizar en L1 o L2, es decir: $nu_1 < q_1$ o $nu_2 < q_2$.

   Si se cumple la condición, se irá al Paso 2; de lo contrario se irá al Paso 4.

2. Se crea el conjunto de tareas candidatas CTC. Una tarea i es candidata cuando cumple las siguientes condiciones:
   - La tarea i aún no ha sido programada.
   - Las predecesoras de la tarea i de la línea h de la unidad $nu_h + 1$ ya han sido programadas.
   - La suma del tiempo de las tareas restantes de la unidad $nu_h + 1$ de la línea h a la que pertenece la tarea i quepa en el ciclo actual de la línea h, es decir: $Ta + tr_h \leq ct_h \cdot c_h$.
   - Si hay una unidad empezada en otra línea j ($x_j = 1$) en el ciclo c_j, y: 1) hay una unidad empezada en la línea h ($x_h = 1$), debe haber tiempo suficiente en el ciclo c_j para realizar la tarea i de la línea h de la unidad $nu_h + 1$ y terminar las tareas que faltan de la unidad $nu_j + 1$ empezada en la línea j, esto es: $Ta + t_{hi} + tr_j \leq ct_j \cdot c_j$; o 2) no hay una unidad empezada en la línea h ($x_h = 0$), entonces, la suma del tiempo de las tareas (tt_h) de la unidad $nu_h + 1$ de la línea h y el tiempo restante de las tareas (tr_j) de la unidad $nu_j + 1$ empezada en la línea j, deben de caber en el ciclo c_j, esto es: $Ta + tt_h + tr_j \leq ct_j \cdot c_j$.

   Si el conjunto de tareas candidatas CTC es vacío se irá al Paso 3; en caso contrario:

   2.1 Se secuencia la tarea i del CTC con el índice de prioridad φ (de la lista PL) más grande. Se actualizan los valores pertenecientes a la línea h de la tarea i seleccionada: $tr_h = tr_h - t_{hi}$; $Ta = Ta + t_{hi}$; $x_h = 1$; y el ciclo de la línea j, $c_j = \left\lceil \frac{Ta}{ct_j} + \mu \right\rceil$.

   2.2 Si se han realizado todas las tareas de la unidad $nu_h + 1$ de la línea h ($tr_h = 0$), entonces se actualizan las variables referentes a la línea h: $nu_h = nu_h + 1$; $tr_h = tt_h$; $x_h = 0$; $MT[h][c_h] = MT[h][c_h] + 1$; $c_h = \left\lceil \frac{Ta}{ct_h} + \mu \right\rceil$; y se va al Paso 3. De lo contrario se va al Paso 2.

3. Actualización del reloj Ta: cuando no haya ninguna unidad empezada en ninguna de las líneas ($x_1 = 0$ y $x_2 = 0$) y se cumpla uno de los siguientes Conjuntos de condiciones:

   Conjunto 1:

   - Hay unidades por realizar en una o ambas líneas, es decir: $nu_1 < q_1$ y/o $nu_2 < q_2$.
   - No hay unidades por recuperar en ninguna de las líneas: $nu_1 \geq min\{c_1, q_1\}$ y $nu_2 \geq min\{c_2, q_2\}$.

   Conjunto 2:

   - Hay unidades por realizar en ambas líneas: $nu_1 < q_1$ y $nu_2 < q_2$.
   - La suma del tiempo de las tareas de la unidad de la L1 (tt_1) y de la unidad de la L2 (tt_2), no cabe en el tiempo del ciclo c_h de sus respectivas líneas, es decir: $ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tt_2$.

   Conjunto 3:

   - Hay unidades por realizar solo en L1 ($nu_1 < q_1$ y $nu_2 = q_2$).
   - La suma del tiempo de todas las tareas de la unidad en L1 no cabe en el tiempo del ciclo c_1, es decir: $ct_1 \cdot c_1 < Ta + tt_1$.

   Conjunto 4:

   - Hay unidades por realizar solo en L2 ($nu_1 = q_1$ y $nu_2 < q_2$).
   - La suma del tiempo de todas las tareas de la unidad en L2 no cabe en el tiempo del ciclo c_2, es decir: $ct_2 \cdot c_2 < Ta + tt_2$.

   Si no se cumple ninguno de los conjuntos de condiciones anteriores, ir al Paso 1.

   La operativa para la actualización del reloj Ta consiste en:

   3.1. Si se cumple uno de los Conjuntos y hay unidades por realizar solo en una de las líneas, es decir: $nu_1 < q_1$ o $nu_2 < q_2$, entonces, se actualizará el tiempo de reloj Ta al tiempo de ciclo de la línea h donde todavía quedan unidades por realizar: $Ta = ct_h \cdot c_h$; se actualiza el ciclo c_h de la línea h.

   3.2. Si se cumple uno de los Conjuntos y hay unidades por realizar en ambas líneas, es decir: $nu_1 < q_1$ y $nu_2 < q_2$, entonces, se actualizará el tiempo de reloj Ta al tiempo de ciclo de la línea h más cercano entre L1 y L2: $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\}$; se actualizarán los ciclos c_h de las líneas.

   3.3. Ir al Paso 1.

4. En este punto se termina el procedimiento heurístico:

   - Se han realizado todas las unidades en L1 y L2: $nu_1 = q_1$ y $nu_2 = q_2$. Se realiza el cálculo de los valores z_h y a_h (ver Figura 22 y Figura 23) y el procedimiento finaliza con una solución S_N, que puede ser factible (si el

tiempo de la programación no supera el tiempo lcm) o no factible (si el tiempo de la programación supera el tiempo lcm).

A continuación en la Figura 34 se muestra el pseudocódigo del procedimiento HB-S:

Inicialización: $Ta = 0;\ nu_1 = 0;\ nu_2 = 0\ ;\ c_1 = 1;\ c_2 = 1; x_1 = 0;\ x_2 = 0; tr_1 = tt_1;\ tr_2 = tt_2;$ $MT[h][x] = 0\ \left(h \in H;\ x = 1, \dots, \left[max\{q_1, q_2\}\right]\right)$.

1. ***Mientras*** ($nu_1 < q_1$ o $nu_2 < q_2$) *hacer*
   2. Crear conjunto de tareas candidatas (CTC). Una tarea i es candidata si:
      - La tarea i aún no ha sido programada.
      - Las predecesoras de la tarea i de la línea h de la unidad $nu_h + 1$ ya han sido programadas.
      - El tiempo restante de las tareas de la unidad $nu_h + 1$ de la línea h a la que pertenece la tarea i cabe en la línea h; esto es: $Ta + tr_h \leq ct_h \cdot c_h$
      - Si hay otra unidad empezada en otra línea j ($x_j = 1$) en el ciclo c_j, y: 1) hay una unidad empezada en la línea h ($x_h = 1$), debe haber tiempo suficiente en el ciclo c_j para realizar la tarea i de la unidad $nu_h + 1$ de la línea h y terminar las tareas que faltan de la unidad $nu_j + 1$ empezada en la línea j, esto es: $Ta + t_{hi} + tr_j \leq ct_j \cdot c_j$; o 2) no hay una unidad empezada en la línea h ($x_h = 0$), entonces, la suma del tiempo de las tareas (tt_h) de la unidad $nu_h + 1$ de la línea h y el tiempo restante de las tareas (tr_j) de la unidad $nu_j + 1$ empezada en la línea j, deben caber en el ciclo c_j, esto es: $Ta + tt_h + tr_j \leq ct_j \cdot c_j$.

   2.1 ***Si*** ($CTC = \emptyset$) *entonces*
   Ir al Paso 3;
   En caso contrario, secuenciar la tarea $i \in CTC$ con el valor de prioridad φ (de la lista PL) más grande, y actualizar los valores pertenecientes a la línea h de la tarea i: $tr_h = tr_h - t_{hi}$; $Ta = Ta + t_{hi}$; $x_h = 1$; y el ciclo de la línea j, $c_j = \left\lceil \frac{Ta}{ct_j} + \mu \right\rceil$;

   2.2 ***Si*** ($tr_h = 0$) *entonces*
   Actualizar los valores referentes a la línea h: $nu_h = nu_h + 1;\ tr_h = tt_h;\ x_h = 0;$ $MT[h][c_h] = MT[h][c_h] + 1;\ c_h = \left\lceil \frac{Ta}{ct_h} + \mu \right\rceil$; ir al Paso 3;
   En caso contrario, ir al Paso 2.

   3. ***Si*** (($x_1 = 0$ y $x_2 = 0$) y ((($nu_1 < q_1$ y/o $nu_2 < q_2$) y ($nu_1 \geq min\{c_1, q_1\}$ y $nu_2 \geq min\{c_2, q_2\}$)) o (($nu_1 < q_1$ y $nu_2 < q_2$) y ($ct_1 \cdot c_1 < Ta + tt_1$ y $ct_2 \cdot c_2 < Ta + tt_2$)) o (($nu_1 < q_1$ y $nu_2 = q_2$) y $ct_1 \cdot c_1 < Ta + tt_1$) o (($nu_1 = q_1$ y $nu_2 < q_2$) y $ct_2 \cdot c_2 < Ta + tt_2$))) *entonces*
      3.1. ***Si*** ($nu_1 = q_1$ y $nu_2 < q_2$) *entonces*
      $Ta = ct_2 \cdot c_2;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      Si ($nu_1 < q_1$ y $nu_2 = q_2$) *entonces*
      $Ta = ct_1 \cdot c_1;\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil$.
      3.2. ***Si*** ($nu_1 < q_1$ y $nu_2 < q_2$) *entonces*
      $Ta = min\{ct_1 \cdot c_1, ct_2 \cdot c_2\};\ c_1 = \left\lceil \frac{Ta}{ct_1} + \mu \right\rceil;\ c_2 = \left\lceil \frac{Ta}{ct_2} + \mu \right\rceil$.
      3.3. Ir al Paso 1.
      En caso contrario, ir al Paso 1.
   Fin Mientras
4. Calcular los valores del buffer z_h y a_h (ver Figura 22 y Figura 23).
   Si ($Ta \leq lcm$) *entonces*
   El procedimiento finaliza con una solución factible.
   En caso contrario, el procedimiento finaliza sin solución factible.

Figura 34. Pseudocódigo del procedimiento HB-S.

A continuación, se presentan la resolución de un ejemplo para ilustrar el funcionamiento del procedimiento HB-S. El ejemplo está basado en los grafos de precedencias de la Figura 35 y en los valores de prioridad de las tareas de la tabla PL (Figura 33).

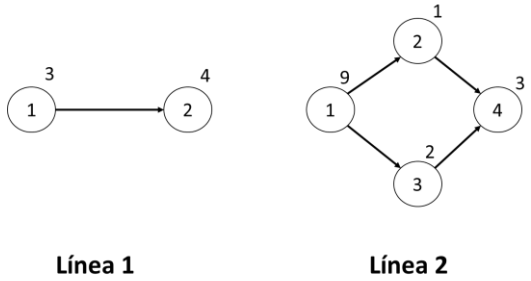

Figura 35. Grafos de precedencias de las tareas asignadas a la línea 1 y línea 2.

Datos:

$ct_1 = 15$; $ct_2 = 30$; $lcm = 30$;

$q_1 = 30/15 = 2$; $q_2 = 30/30 = 1$;

$T_1 = \{1,2\}$; $T_2 = \{1,2,3,4\}$;

$t_{11} = 3$; $t_{12} = 4$; $t_{21} = 9$; $t_{22} = 1$; $t_{23} = 2$; $t_{24} = 3$;

$tt_1 = (3 + 4) = 7$; $tt_2 = (9 + 1 + 2 + 3) = 15$;

$\mu = 10^{-6}$.

Inicialización de variables: $Ta = 0$; $nu_1 = 0$; $nu_2 = 0$; $c_1 = 1$; $c_2 = 1$; $x_1 = 0$; $x_2 = 0$; $tr_1 = 7$; $tr_2 = 15$; $MT[h][x] = 0$ ($h \in H$; $x = 1, \dots, [max\{2,1\}]$).

1. Paso 1. Se comprueba si el procedimiento cumple la condición. Como hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 0 < 2$; $nu_2 < q_2 \rightarrow 0 < 1$) se va al Paso 2.
2. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 1 de la unidad $nu_1 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; no tiene predecesoras; el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 0 + 7 \leq 15 \cdot 1$). La tarea 2 de la unidad $nu_1 + 1$ (0+1=1) de L1 no es candidata porque: su predecesora (tarea 1) aún no han sido programada. En L2, la tarea 1 de la unidad $nu_2 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; no tiene predecesoras; el tiempo de las tareas restantes de la unidad $nu_2 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 0 + 15 \leq 30 \cdot 1$). Las tareas 2, 3 y 4 de la unidad $nu_2 + 1$ (0+1=1) de L2 no son candidatas porque su predecesora (tarea 1) no han sido programada. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{1_{1,1}, 1_{2,1}\}$; como CTC no está vacío se va al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su

valor de prioridad (φ_{hui}), $CTC = \{8_{2,1,1}, 7_{1,1,1}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 1 de la unidad 1 de la línea 2 ($1_{2,1}$): entre los instantes 0 y 9; y se actualizan los valores: $tr_2 = 15 - 9 = 6$; $Ta = 0 + 9 = 9$; $x_2 = 1$; $c_1 = \left\lceil \frac{9}{15} + 10^{-6} \right\rceil = 1$. Como no se cumple la condición del Paso 2.2, se vuelve al Paso 2.

3. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 1 de la unidad $nu_1 + 1$ (0+1=1) no es una tarea candidata porque: el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (0+1=1) no cabe en el ciclo actual: $Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 9 + 7 \nleq 15 \cdot 1$. La tarea 2 de la unidad $nu_1 + 1$ (0+1=1) de L1 no es candidata porque: su predecesora (tarea 1) aún no han sido programada. En L2 la tarea 2 de la unidad $nu_2 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; su predecesora ya ha sido programada (tarea 1); el tiempo de las tareas restantes de la unidad $nu_2 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 9 + 6 \leq 30 \cdot 1$). En L2 la tarea 3 de la unidad $nu_2 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; su predecesora ya ha sido programada (tarea 1); el tiempo de las tareas restantes de la unidad $nu_2 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 9 + 6 \leq 30 \cdot 1$). La tarea 4 de la unidad $nu_2 + 1$ (0+1=1) de L2 no es candidata porque sus predecesoras (tarea 2 y 3) no han sido programadas. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{2_{2,1}, 3_{2,1}\}$; como CTC no esta vacío se pasa al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{5_{2,1,2}, 4_{2,1,3}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 2 de la unidad 1 de la línea 2 ($2_{2,1}$): entre los instantes 9 y 10; y se actualizan los valores: $tr_2 = 6 - 1 = 5$; $Ta = 9 + 1 = 10$; $x_2 = 1$; $c_1 = \left\lceil \frac{10}{15} + 10^{-6} \right\rceil = 1$. Como no se cumple la condición del Paso 2.2, se vuelve al Paso 2.
4. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 1 de la unidad $nu_1 + 1$ (0+1=1) no es una tarea candidata porque: el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (0+1=1) no cabe en el ciclo actual: $Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 10 + 7 \nleq 15 \cdot 1$. La tarea 2 de la unidad $nu_1 + 1$ (0+1=1) de L1 no es candidata porque su predecesora (tarea 1) aún no han sido programada. En L2 la tarea 3 de la unidad $nu_2 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; su predecesora ya ha sido programada (tarea 1); el tiempo de las tareas restantes de la unidad $nu_2 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 10 + 5 \leq 30 \cdot 1$). La tarea 4 de la unidad $nu_2 + 1$ (0+1=1) de L2 no es candidata porque su predecesora (tarea 3) no han sido programada. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC =$

$\{3_{2,1}\}$; como CTC no está vacío se pasa al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{4_{2,1,3}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 3 de la unidad 1 de la línea 2 ($3_{2,1}$): entre los instantes 10 y 12; y se actualizan los valores: $tr_2 = 5 - 2 = 3$; $Ta = 10 + 2 = 12$; $x_2 = 1$; $c_1 = \left\lceil \frac{12}{15} + 10^{-6} \right\rceil = 1$. Como no se cumple la condición del Paso 2.2, se vuelve al Paso 2.

5. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 1 de la unidad $nu_1 + 1$ (0+1=1) no es una tarea candidata porque: el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (0+1=1) no cabe en el ciclo actual: $Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 12 + 7 \not\leq 15 \cdot 1$. La tarea 2 de la unidad $nu_1 + 1$ (0+1=1) de L1 no es candidata porque su predecesora (tarea 1) aún no han sido programada. En L2 la tarea 4 de la unidad $nu_2 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; sus predecesoras ya han sido programadas (tareas 1, 2 y 3); el tiempo de las tareas restantes de la unidad $nu_2 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_2 \leq ct_2 \cdot c_2 \rightarrow 12 + 5 \leq 30 \cdot 1$). Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{4_{2,1}\}$; como CTC no está vacío se va al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{1_{2,1,4}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 4 de la unidad 1 de la línea 2 ($4_{2,1}$): entre los instantes 12 y 15; y se actualizan los valores: $tr_2 = 3 - 3 = 0$; $Ta = 12 + 3 = 15$; $x_2 = 1$; $c_1 = \left\lceil \frac{15}{15} + 10^{-6} \right\rceil = 2$. Como se cumple la condición del Paso 2.2 ($tr_2 = 0$), se actualizan los valores: $nu_2 = 0 + 1 = 1$; $tr_2 = 15$; $x_2 = 0$; $MT[2][1] = 0 + 1 = 1$; $c_2 = \left\lceil \frac{15}{30} + 10^{-6} \right\rceil = 1$; y se va al Paso 3.
6. Paso 3. No se actualiza el tiempo de reloj Ta. No hay ninguna unidad empezada en ninguna línea ($x_1 = 0$ y $x_2 = 0$), pero no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en la L1 ($nu_1 < q_1 \rightarrow 0 < 2$ y $nu_2 < q_2 \rightarrow 1 \not< 1$), pero hay unidades por recuperar en L1 ($nu_1 \geq mi\,n\{c_1, q_1\} \rightarrow 0 \not\geq min\{2,2\}$ y $nu_2 \geq mi\,n\{c_2, q_2\} \rightarrow 1 \geq min\{1,1\}$); conjunto 2: no hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 0 < 2$ y $nu_2 < q_2 \rightarrow 1 \not< 1$); conjunto 3: hay unidades por realizar solo en L1 ($nu_1 < q_1 \rightarrow 0 < 2$ y $nu_2 = q_2 \rightarrow 1 = 1$), pero la suma del tiempo de las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1 sí que caben en su ciclo c_1 ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 15 \cdot 2 \not< 15 + 7$); conjunto 4: no hay unidades por realizar en L2 ($nu_2 < q_2 \rightarrow 1 \not< 1$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
7. Paso 1. Se comprueba si el procedimiento cumple la condición. Como hay unidades por realizar en L1 ($nu_1 < q_1 \rightarrow 0 < 2$; $nu_2 < q_2 \rightarrow 1 \not< 1$) se va al Paso 2.

8. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 1 de la unidad $nu_1 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; no tiene predecesoras; el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 15 + 7 \leq 15 \cdot 2$). La tarea 2 de la unidad $nu_1 + 1$ (0+1=1) de L1 no es candidata porque: su predecesora (tarea 1) aún no han sido programada. En L2 no hay unidades por realizar. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{1_{1,1}\}$; como CTC no está vacío se va al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{7_{1,1,1}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 1 de la unidad 1 de la línea 1 ($1_{1,1}$): entre los instantes 15 y 18; y se actualizan los valores: $tr_1 = 7 - 3 = 4$; $Ta = 15 + 3 = 18$; $x_1 = 1$; $c_2 = \left\lceil \frac{18}{30} + 10^{-6} \right\rceil = 1$. Como no se cumple la condición del Paso 2.2, se vuelve al Paso 2.
9. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 2 de la unidad $nu_1 + 1$ (0+1=1) es candidata porque: la tarea aún no ha sido programada; su predecesora ya ha sido programada (tarea 1); el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (0+1=1) caben en el ciclo actual ($Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 15 + 4 \leq 15 \cdot 2$). En L2 no hay unidades por realizar. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{2_{1,1}\}$; como CTC no está vacío se va al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{6_{1,1,2}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 2 de la unidad 1 de la línea 1 ($2_{1,1}$): entre los instantes 18 y 22; y se actualizan los valores: $tr_1 = 4 - 4 = 0$; $Ta = 18 + 4 = 22$; $x_1 = 1$; $c_2 = \left\lceil \frac{22}{30} + 10^{-6} \right\rceil = 1$. Como se cumple la condición del Paso 2.2 ($tr_1 = 0$), se actualizan los valores: $nu_1 = 0 + 1 = 1$; $tr_1 = 7$; $x_1 = 0$; $MT[1][2] = 0 + 1 = 1$; $c_1 = \left\lceil \frac{22}{15} + 10^{-6} \right\rceil = 2$; y se va al Paso 3.
10. Paso 3. No se actualiza el tiempo de reloj Ta. No hay ninguna unidad empezada en ninguna línea ($x_1 = 0$ y $x_2 = 0$) pero no se cumple ninguno de los cuatro conjuntos de condiciones: conjunto 1: hay unidades por realizar en la L1 ($nu_1 < q_1 \rightarrow 1 < 2$ y $nu_2 < q_2 \rightarrow 1 \not< 1$), pero hay unidades por recuperar en L1 ($nu_1 \geq min\{c_1, q_1\} \rightarrow 1 \not\geq min\{2,2\}$ y $nu_2 \geq min\{c_2, q_2\} \rightarrow 1 \geq min\{1,1\}$); conjunto 2: no hay unidades por realizar en ambas líneas ($nu_1 < q_1 \rightarrow 1 < 2$ y $nu_2 < q_2 \rightarrow 1 \not< 1$); conjunto 3: hay unidades por realizar solo en L1 ($nu_1 < q_1 \rightarrow 1 < 2$ y $nu_2 = q_2 \rightarrow 1 = 1$), pero la suma del tiempo de las tareas de la unidad $nu_1 + 1$ (0+1=1) de L1 sí que caben en su ciclo c_1 ($ct_1 \cdot c_1 < Ta + tt_1 \rightarrow 15 \cdot 2 \not< 22 + 7$); conjunto 4: no hay unidades por

realizar en L2 ($nu_2 < q_2 \rightarrow 1 \not< 1$); como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.

11. Paso 1. Se comprueba si el procedimiento cumple la condición. Como hay unidades por realizar en la L1 ($nu_1 < q_1 \rightarrow 1 < 2;\ nu_2 < q_2 \rightarrow 1 \not< 1$) se va al Paso 2.
12. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 1 de la unidad $nu_1 + 1$ (1+1=2) es candidata porque: la tarea aún no ha sido programada; no tiene predecesoras; el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (1+1=2) caben en el ciclo actual ($Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 22 + 7 \leq 15 \cdot 2$). La tarea 2 de la unidad $nu_1 + 1$ (1+1=2) de L1 no es candidata porque: su predecesora (tarea 1) aún no han sido programada. En L2 no hay unidades por realizar. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{1_{1,2}\}$; como CTC no está vacío se pasa al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{3_{1,2,1}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 1 de la unidad 2 de la línea 1 ($1_{1,2}$): entre los instantes 22 y 25; y se actualizan los valores: $tr_1 = 7 - 3 = 4$; $Ta = 22 + 3 = 25$; $x_1 = 1$; $c_2 = \left\lceil \frac{25}{30} + 10^{-6} \right\rceil = 1$. Como no se cumple la condición del Paso 2.2, se vuelve al Paso 2.
13. Paso 2. Se crea el conjunto de tareas candidatas CTC. En L1, la tarea 2 de la unidad $nu_1 + 1$ (1+1=2) es candidata porque: la tarea aún no ha sido programada; su predecesora ya ha sido programada (tarea 1); el tiempo de las tareas restantes de la unidad $nu_1 + 1$ (1+1=2) caben en el ciclo actual ($Ta + tr_1 \leq ct_1 \cdot c_1 \rightarrow 25 + 4 \leq 15 \cdot 2$). En L2 no hay unidades por realizar. Por lo tanto $CTC = \{i_{h,u}\} \rightarrow CTC = \{2_{1,2}\}$; como CTC no está vacío se va al Paso 2.1. En Paso 2.1 las tareas de CTC se ordenan de manera descendente de acuerdo con su valor de prioridad (φ_{hui}), $CTC = \{2_{1,2,2}\}$, y se secuencia la tarea con el índice de prioridad φ más grande, es decir la tarea 2 de la unidad 2 de la línea 1 ($2_{1,2}$): entre los instantes 25 y 24; y se actualizan los valores: $tr_1 = 4 - 4 = 0$; $Ta = 25 + 4 = 29$; $x_1 = 1$; $c_2 = \left\lceil \frac{29}{30} + 10^{-6} \right\rceil = 1$. Como se cumple la condición del Paso 2.2 ($tr_1 = 0$), se actualizan los valores: $nu_1 = 1 + 1 = 2$; $tr_1 = 7$; $x_1 = 0$; $MT[1][2] = 1 + 1 = 2$; $c_1 = \left\lceil \frac{30}{15} + 10^{-6} \right\rceil = 3$; y se va al Paso 3.
14. Paso 3. No se actualiza el tiempo de reloj Ta, ya que se han realizado todas las unidades en todas las líneas. Como no se cumple ninguna de las condiciones, entonces se vuelve al Paso 1.
15. Paso 1. Como ya no hay unidades por realizado en ninguna de las líneas ($nu_1 < q_1 \rightarrow 2 \not< 2;\ nu_2 < q_2 \rightarrow 1 \not< 1$) se va al Paso 4.

16. Paso 4. Como se han realizado todas las unidades en ambas líneas, se calculan los valores a_h y z_h y el procedimiento finaliza con una solución factible.

De los resultados obtenidos $a_h = \{1,0\}$ y $z_h = \{1,0\}$, en la línea L1 se debe de disponer de un buffer de tamaño 1 unidad ($a_1 = 1$). Como se puede observar en la Figura 36, habrá una pieza terminada en el buffer de L1 en el ciclo 1. Se puede observar que la unidad $u2$ de L1 será la unidad procesada $u02$ del siguiente superciclo. Por otra parte, en la línea L2 no se necesita de un buffer.

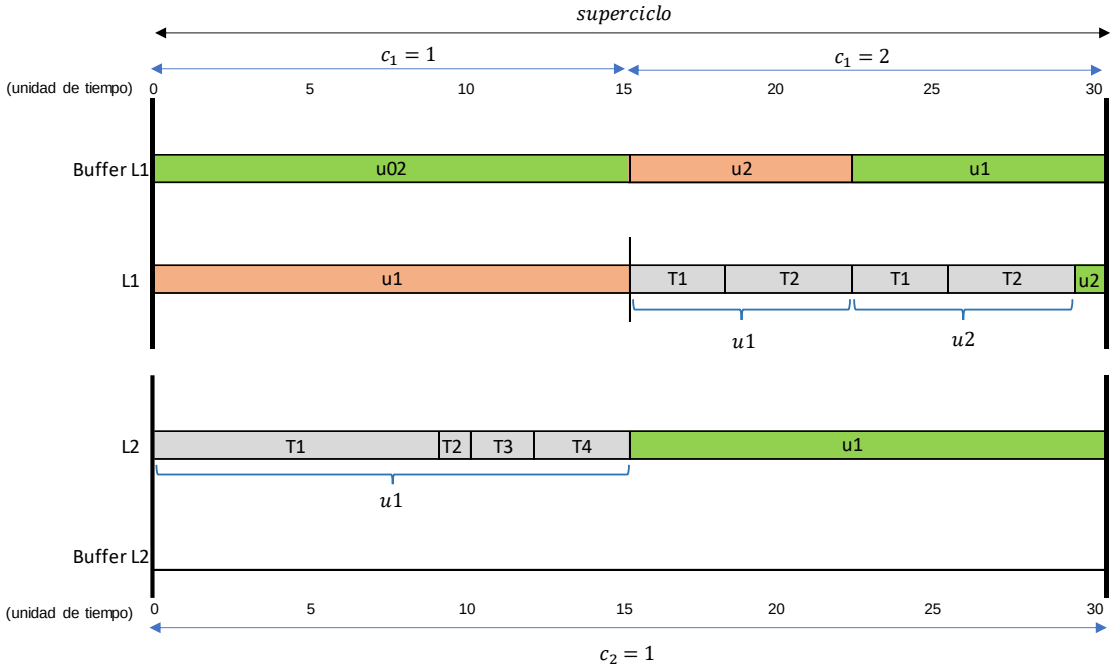

Figura 36. Representación de la solución obtenida para el ejemplo al utilizar el procedimiento HB-S.

Generación de una solución vecina

La generación de una solución vecina S_N perteneciente a $N(S_C)$ se realiza en dos pasos: 1) cambiar (*swap*) el valor de prioridad φ de dos tareas de la PL seleccionadas al azar, de esta forma se crea una nueva PL; y 2) construir una solución vecina S_N al utilizar los valores de prioridad φ de la nueva PL y el procedimiento HB-S.

A continuación, se muestra un ejemplo de cómo se crea una nueva PL (Figura 37) a partir de una PL inicial (Figura 33), al realizar un movimiento *swap*. Dos tareas de la PL representada en la Figura 33 son seleccionadas al azar, por ejemplo, la tarea $1_{1,1}$ y la tarea $4_{2,1}$, e intercambian sus valores de prioridad, generando una nueva PL (Figura 37).

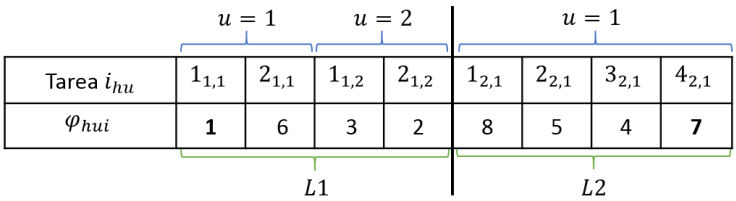

Figura 37. Ejemplo de movimiento *swap* de los valores de prioridad de la tarea 1 de la unidad 1 de la línea 1 y la tarea 4 de la unidad 1 de la línea 2, respecto a la Figura 33.

Reducción de la temperatura

La temperatura t del algoritmo HB-SA influye en la probabilidad de aceptar peores soluciones vecinas. Cada número determinado de iteraciones itt la temperatura t se irá reduciendo gradualmente. Una de las formas más habituales en la literatura para realizar esta reducción es la reducción geométrica, es decir, $A(t) = \alpha \cdot t$, donde α es el factor de enfriamiento: $0 < \alpha < 1$ (Dowsland and Adenso-Díaz, 2003; Gendreau and Potvin, 2010). El valor del parámetro α, así como la temperatura inicial t_0, la temperatura mínima t_{min} y el número de iteraciones donde la temperatura permanece igual itt, se convierten en parámetros del algoritmo, los cuales deben ser calibrados.

Función objetivo

La función objetivo del problema es minimizar la suma del número de unidades tratadas y no tratadas de producto que hay en el buffer de la línea h ($h \in H$), por tanto su tamaño en un superciclo: F.Obj. [MIN] $E(S) = \sum_{h \in H} a_h$.

Criterio de parada

El HB-SA propuesto finaliza cuando se cumple al menos uno de los tres siguientes criterios de parada: 1) cuando una solución es factible y su valor $E(S)$ es igual a 0 (se ha obtenido una solución óptima); 2) cuando se alcanza la temperatura mínima t_{min}; o 3) cuando se excede un tiempo de cálculo prestablecido max_time_{Bf}.

4.5.3 Experimento computacional

En esta subsección se han realizado dos experimentos computacionales con el objetivo de evaluar el rendimiento del procedimiento propuesto.

Previo a la realización de los experimentos, en la Subsección 4.5.3.1 se realiza la calibración de los parámetros de la metaheurística HB-SA.

En el primer experimento computacional (Subsección 4.5.3.2) se evalúa el rendimiento de la metaheurística HB-SA y sus resultados se comparan con los obtenidos por los otros procedimientos presentados anteriormente. El segundo experimento (Subsección

4.5.3.3) se realiza un análisis de sensibilidad de la metaheurística, que tiene como objetivo estudiar cómo afecta a los resultados, el limitar el tiempo máximo de cálculo max_time_{Bf} del procedimiento HB-SA. Para este fin, el tiempo max_time_{Bf} se ha fijado a diferentes valores.

La metaheurística HB-SA desarrollada ha sido codificada y ejecutada en Java SE8, y testeada en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

4.5.3.1 Calibración de los parámetros del HB-SA

La calibración o ajuste de los parámetros de un algoritmo es en muchas ocasiones una labor difícil (García-Villoria and Pastor, 2009). El valor de los parámetros influye de manera importante en el comportamiento del algoritmo y por ende a los resultados obtenidos.

Los parámetros t_{min}, t_0, α, itt utilizados en HB-SA han sido calibrados utilizando el algoritmo Nelder and Mead (N&M) (Nelder and Mead, 1965). El algoritmo N&M ha sido aplicado en la literatura a la hora de calibrar los diversos parámetros de un algoritmo (ver García-Villoria et al., 2010; Corominas et al., 2013).

Para calibrar los parámetros de la metaheurística HB-SA se ha generado un conjunto de entrenamiento (del mismo modo que en la Subsección 4.2.4.1) formado por 40 ejemplares representativos a partir de los grafos de precedencias de la Figura 18 (10 ejemplares), Figura 19 (10 ejemplares), Figura 20 (10 ejemplares) y Figura 21 (10 ejemplares). Los ejemplares de entrenamiento son solucionados sin un tiempo de cálculo límite $max\_time_{Bf} = infinito$ y con el valor $GN = 10^5$. El tamaño de los ejemplares está comprendido entre 2 y 40 tareas.

Los valores obtenidos de la calibración para los parámetros de HB-SA son los siguientes: $t_{min} = 0.004$; $t_0 = 161$; $\alpha = 0.914$; $itt = 89$.

4.5.3.2 Experimento computacional 1

En el experimento computacional 1 se ha realizado el mismo experimento que en la Subsección 4.3.3. Para este caso, se ha fijado el tiempo de cálculo límite a $max\_time_{Bf} = 3600$ segundos (mismo tiempo límite que en los modelos V1, V2 y V3) y el valor $GN = 10^5$. El experimento se ha resuelto 5 veces de manera independiente, y el promedio de los resultados y del tiempo de cálculo de cada ejemplar se recogen en el Anexo B.2.1 (documento externo “Anexos”). Los resultados obtenidos por HB-SA son comparados con los obtenidos por los otros procedimientos presentados.

En la Tabla 8 se presenta la misma información que en la Tabla 1, pero en este caso, para que una solución de HB-SA sea considerada óptima, se tiene que haber conseguido el óptimo las 5 veces que se ha ejecutado el experimento. De los resultados se observa que HB-SA ha obtenido una solución factible en todos los ejemplares del Conjunto 1, 2 y 4. Obteniendo 117 soluciones factibles (considerando las óptimas y las no óptimas), de las cuales 88 son óptimas. Sin embargo, en 3 ejemplares no ha sido capaz de encontrar una solución factible. HB-SA ha obtenido 25 soluciones factibles más que los modelos V1 y V3, 12 soluciones factibles más que el modelo V2, 5 más que HB-1A y 3 más que HB-1B y HB-1AB. En global, es el procedimiento que ha obtenido un mayor número de soluciones factibles.

Tabla 8. Resumen de los resultados obtenidos por los procedimientos V1, V2, V3, HB-1A, HB-1B, HB-1AB y HB-SA.

Procedimiento	Conjunto 1 (60)	Conjunto 2 (20)	Conjunto 3 (20)	Conjunto 4 (20)	Total (120)
V1	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
V2	41 / 6 / 13	15 / 4 / 1	18 / 1 / 1	19 / 1 / 0	93 / 12 / 15
V3	37 / 4 / 19	12 / 1 / 7	18 / 1 / 1	19 / 0 / 1	86 / 6 / 28
HB-1A	58 / 2 / 0	9 / 9 / 2	3 / 11 / 6	14 / 6 / 0	84 / 28 / 8
HB-1B	33 / 27 / 0	6 / 13 / 1	3 / 12 / 5	10 / 10 / 0	52 / 62 / 6
HB-1AB	58 / 2 / 0	9 / 10 / 1	4 / 11 / 5	14 / 6 / 0	85 / 29 / 6
HB-SA	58 / 2 / 0	10 / 10 / 0	6 / 11 / 3	14 / 6 / 0	88 / 29 / 3

Por otra parte, en la Tabla 9 se presenta la misma información que en Tabla 2, pero en este caso, para HB-SA el total del tamaño del buffer y del tiempo de cálculo, es el promedio de las 5 ejecuciones de cada ejemplar. Los resultados que se muestran en la tabla consideran los 84 ejemplares que son resueltos por todos los procedimientos.

En la Tabla 9 se puede observar que el procedimiento HB-SA ha obtenido un mejor promedio del tamaño del buffer que las heurísticas HB-1A, HB-1B y HB-1AB, y está bastante cerca de los modelos V1 y V3 (obteniendo un promedio un 8.9% peor). Sin embargo, V2 obtiene un promedio un 32% mejor que HB-SA. Cabe destacar que, en HB-SA, el tiempo promedio de cálculo (considerando solo los ejemplares resueltos por los 7 procedimientos) es de 2.55 segundos, siendo 53 segundos el tiempo máximo de cálculo registrado para resolver uno de los ejemplares.

Tabla 9. Promedio del tamaño de los buffers y del tiempo de cálculo obtenidos por los procedimientos V1, V2, V3, HB-1A, HB-1B, HB-1AB y HB-SA.

	V1	V2	V3	HB-1A	HB-1B	HB-1AB	HB-SA
Promedio tamaño buffers	1.060	0.786	1.060	1.321	2.488	1.321	1.155
Tiempo promedio (s)	195.14	144.4	196.73	0.0004	0.0004	0.0007	2.55

En resumen, se puede constatar que V2 es el procedimiento presentado que obtiene un mejor (menor) promedio total del tamaño del buffer. Obteniendo unos resultados del tamaño del buffer mejores que HB-SA. Sin embargo, HB-SA necesita un tiempo de cálculo muy pequeño, reduciendo el tiempo promedio de V2 en un 98%. Además, HB-SA ha obtenido una solución factible en 117 ejemplares (y una solución óptima en 88 ejemplares). De esta forma el HB-SA es el procedimiento que aporta un mayor equilibrio entre: número de soluciones factibles, calidad de las solución y tiempo de cálculo.

4.5.3.3 Experimento computacional 2

El segundo experimento computacional, tiene como objetivo estudiar cómo afecta a los resultados, el limitar el tiempo máximo de cálculo del procedimiento HB-SA. Para este fin, el tiempo máximo de cálculo max_time_{Bf} se ha fijado igual a diferentes valores: 0.05, 0.1, 0.5, 1, 2.5, 5, 10 y 20 segundos. La elección de estos valores viene dada por los tiempos CPU necesarios en el experimento computacional 1.

El experimento se ha resuelto 5 veces de manera independiente, y el promedio de los resultados y del tiempo de cálculo de cada ejemplar se recoge en el Anexo B.2.2 (véase en el documento externo “Anexos”).

En la Tabla 10 (que presenta la misma información que en la Tabla 8) se puede observar que el HB-SA obtiene el mismo número de soluciones no factibles para todos los valores de max_time_{Bf}. También, el HB-SA obtiene el mismo número de soluciones óptimas, excepto para los valores max_time_{Bf} de 0.05 a 2.5 segundos en el Conjunto 3, donde se obtiene una solución óptima menos para estos valores. En general, se observa que estos resultados son prácticamente los mismos que se han obtenido para el HB-SA ejecutándose con un tiempo límite de 3600 segundos (Tabla 8).

Tabla 10. Resumen de los resultados obtenidos por HB-SA al fijar el max_time_{Bf} a diferentes valores.

max_time_{Bf} (s)	Conjunto 1 (60)	Conjunto 2 (20)	Conjunto 3 (20)	Conjunto 4 (20)	Total (120)
0.05	58 / 2 / 0	10 / 10 / 0	5 / 12 / 3	14 / 6 / 0	87 / 30 / 3
0.1	58 / 2 / 0	10 / 10 / 0	5 / 12 / 3	14 / 6 / 0	87 / 30 / 3
0.5	58 / 2 / 0	10 / 10 / 0	5 / 12 / 3	14 / 6 / 0	87 / 30 / 3
1	58 / 2 / 0	10 / 10 / 0	5 / 12 / 3	14 / 6 / 0	87 / 30 / 3
2.5	58 / 2 / 0	10 / 10 / 0	5 / 12 / 3	14 / 6 / 0	87 / 30 / 3
5	58 / 2 / 0	10 / 10 / 0	6 / 11 / 3	14 / 6 / 0	88 / 29 / 3
10	58 / 2 / 0	10 / 10 / 0	6 / 11 / 3	14 / 6 / 0	88 / 29 / 3
20	58 / 2 / 0	10 / 10 / 0	6 / 11 / 3	14 / 6 / 0	88 / 29 / 3
3600	58 / 2 / 0	10 / 10 / 0	6 / 11 / 3	14 / 6 / 0	88 / 29 / 3

En la Tabla 11 se muestran los resultados obtenidos al fijar el tiempo máximo de cálculo max_time_{Bf} a diferentes valores, para los 117 ejemplares en los cuales una solución factible u óptima es lograda por los nueve valores de max_time_{Bf} utilizados. La primera fila de la tabla muestra los diferentes valores a los que se ha fijado el tiempo máximo de cálculo; en la segunda fila se muestra el promedio del tamaño de los buffers; en la tercera fila se muestra el incremento en porcentaje del promedio total de cada valor max_time_{Bf} respecto a la mejor solución (es decir, respecto a la solución de la columna "3600"); en la cuarta fila se muestra el promedio del total del tiempo de cálculo (en segundos). Cabe puntualizar que, el valor obtenido por HB-SA al ejecutarse con un tiempo $max\_time_{Bf} = 3600$ difiere del presentado anteriormente en la Tabla 9. Esto es debido a que en este experimento se consideran 117 ejemplares y no los 84 ejemplares resueltos por todos los procedimientos presentados, como en la Tabla 9.

Tabla 11. Promedio de los resultados obtenidos por HB-SA al fijar el max_time_{Bf} a diferentes valores.

max_time_{Bf} (s)	0.05	0.1	0.5	1	2.5	5	10	20	3600
Promedio del tamaño buffers	2.3	2.283	2.194	2.170	2.118	2.082	2.052	2.070	2.025
Diferencia (%)	+13.62	+12.73	+8.37	+7.18	+4.60	+2.84	+1.36	+2.22	-
Tiempo promedio (s)	0.027	0.053	0.260	0.491	1.070	1.691	2.508	3.360	6.917

En la Tabla 11 se puede observar que, como era de esperar, los resultados tienden a mejorar conforme se aumenta el tiempo máximo de cálculo. Cabe destacar que los valores de tiempo max_time_{Bf} que obtienen peores resultados, es debido a las peores soluciones obtenidas en unos pocos ejemplares. Estos ejemplares se localizan en el Conjunto 2 y 3. También se observa que con un tiempo $max\_time_{Bf} = 10$, se han obtenido mejores resultados que con $max\_time_{Bf} = 20$, esto es por la aleatoriedad del procedimiento HB-SA. Además, se observa que el segundo caso que obtiene un mejor promedio del total del tamaño de los buffers (HB-SA con un tiempo $max\_time_{Bf} = 10$), reduce en un 63% el tiempo de $max\_time_{Bf} = 3600$ (6.917 segundos). Incluso los casos con tiempos de max_time_{Bf} más pequeños, como $max\_time_{Bf} = 0.5$ o 1, reducen en un 96% y 92%, respectivamente, el tiempo empleado por $max\_time_{Bf} = 3600$, sin obtener unos resultados mucho peores.

Por otra parte, al ejecutar el HB-SA con un tiempo $max\_time_{Bf} = 3600$, se visualiza que no hay ningún ejemplar que se aproxime al tiempo máximo de cálculo, siendo 105 segundos el tiempo máximo de cálculo registrado para resolver un ejemplar.

En la Tabla 12 se presenta la misma información que en la Tabla 2, pero en este caso solo se consideran los resultados obtenidos por las heurísticas HB-1A, HB-1B, HB-1AB y la metaheurística HB-SA con los diferentes valores de max_time_{Bf}. Para la obtención de estos promedios se ha considerado el valor total del buffer (de ambas líneas) de, únicamente, los ejemplares para los cuales se ha encontrado una solución factible u óptima por los 4 procedimientos. En este experimento han sido 112 ejemplares.

Tabla 12. Promedio de los resultados obtenidos por HB-1A, HB-1B, HB-1AB y HB-SA al fijar el max_time_{Bf} a diferentes valores.

	HB-1A	HB-1B	HB-1AB	HB-SA								
				0.05	0.1	0.5	1	2.5	5	10	20	3600
Promedio del tamaño buffers	1.018	5.018	1.018	1	0.982	0.946	0.936	0.904	0.891	0.891	0.893	0.893
Tiempo promedio (s)	0.0007	0.0007	0.001	0.017	0.033	0.158	0.295	0.638	1.029	1.503	1.966	3.804

De los resultados de la Tabla 12 se observa que el HB-SA con $max\_time_{Bf} = 5\ y\ 10$ obtiene el mejor promedio respecto al tamaño de los buffers. Cabe destacar que el HB-SA con $max\_time_{Bf} = 3600$ no es el que obtiene el mejor promedio, como sí ocurría en la Tabla 11 (donde se consideran 117 ejemplares). Esto es debido a el que HB-SA con $max\_time_{Bf} = 3600$ (ahora considera 112 ejemplares, Tabla 12) obtiene un peor resultado en una instancia, respecto a HB-SA con $max\_time_{Bf} = 5\ y\ 10$.

También se puede observar que HB-SA con $max\_time_{Bf} = 0.05$ obtiene un valor promedio del tamaño de los buffers un 12.2% peor que HB-SA con $max\_time_{Bf} = 5$. Del mismo modo, HB-1A obtiene un resultado un 14.2% peor que HB-SA con $max\_time_{Bf} = 5$. Pero, en ambos casos (HB-SA con $max\_time_{Bf} = 0.05$ y HB-1A), con un tiempo de cálculo inferior. Por otro lado, HB-1A obtiene un valor promedio del tamaño de los buffers un 1.8% peor que HB-SA con $max\_time_{Bf} = 0.05$, pero con un menor tiempo de cálculo.

4.6 Conclusiones

A lo largo de este capítulo, se han presentado diferentes procedimientos de resolución exactos y no exactos para el (sub)problema de dimensionado de buffers en una estación multilínea, con el objetivo de minimizar el tamaño de los buffers (en ambas líneas) de la estación.

Los procedimientos aquí propuestos, secuencian y programan las tareas asignadas a una estación multilínea, cumpliendo con los supuestos y condiciones definidos en la Subsección 2.2.2.

Todos los procedimientos presentados han sido evaluados y comparados entre sí vía un experimento computacional con 120 ejemplares.

De los resultados obtenidos, se observa que lo modelos de PLEM no son capaces de encontrar, en algunos ejemplares, una solución factible dentro de un tiempo de cálculo límite. Siendo el modelo V2 (misma secuencia de tareas para todas las unidades), el que aporta un mayor número de soluciones factibles (considerando las soluciones óptimas y no óptimas). Además, obtiene un tiempo promedio de cálculo menor que los modelos V1 y V3.

De forma similar, los procedimientos heurísticos HB-1A y HB-1B, muestran que son capaces de obtener un mayor número de soluciones factibles que los modelos V1, V2 y V3. Obtienen un valor promedio de la solución cercano a los modelos V1 y V3, pero en un tiempo de cálculo muy inferior al conseguido por los modelos V1, V2 y V3. Por otra parte, el cóctel de heurísticas HB-1AB, obtiene un mayor número de soluciones factibles que lo demás procedimientos presentados, manteniendo un breve tiempo de cálculo.

La metaheurística HB-SA desarrollada, muestra ser la que mejor rendimiento presenta en términos de número de soluciones factibles obtenidas, calidad de las soluciones y tiempo de cálculo. La calidad de las soluciones obtenidas con HB-SA están muy cerca de las obtenidos por V1 y V3, y mejoran a las obtenidas por HB-1A, HB-1B y HB-1AB; pero son peores que las obtenidas por V2. Sin embargo, los tiempos de cálculo obtenidos por HB-SA son breves en comparación a los modelos V1, V2 y V3. A partir de otro experimento computacional realizado para analizar el rendimiento de la metaheurística HB-SA al variar el tiempo límite de cálculo, se observa que, con un tiempo de cálculo breve, las soluciones obtenidas son casi siempre factibles (117 soluciones factibles, de las cuales 88 son óptimas) y con una calidad no mucho peor que las obtenidas con un tiempo de cálculo elevado. Por tanto, no parece ser necesario utilizar un tiempo elevado de cálculo, ya que la calidad de las soluciones conseguidas por HB-SA con un tiempo límite breve son suficientemente buenas. En este punto, también cabe destacar que HB-1A obtiene buenas soluciones en un tiempo de cálculo inferior a HB-SA. Por lo tanto, es un procedimiento a considerar dependiendo del tiempo disponible para resolver el PALBP-B.

A raíz de lo anterior, se evidencia que la metaheurística HB-SA es el procedimiento que muestra un mejor equilibrio en términos número de soluciones factibles, calidad de las

soluciones y tiempo de cálculo. Además, se concluye que HB-SA obtiene buenas soluciones en un breve tiempo de cálculo, lo que lo hace aplicable en un entorno industrial realista. Se recuerda que el (sub)problema que se está resolviendo forma parte de un problema mayor (PALBP-B), por lo tanto, es deseable resolver el (sub)problema de una forma factible en un breve periodo de tiempo.

Cabe destacar que, como se ha evidenciado en los experimentos computacionales realizados, es posible que para un conjunto de tareas dado a una estación multilínea no exista una solución factible o los procedimientos desarrollados no la encuentren.

Para finalizar, se hace mención que los modelos de programación matemática V1, V2 y V3, presentados en este capítulo, han derivado en dos publicaciones. Y se está preparando una publicación con los procedimientos heurísticos y metaheurístico presentados.

La propuesta del modelo de PLEM V1 desarrollado en esta tesis ha derivado en una publicación: ***Existence and sizing of buffers in parallel assembly lines with multi-line workstations and different cycle times (Pastor et al., 2021).***

La propuesta del modelo de PLEM V2 y V3 desarrollados en esta tesis han derivado en una publicación: ***Mathematical models for buffer sizing problems in parallel assembly lines with multi-line stations and different cycle times (Aguilar et al., 2021).***

Se está preparando una publicación con los procedimientos heurísticos y metaheurístico presentados en este capítulo: ***Heuristic and metaheuristic procedures for buffer sizing problem in parallel assembly lines with multi-line stations and different cycle times (en preparación/enviado).***

Capítulo 5. Métodos de resolución del PALBP-B

En este capítulo se presentan diversos procedimientos no exactos para resolver conjuntamente el (sub)problema de equilibrado de líneas en paralelo con estaciones multilínea y el (sub)problema de dimensionado de buffers. En la Sección 5.1 se analizan los procedimientos desarrollados en la literatura del PALB para la obtención de una solución factible sin considerar el uso de buffers. Posteriormente, en la Sección 5.2, Sección 5.3, Sección 5.4 y Sección 5.5, se proponen procedimientos no exactos para la resolución del problema y en cada sección se realiza un experimento computacional. Las conclusiones del capítulo se presentan en la Sección 5.6.

5.1 Análisis de los procedimientos utilizados en la literatura del PALBP

En esta sección se analizan los trabajos de la literatura del problema de equilibrado de líneas en paralelo con estaciones multilínea sin considerar el uso de buffers. El objetivo es analizar las diferentes técnicas utilizadas en estos trabajos para la generación de una buena solución factible.

De los diferentes estudios presentados en la literatura del PALBP, únicamente se considerarán aquellos que tratan el problema con las siguientes características del problema aquí resuelto: líneas rectas, líneas de modelo único, tiempos de tareas deterministas, y líneas con tiempos de ciclo iguales o diferentes. Los estudios que consideran las características de líneas rectas y modelo único se encuentran recogidos en el Anexo A (Tabla 2, del documento externo “Anexos”).

En la Tabla 13 se muestran los 9 trabajos (extraídos de la Tabla 2 del Anexo A) que cumplen las condiciones mencionadas anteriormente.

En la Tabla 13 se han incluido estudios (como el de Gökçen et al., 2006 y Özcan et al., 2009) que en su procedimiento consideran la posibilidad de trabajar con líneas con tiempos de ciclo diferentes, aunque no realizan un experimento computacional para ese caso. El experimento computacional realizado en los 9 artículos de la Tabla 13 es el propuesto en Gökçen et al. (2006). Cabe destacar que dicho experimento computacional solo contiene ejemplares donde el tiempo de ciclo es el mismo para todas las líneas.

En la Tabla 13 se muestra el valor promedio de la desviación media relativa (DMR) del total de ejemplares.

Tabla 13. Comparación del valor promedio de DMR de los procedimientos.

Autores	Abreviatura procedimiento	Valor promedio DMR
Gökçen et al. (2006)	GOK	1.39
Çerçioğlu et al. (2009)	SA	1.20
Özcan et al. (2009)[1]	TS	1.56
Scholl and Boysen (2009)	MOD	4.97
Scholl and Boysen (2009)	ABS	0.57
Baykasoğlu et al. (2009)	ACO	0.55
Guo and Tang (2009)	SS	0.82
Özbakir et al. (2011)	ACO	0.53
Grangeon and Norre (2012)	SA	0.60

EL DMR representa la desviación del número de estaciones obtenido por cada procedimiento respecto a la cota inferior del número de estaciones. Los datos necesarios para el cálculo se han extraído del experimento computacional de los artículos citados en dicha tabla.

El cálculo del DMR se ha realizado con la siguiente fórmula:

$$Desviacion\ Media\ Relativa\ (DMR) = \frac{K_e - LB_e}{LB_e} \cdot 100$$

Donde e hace referencia al ejemplar; K_e representa el número de estaciones obtenidas por el procedimiento para el ejemplar e; y LB_e es la cota inferior del número de estaciones del ejemplar e, que se calcula:

$$LB_e = \left\lceil \sum_{h \in H} \left(\frac{\sum_{i \in TP_h} t_{hi}}{ct_h} \right) \right\rceil$$

Donde t_{hi} es el tiempo de proceso de la tarea $i \in TP_h$ de la línea $h \in H$; H es el conjunto de líneas; TP_h es el conjunto de tareas del producto de la línea $h \in H$; y ct_h es el tiempo de ciclo de la línea $h \in H$.

De los resultados expuestos en la Tabla 13, se observa que los procedimientos que han obtenido un mejor resultado son los basados en el ACO, seguido de ABS, SA y SS.

El resto de los trabajos presentados en la Tabla 2 del Anexo A no se pueden comparar con los 9 seleccionados por uno o varios de los siguientes motivos: 1) no han realizado el experimento computacional propuesto en Gökçen et al. (2006); 2) no han realizado ningún experimento computacional; o 3) el problema considera otras características (tiempos estocásticos, tiempos de configuración (*setup*), diferente función objetivo, operarios con diferentes habilidades, etc.).

[1] El autor solo ha realizado una parte (55 ejemplares) del experimento computacional propuesto por Gökçen et al. (2006).

Los únicos trabajos de la Tabla 2 del Anexo A que en su procedimiento consideran líneas con tiempos de ciclo diferentes (utilizando el procedimiento LCM de Gökçen et al. (2006)) y realizan un experimento computacional son: Özcan (2018) y Özcan (2019). Pero estos trabajos consideran tiempos de tareas estocásticos y tiempos de *setup* variables, respectivamente. En ambos trabajos tampoco se manifiesta el uso de buffers.

De los artículos de la literatura presentados en la Tabla 13, se ha observado que se utilizan 4 procedimientos diferentes para la obtención de una solución inicial factible. A continuación, se detallan los 4 procedimientos:

1) *Passive case procedure* (Gökçen et al., 2006), se presenta para el caso de tener el mismo producto y mismo tiempo de ciclo en ambas líneas. La operativa a seguir es la de utilizar cualquier heurística para el SALBP y obtener una solución para cada una de las líneas por separado. Posteriormente unir dos estaciones opuestas, siempre que la carga de trabajo de la estación resultante no supere el tiempo de ciclo lcm. Gökçen et al. (2006) no realizan un experimento computacional de este procedimiento. Çerçioğlu et al. (2009) utilizan el mismo procedimiento para la construcción de una solución inicial factible para la metaheurística SA y realizan un experimento computacional.
2) Procedimiento orientado a estaciones, donde las tareas se asignan a las estaciones conforme a una regla de prioridad previamente determinada. Este procedimiento se inicia al abrir una estación k, y asignar una tarea candidata (una tarea es candidata si: no ha sido asignada; sus predecesoras ya se han asignado; y su tiempo de proceso no es mayor al tiempo muerto (diferencia entre el tiempo de ciclo y la carga de trabajo) de la estación en curso), de acuerdo con la regla de prioridad utilizada. Si en la estación k coinciden tareas de dos líneas diferentes, entonces dicha estación es multilínea; de lo contrario, la estación es regular. Si ninguna tarea puede ser asignada en la estación k, y todavía quedan tareas por asignar, entonces se abre una nueva estación $k+1$. Este proceso se repetirá hasta que todas las tareas de todas las líneas se hayan asignado a una estación. Este procedimiento es utilizado en las metaheurísticas desarrolladas por Özcan et al. (2009), Baykasoğlu et al. (2009), Guo and Tang (2009) y Özbakir et al. (2011). Cabe destacar que los procedimientos orientados a estaciones, con una selección aleatoria de la tarea candidata a asignar (como regla de prioridad), son los más usuales en la literatura del PALBP a la hora de construir una solución inicial factible.
3) *Active case prodecure* (Gökçen et al., 2006) es un procedimiento similar al expuesto en el punto 2), pero en este caso si hay una tarea candidata con un

tiempo de proceso igual al tiempo muerto de la estación k en curso, entonces, se asigna dicha tarea; de lo contrario, de manera aleatoria se asigna una tarea candidata a la estación k. Este procedimiento es utilizado en Gökçen et al. (2006) y en el SS propuesto por Guo and Tang (2009). En ambos trabajos se ha realizado un experimento computacional.

4) Procedimiento orientado a tareas. En este procedimiento las tareas son ordenadas de acuerdo con un criterio de prioridad. La tarea con el mejor valor del criterio es seleccionada y asignada a la estación con más carga de trabajo, cumpliendo las restricciones de precedencia y de tiempo. Grangeon and Norre (2012) utilizan este procedimiento para generar una solución inicial factible y soluciones vecinas en su procedimiento SA. Los autores realizaron un experimento computacional.

5.2 Heurísticas

Es sabido que el problema PALB es de naturaleza NP-difícil, lo que significa que cuanto más grande es un ejemplar, el tiempo de cálculo para resolverlo de manera óptima con un procedimiento exacto incrementa, habitualmente, de forma exponencial. Esta característica del PALBP hace que los procedimientos exactos solo puedan ser utilizados en ejemplares de un tamaño pequeño o mediano, siendo así necesario el uso de métodos de resolución no exactos para afrontar ejemplares de tamaños reales.

Por este motivo, en la literatura del PALBP se han presentado diversos procedimientos no exactos, como heurísticas o metaheurísticas (ver Gökçen et al., 2006; Özbakir et al., 2011; Özcan, 2018; entre otros), los cuales pueden solucionar ejemplares de tamaño real en un tiempo de cálculo razonable.

Por consiguiente, en las siguientes secciones se han desarrollado varios procedimientos heurísticos basados en reglas de prioridad para resolver el PALBP-B, basándose, además, en los resultados obtenidos al resolver el (sub)problema de dimensionado de buffers en una estación multilínea.

5.2.1 Reglas de prioridad

Los procedimientos propuestos utilizan una o más reglas de prioridad para construir una solución factible. Las reglas de prioridad utilizan datos de los ejemplares del problema para calcular la prioridad de las tareas, e independientemente del método utilizado para generar una solución, las tareas son ordenadas de acuerdo con su valor de prioridad.

A continuación, en la Tabla 14 se muestran 12 reglas de prioridad extraídas de la literatura (Talbot et al., 1986; Hackman et al., 1989; Scholl, 1999), y los parámetros que utilizan.

Parámetros:

H Conjunto de líneas.

TP_h Conjunto de tareas del producto de la línea h; $h \in H$.

ct_h Tiempo de ciclo de la línea h; $h \in H$.

lcm Tiempo de ciclo de todas las líneas: $lcm = m.c.m.\{ct_1, ct_2, \dots, ct_{card\{H\}}\}$.

q_h Número de unidades a producir del producto de la línea h en un superciclo: $q_h = lcm/ct_h$; $h \in H$.

t_{hi} Tiempo de la tarea i de la línea h; $h \in H$; $i \in TP_h$.

t'_{hi} Tiempo de la tarea i de la línea h normalizado al lcm: $t'_{hi} = t_{hi} \cdot q_h$; $h \in H; i \in TP_h$.

UB Cota superior del número de estaciones: $UB = \sum_{h \in H} card\{TP_h\}$.

LB Cota inferior del número de estaciones: $LB = \left\lceil \sum_{h \in H} (\sum_{i \in TP_h} t_{hi}/ct_h) \right\rceil$.

SI_{hi} Conjunto de las sucesoras inmediatas de la tarea i de la línea h; $h \in H$; $i \in TP_h$.

ST_{hi} Conjunto de las sucesoras totales de la tarea i de la línea h; $h \in H$; $i \in TP_h$.

PT_{hi} Conjunto de las predecesoras totales de la tarea i de la línea h; $h \in H$; $i \in TP_h$.

Tabla 14. Reglas de prioridad.

No.	Nombre	Descripción	Procedimiento
1	T	Máximo tiempo de tarea	t'_{hi}
2	NSI	Máximo número de sucesoras inmediatas	$NSI_{hi} = \lvert NSI_{hi} \rvert$
3	NS	Máximo número total de sucesoras	$NS_{hi} = \lvert NS_{hi} \rvert$
4	NT	Mínimo número de tarea	i
5	RPW	Máxima suma del tiempo propio y de las tareas sucesoras	$RPW_{hi} = t'_{hi} + \sum_{j \in NS_{hi}} t'_{hj}$
6	E	Mínimo, estación más temprana en que se podría asignar la tarea *i* de la línea *h*	$E_{hi} = \left\lceil \frac{t'_{hi} + \sum_{j \epsilon PT_{hi}} t'_{hj}}{lcm} \right\rceil$
7	L	Mínimo, estación más tardía a la que se puede asignar la tarea *i* de la línea *h*	$L_{hi} = UB + 1 - \left\lceil \frac{t'_{hi} + \sum_{j \in NS_{hi}} t'_{hj}}{lcm} \right\rceil$
8	H	Mínima holgura	$H_{hi} = L_{hi} - E_{hi}$
9	TL	Máximo tiempo de tarea dividido por L	$TL_{hi} = t'_{hi}/L_{hi}$
10	LTS	Mínimo L dividido por el total de sucesoras de la tarea más 1	$LTS_{hi} = L_{hi}/(NS_{hi} + 1)$
11	TSH	Máximo número total de sucesoras dividido por la holgura más 1	$TSH_{hi} = NS_{hi}/(H_{hi} + 1)$
12	ARPW	Máximo RPW dividido por el total de sucesoras de la tarea más 1.	$ARPW_{hi} = RPW_{hi}/(NS_{hi} + 1)$

En los procedimientos greedy basados en reglas de prioridad presentados en las siguientes subsecciones, los empates entre tareas con el mismo valor de su regla de prioridad se resuelven utilizando, como segundo criterio, el “máximo tiempo de tarea”; en caso de persistir el empate se utilizará un tercer criterio: “mínimo número de tarea”.

En el tercer criterio, puede ocurrir que tareas de diferentes líneas tengan el mismo número de tarea, en este caso, predominará la tarea perteneciente a la línea con el índice h menor. Por ejemplo, la tarea 3 de la línea 1, predomina respecto a la tarea 3 de la línea 2.

5.2.2 Heurísticas greedy de un solo paso

En esta sección se presentan 48 procedimientos heurísticos greedy de un solo paso para resolver el PALBP-B.

Las 48 heurísticas presentadas son heurísticas greedy basadas, cada una de ellas, en una regla de prioridad determinista, por lo que cada una de ellas genera una sola solución factible. 24 de estas heurísticas están basadas en procedimientos orientado a estaciones (OE) (Subsección 5.2.2.1), y las otras 24 heurísticas están basadas en procedimientos orientado a tareas (OT) (Subsección 5.2.2.2).

Para evaluar la solución S obtenida por las heurísticas, el objetivo es minimizar el número total de estaciones utilizadas K. Este objetivo es uno de los más utilizados en la literatura del PALBP. Como se ha introducido anteriormente, el problema que se aborda en esta tesis, no solo se centra en minimizar el número de estaciones utilizadas, sino, también, en minimizar el tamaño de los buffers en las estaciones multilínea. De esta forma, en este trabajo, minimizar la suma total del tamaño de todos los buffers Bf se considera como un objetivo secundario y servirá para desempatar entre soluciones que aporten el mismo número de estaciones K. Por tanto, la función objetivo se puede expresar de la siguiente forma: F.Obj. [MIN] $E(S) = K + Bf \cdot 10^{-6}$; donde $E(S)$ es el valor de la solución S, y $Bf \cdot 10^{-6}$ se reduce a un valor muy pequeño, utilizado para desempatar.

A continuación se recuerda cuáles son las principales premisas asumidas en el PALBP-B: 1) el grafo de precedencias de los productos es conocido; 2) el número de líneas de montaje en paralelo son 2; 3) la asignación de los productos a las líneas es conocido; 4) las líneas de montaje trabajan con tiempos de ciclo ct_h diferentes; 5) el tiempo de proceso de las tareas es conocido y determinista; 6) cada tarea solo se puede asignar a una estación; 7) cada estación esta operada por un único operario; 8) los operarios tienen la capacidad de realizar cualquier tarea; 9) el tiempo de desplazamiento de los operarios en las estaciones es negligible; 10) se dispone de un sistema adecuado para transportar las unidades de una estación a otra, con unos tiempos de transporte negligibles; 11) solo una unidad de producto es transportada de una estación a otra cada tiempo ct_h de la línea h.

5.2.2.1 Procedimientos orientados a estaciones

En esta subsección se presentan dos procedimientos orientados a estaciones, OE-1 (ver Figura 38) y OE-2 (ver Figura 39). Los procedimientos OE presentados siguen la siguiente estructura:

1) Se abre una nueva estación k.
2) Se crea un conjunto de tareas candidatas (CTC) de ambas líneas. Las tareas candidatas son aquellas que: cumplen con las restricciones de precedencia; la carga de trabajo de la estación k más el tiempo de proceso de la tarea no superan el tiempo disponible en la estación ($lcm \geq$ carga trabajo de la estación k + tiempo tarea); y todavía no han sido asignadas a ninguna estación.
3) Si el conjunto de tareas candidatas está vacío y todavía quedan tareas por asignar, entonces se vuelve al paso 1). Si ya no quedan tareas por asignar en ninguna de las líneas, entonces se va al paso 6).
4) Las tareas de CTC se ordenan conforme una regla de prioridad previamente determinada, y se asigna la primera tarea de CTC a la estación k.
5) Si en la estación k no coinciden tareas de 2 líneas diferentes, entonces dicha estación es regular y se vuelve al paso 2). De lo contrario (la estación es multilínea), e independientemente de la regla de prioridad, se ejecutará un procedimiento para resolver el (sub)problema de dimensionado de buffers. Si la solución aportada por el procedimiento para el dimensionado de buffers es no factible, entonces, no se acepta la última tarea asignada a la estación, dicha tarea se quita del conjunto CTC, y si CTC no está vació se vuelve al paso 4), en caso contrario, si CTC está vacío, se vuelve al paso 1). De lo contrario, si la solución aportada es factible, se vuelve al paso 2).
6) Se finaliza el procedimiento y se calcula el valor $E(S)$.

En los procedimientos OE desarrollados a continuación se utiliza la metaheurística HB-SA (Sección 4.5) para la resolución del (sub)problema de dimensionado de buffers en las estaciones multilínea. La elección de HB-SA viene dada por los buenos resultados conseguidos en el experimento computacional (Subsección 4.5.3), al obtener buenas soluciones en un breve periodo de cálculo. El tiempo de cálculo máximo permitido en HB-SA es un valor a fijar posteriormente en el experimento de la Subsección 5.2.3.

A continuación, se presenta el procedimiento OE-1 (Figura 38), a partir del cual se han desarrollado 12 heurísticas con las 12 reglas de prioridad de la Tabla 14. Las heurísticas desarrolladas a partir del procedimiento OE-1 se conocerán como OE-1-1 a OE-1-12.

Por ejemplo, el procedimiento OE-1-5, es la combinación del procedimiento OE-1 y la regla de prioridad número 5 de la Tabla 14.

Sea lcm el tiempo de ciclo de las líneas.
Sea k el número de la estación.
Sea h la línea que pertenece al conjunto H.
Sea CTA el conjunto del total de tareas por asignar.
Sea CTC el conjunto de tareas candidatas de todas las líneas.
Sea WL_k la carga de trabajo de la estación k.
Sea t'_{hi} el tiempo de proceso normalizado al lcm de la tarea i de la línea h.
Sea max_time_{Bf} el tiempo de cálculo máximo de la metaheurística HB-SA.
0. Inicializar:
    $k \coloneqq 0$
    CTA conjunto formado por todas las tareas de todas las líneas
    $max\_time_{Bf} \coloneqq$ tiempo predeterminado
1. ***Mientras*** ($CTA \neq \emptyset$) *hacer*
2. Abrir una nueva estación $k \coloneqq k + 1$ y $WL_k \coloneqq 0$.
3. Crear CTC. Para todas las tareas i de la línea h y para todas las líneas $h \in H$, una tarea i_h será candidata si sus predecesoras ya han sido asignadas, pertenece a CTA y su tiempo de proceso t'_{hi} no viola la restricción del tiempo de ciclo, es decir: $t'_{hi} + WL_k \leq lcm$.
4. ***Si*** ($CTC = \emptyset$) *entonces*
5. Ir al Paso 1.
6. Ordenar las tareas de CTC conforme una regla de prioridad previamente determinada.
7. Asignar la primera tarea i_h de CTC a la estación k.
8. ***Si*** (la estación k es multilínea) *entonces*
9. Ejecutar metaheurística HB-SA (max_time_{Bf}) y obtener una solución.
10. ***Si*** (la solución aportada por la metaheurística *HB-SA=no factible*) *entonces*
11. La tarea i_h asignada se elimina de la estación k y de CTC.
12. ***Si*** ($CTC = \emptyset$) *entonces*
13. ir al Paso 2.
14. ***En caso contrario*** ir al Paso 7.
15. Eliminar la tarea asignada i_h del CTA y actualizar: $WL_k \coloneqq WL_k + t'_{hi}$.
16. Ir al Paso 3.
17. ***Fin Mientras***
18. Devuelve la solución encontrada.

Figura 38. Procedimiento orientado a estaciones OE-1 para la construcción de una solución para el PALBP-B.

A continuación se presenta el procedimiento OE-2 (Figura 39), a partir del cual se han desarrollado 12 heurísticas con las 12 reglas de prioridad de la Tabla 14. Las heurísticas desarrolladas a partir del procedimiento OE-2 se conocerán como OE-2-1 a OE-2-12. A diferencia del anterior procedimiento OE-1, en OE-2 en los puntos 7 al 11 (Figura 39) se comprueba si hay alguna tarea de CTC que su tiempo sea igual al tiempo restante de la estación k, de ser así, e independientemente de la regla de prioridad, se asigna la tarea que cumple esta condición a la estación k. Además, con el objetivo de incrementar al máximo la carga de trabajo de la estación k en curso, en los puntos 13 al 16 (Figura 39) se comprueba si en la actual estación k se puede asignar, por tiempo, alguna otra tarea

de CTC, además de la tarea con el mejor valor de prioridad ya seleccionada. De no ser esto posible, entonces, la tarea seleccionada se cambia por la tarea de CTC con el tiempo de proceso más grande (punto 17, Figura 39), minimizando, de esta forma, el tiempo muerto en la estación.

Sea lcm el tiempo de ciclo de las líneas.
Sea k el número de la estación.
Sea h la línea que pertenece al conjunto H.
Sea CTA el conjunto del total de tareas por asignar.
Sea CTC el conjunto de tareas candidatas de todas las líneas.
Sea WL_k la carga de trabajo de la estación k.
Sea t'_{hi} el tiempo de proceso normalizado al lcm de la tarea i de la línea h.
Sea max_time_{Bf} el tiempo de cálculo máximo de la metaheurística HB-SA.

0. Inicializar:
   $k \coloneqq 0$
   CTA conjunto formado por todas las tareas de todas las líneas
   $max\_time_{Bf} \coloneqq$ tiempo predeterminado
1. ***Mientras*** ($CTA \neq \emptyset$) *hacer*
2. Abrir una nueva estación $k \coloneqq k + 1$ y $WL_k \coloneqq 0$.
3. Crear CTC. Para todas las tareas i de la línea h y para todas las líneas $h \in H$, una tarea i_h será candidata si sus predecesoras ya han sido asignadas, pertenece a CTA y su tiempo de proceso t'_{hi} no viola la restricción del tiempo de ciclo, es decir: $t'_{hi} + WL_k \leq lcm$.
4. ***Si*** ($CTC = \emptyset$) *entonces*
5. Ir al Paso 1.
6. Ordenar las tareas de CTC conforme una regla de prioridad previamente determinada.
7. ***Para todo*** ($i_h \in CTC$) *hacer*
8. ***Si*** ($t'_{hi} = lcm - WL_k$) *entonces*
9. Seleccionar la tarea i_h de CTC.
10. Ir al Paso 18.
11. ***Fin Para***
12. Seleccionar la primera tarea i_h de CTC.
13. ***Para todo*** ($j_h \in CTC | j_h \neq i_h$) *hacer*
14. ***Si*** ($t'_{hi} + t'_{hj} + WL_k \leq lcm$) *entonces*
15. Ir al Paso 18.
16. ***Fin Para***
17. Seleccionar la tarea i_h de CTC con el tiempo de proceso más grande.
18. Asignar la tarea i_h de CTC a la estación k.
19. ***Si*** (la estación k es multilínea) *entonces*
20. Ejecutar metaheurística HB-SA (max_time_{Bf}) y obtener una solución.
21. ***Si*** (la solución aportada por la metaheurística *HB-SA=no factible*) *entonces*
22. La tarea i_h asignada se elimina de la estación k y de CTC.
23. ***Si*** ($CTC = \emptyset$) *entonces*
24. Ir al Paso 2.
25. ***En caso contrario*** ir al Paso 7.
26. Eliminar la tarea asignada i_h de CTA y actualizar: $WL_k \coloneqq WL_k + t'_{hi}$.
27. Ir al Paso 3.
28. ***Fin Mientras***
29. Devuelve la solución encontrada.

Figura 39. Procedimiento orientado a estaciones OE-2 para la construcción de una solución para el PALBP-B.

5.2.2.2 Procedimientos orientados a tareas

En esta sección se presentan dos procedimientos orientados a tareas, OT-1 (ver Figura 40) y OT-2 (ver Figura 41).

Los procedimientos OT propuestos siguen la siguiente estructura:

1) Se abre una nueva estación k.
2) Se crea un conjunto de tareas candidatas (CTC) el cual puede estar formado por tareas de ambas líneas. Una tarea es candidata si todavía no ha sido asignada a ninguna estación y sus predecesoras ya se han asignado.
3) Si el conjunto de tareas candidatas está vacío significa que ya no quedan tareas por asignar en ninguna de las líneas, entonces se va al paso 7).
4) Las tareas del CTC se ordenan conforme a una regla de prioridad previamente establecida, y se selecciona la tarea i con el mejor valor de prioridad.
5) Se busca la última estación k que contenga un predecesor directo de la tarea i. Se comprueba si la carga de trabajo de dicha estación más el tiempo de proceso de la tarea i no excede el tiempo de ciclo; de cumplirse esta condición, se asigna la tarea i a la estación k. En caso contrario, se pasa a la siguiente estación abierta y se verifica que la carga de trabajo resultante en esa estación no exceda el tiempo de ciclo; este proceso se realiza iterativamente hasta encontrar una estación abierta donde se pueda asignar la tarea o hasta llegar la estación K (número total de estaciones abiertas hasta el momento). De no haber ninguna estación (entre la última estación k que contiene un predecesor directo de la tarea i y K) donde se pueda asignar la tarea i, entonces se abre una nueva estación k y se le asigna la tarea i.
6) Si en la estación k no coinciden tareas de 2 líneas diferentes, entonces dicha estación es regular y se vuelve al paso 2). De lo contrario (la estación es multilínea), e independientemente de la regla de prioridad, se ejecutará un procedimiento para resolver el (sub)problema de dimensionado de buffers. Si la solución aportada por el procedimiento para el dimensionado de buffers es no factible, entonces, no se acepta la última tarea asignada a la estación, y dicha tarea se asigna a la siguiente estación más temprana que quepa. Si la tarea no se puede asignar a ninguna estación, entonces, se va al paso 1). De lo contrario, si la solución aportada es factible, se vuelve al paso 2).
7) Se finaliza el procedimiento y se calcula el valor $E(S)$.

Del mismo modo que los procedimientos OE desarrollado en el apartado anterior, los procedimientos OT también utilizan la metaheurística HB-SA (Sección 4.5). para

resolver el (sub)problema de dimensionado de buffers en las estaciones multilínea. El tiempo de cálculo máximo permitido en HB-SA es un valor a fijar posteriormente, en el experimento de la Subsección 5.2.3.

A continuación, se presenta el procedimiento OT-1 (Figura 40), a partir del cual se han desarrollado 12 heurísticas con las 12 reglas de prioridad de la Tabla 14. Las heurísticas desarrolladas a partir del procedimiento OT-1 se conocerán como OT-1-1 a OT-1-12.

Sea lcm el tiempo de ciclo de las líneas.
Sea K el número de estaciones utilizadas, $k = 1,..,K$.
Sea h la línea que pertenece al conjunto H.
Sea CTA el conjunto del total de tareas por asignar.
Sea CTC el conjunto de tareas candidatas de todas las líneas.
Sea WL_k la carga de trabajo de la estación k.
Sea t'_{hi} el tiempo de proceso normalizado al lcm de la tarea i de la línea h.
Sea max_time_{Bf} el tiempo de cálculo máximo de la metaheurística HB-SA.
0. Inicializar:
    $K \coloneqq 1$
    $WL_K \coloneqq 0$
    CTA conjunto formado por todas las tareas de todas las líneas
    $max\_time_{Bf} \coloneqq$ tiempo predeterminado
1. ***Mientras*** ($CTA \neq \emptyset$) *hacer*
2. Crear CTC. Para todas las tareas i de la línea h y para todas las líneas $h \in H$, una tarea i_h será candidata si sus predecesoras ya han sido asignadas y pertenece a CTA.
3. Ordenar las tareas de CTC conforme una regla de prioridad previamente determinada.
4. Seleccionar la primera tarea i_h de CTC.
5. $k_{hi} \coloneqq$ estación con el número más grande que contiene un predecesor directo de la tarea i_h.
6. $k_{best} \coloneqq 0$.
7. ***Para*** ($n \coloneqq k_{hi}, \dots, K$) *hacer*
8. ***Si*** ($t'_{hi} + WL_n \leq lcm$ y $k_{hi} \leq K$) *entonces*
9. $k_{best} \coloneqq n$.
10. Ir al paso 14.
11. ***Fin Para***
12. ***Si*** ($k_{best} = 0$) *entonces*
13. Abrir una nueva estación $K \coloneqq K + 1$, $WL_K \coloneqq 0$ y $k_{best} \coloneqq K$
14. Asignar la tarea i_h a la estación k_{best}.
15. ***Si*** (la estación k_{best} es multilínea) *entonces*
16. Ejecutar metaheurística HB-SA (max_time_{Bf}) y obtener una solución.
17. ***Si*** (la solución aportada por la metaheurística *HB-SA=no factible*) *entonces*
18. La tarea i_h asignada se elimina de la estación k_{best}; $k_{hi} \coloneqq k_{best} + 1$.
19. Ir al Paso 6.
20. Eliminar la tarea asignada i_h de CTA y actualizar: $WL_{k_{best}} \coloneqq WL_{k_{best}} + t'_{hi}$.
21. ***Fin Mientras***
22. Devuelve la solución encontrada.

Figura 40. Procedimiento orientado a tareas OT-1 para la construcción de una solución para el PALBP-B.

A continuación se presenta el procedimiento OT-2 (Figura 41), a partir del cual se han desarrollado 12 heurísticas con las 12 reglas de prioridad de la Tabla 14.

Sea lcm el tiempo de ciclo de las líneas.
Sea K el número de estaciones utilizadas, $k = 1,..,K$.
Sea h la línea que pertenece al conjunto H.
Sea CTA el conjunto del total de tareas por asignar.
Sea CTC el conjunto de tareas candidatas de todas las líneas.
Sea WL_k la carga de trabajo de la estación k.
Sea t'_{hi} el tiempo de proceso normalizado al lcm de la tarea i de la línea h.
Sea NK_{hi} el conjunto de estaciones no válidas para la tarea i de la línea h.
Sea max_time_{Bf} el tiempo de cálculo máximo de la metaheurística HB-SA.
0. Inicializar:
    $K \coloneqq 1$
    $WL_K \coloneqq 0$
    $WL_0 \coloneqq -1$
    CTA conjunto formado por todas las tareas de todas las líneas
    $max\_time_{Bf} \coloneqq$ tiempo predeterminado
1. ***Mientras*** ($CTA \neq \emptyset$) *hacer*
2. Crear CTC. Para todas las tareas i de la línea h y para todas las líneas $h \in H$, una tarea i_h será candidata si sus predecesoras ya han sido asignadas y pertenece a CTA.
3. Ordenar las tareas de CTC conforme una regla de prioridad previamente determinada.
4. Seleccionar la primera tarea i_h de CTC.
5. $NK_{hi} \coloneqq \emptyset$
6. $k_{hi} \coloneqq$ estación con el número más grande que contiene un predecesor directo de la tarea i_h.
7. $k_{best} \coloneqq 0$
8. ***Para*** ($n \coloneqq k_{hi}, \dots, K | \; n \notin NK_{hi}$) *hacer*
9. ***Si*** ($t'_{hi} = lcm - WL_n$) *entonces*
10. La tarea i_h se asigna a la estación $k_{best} \coloneqq n$.
11. Ir al Paso 20.
12. ***Fin Para***
13. ***Para*** ($n \coloneqq k_{hi}, \dots, K | \; n \notin NK_{hi}$) *hacer*
14. ***Si*** ($WL_n > WL_{k_{best}}$ y $t'_{hi} + WL_n \leq lcm$) *entonces*
15. $k_{best} \coloneqq n$.
16. ***Fin Para***
17. ***Si*** ($k_{best} = 0$) *entonces*
18. Abrir una nueva estación $K \coloneqq K + 1$, $WL_K \coloneqq 0$ y $k_{best} \coloneqq K$.
19. Asignar la tarea i_h a la estación k_{best}.
20. ***Si*** (la estación k_{best} es multilínea) *entonces*
21. Ejecutar metaheurística HB-SA (max_time_{Bf}) y obtener una solución.
22. ***Si*** (la solución aportada por la metaheurística *HB-SA=no factible*) *entonces*
23. La tarea i_h asignada se elimina de la estación k_{best}; k_{best} se incluye en NK_{hi}.
24. Ir al Paso 7.
25. Eliminar la tarea asignada i_h de CTA y actualizar: $WL_{k_{best}} \coloneqq WL_{k_{best}} + t'_{hi}$.
26. ***Fin Mientras***
27. Devuelve la solución encontrada.

Figura 41. Procedimiento orientado a tareas OT-2 para la construcción de una solución para el PALBP-B.

Las heurísticas desarrolladas a partir del procedimiento OT-2 se conocerán como OT-2-1 a OT-2-12. A diferencia del anterior procedimiento OT-1, en OT-2 en los puntos 8 al 12 (Figura 41) se comprueba si hay alguna estación que, para la tarea i seleccionada, su tiempo restante sea igual al tiempo de la tarea, de ser así se asigna la tarea i a la estación k que cumple esta condición. De no haber ninguna estación que cumpla la condición anterior, en los puntos 13 al 16 (Figura 41), se asignará la tarea ya seleccionada a la estación abierta (que cumpla restricciones de precedencia y tiempo) con mayor carga de trabajo.

5.2.3 Experimento computacional

En las siguientes subsecciones se muestra cómo se genera el conjunto de ejemplares para PALBP-B (Subsección 5.2.3.1). Los ejemplares generados para PALBP-B son también utilizados en los experimentos computacionales de las siguientes secciones de este capítulo.

En la Subsección 5.2.3.2 se realiza un experimento computacional para evaluar el comportamiento de las heurísticas greedy desarrolladas.

Las heurísticas desarrollados han sido codificadas y ejecutadas en Java SE8, y testeadas en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

5.2.3.1 Generación de ejemplares para PALBP-B

En este apartado se muestra cómo se generan los ejemplares para el PALBP-B, basados en los datos propuestos en Otto et al. (2013). Primero, se introducen las principales características de los ejemplares generados por el procedimiento SALBPGen propuesto por Otto et al. (2013) para SALBP. Posteriormente, se detalla cómo se generan los ejemplares para el PALBP-B.

Cabe destacar que en la literatura PALBP solo existe un estudio, Araújo et al. (2015), donde se utilizan parte de los ejemplares propuestos por Otto et al. (2013).

Otto et al. (2013) destacan que los ejemplares clásicos de la literatura SALBP, como por ejemplo los que se recogen en Scholl (1993) y Scholl (1999), padecen de varias deficiencias, como ser poco diversificados y aleatorizados. Por este motivo, los autores proponen el procedimiento SALBPGen para generar ejemplares de acuerdo con unas pautas y principios específicos: estructura de los grafos de precedencias, *order strength* y la distribución del tiempo de procesos de las tareas. En su estudio, Otto et al. (2013)

generan 2100 ejemplares, los cuales puede ser utilizados directamente o modificados. Estos ejemplares se pueden dividir en 4 grupos respecto a su tamaño:

- Grupo 1: ejemplares con 20 tareas.
- Grupo 2: ejemplares con 50 tareas.
- Grupo 3: ejemplares con 100 tareas.
- Grupo 4: ejemplares con 1000 tareas.

Cada uno de estos grupos está formado por 525 ejemplares (con diferentes grafos de precedencias y tiempos de procesos), con el mismo tiempo de ciclo ct=1000 para todos los ejemplares de todos los grupos.

Además, los ejemplares también se pueden agrupar respecto a las características de los grafos de precedencias: *chain* (grafos donde al menos el 40% de la estructura es en cadena, es decir, una tarea tiene solo una predecesora y una sucesora inmediata); *bottleneck* (grafos que contienen cuellos de botellas); y *mixed* (grafos que contienen cadenas y cuellos de botella, y la proporción de estos es aleatoria).

Los ejemplares también se pueden clasificar según su *order strength* (OS): bajo (OS=0.2); medio (OS=0.6); y alto (OS=0.9).

También se pueden clasificar según la distribución del tiempo de proceso de las tareas: *peak at bottom* (PB, distribución normal en $0.1 \cdot ct$); *peak at middle* (PM, distribución normal en $0.5 \cdot ct$); y *bimodal* (es la combinación de la distribución PM y BM).

Finalmente, los ejemplares también se pueden clasificar según su dificultad. A partir de un estudio computacional, donde cada ejemplar fue resuelto 10000 veces, Otto et al. (2013) obtienen una serie de resultados y observaciones para cada uno de los ejemplares. Una de las observaciones es la dificultad de resolución de cada ejemplar (para la resolución utilizaron un procedimiento heurístico que asigna las tareas de manera aleatoria, respetando las relaciones de precedencias y tiempo de ciclo), que los permite clasificar en:

- Trivial: siempre se obtiene una solución óptima.
- *Less tricky*: entre el (0%, 50%) de los casos no se obtiene la solución óptima.
- *Tricky*: entre el [50%, 95%) de los casos no se obtiene la solución óptima.
- *Very tricky*: entre el [95%, 99.5%) de los casos no se obtiene la solución óptima.
- *Extremly tricky*: en ≥ 99.5% de los casos no se obtiene la solución óptima.
- *Open*: el ejemplar no se puede categorizar porque no se conoce la solución óptima.

Generación de ejemplares para PALBP-B

Con el objetivo de evaluar el rendimiento de los diferentes procedimientos presentados, se han generado 240 ejemplares para PALBP-B a partir de algunos de los 2100 ejemplares originales para SALBP generados por Otto et al. (2013). Para la generación de los 240 ejemplares para el PALBP-B, se tendrán en cuenta ciertas características de los datos originales de Otto et al. (2013), ya que no todos los ejemplares originales son válidos para generar ejemplares para el PALBP-B. Cada ejemplar del PALBP-B estará formado por 2 ejemplares del SALBP, uno para cada línea; por lo tanto, el tamaño del ejemplar (en número de tareas) del PALBP-B resulta de la suma del número de tareas del ejemplar asignado a la línea 1 y del ejemplar asignado a la línea 2.

Los nuevos ejemplares para PALBP-B, se pueden dividir en 4 grupos respecto a su tamaño:

- Grupo 1 (G1): 60 ejemplares de 40 tareas, creados a partir de ejemplares SALBP de 20 tareas de Otto et al. (2013).
- Grupo 2 (G2): 60 ejemplares de 100 tareas, creados a partir de ejemplares SALBP de 50 tareas de Otto et al. (2013).
- Grupo 3 (G3): 60 ejemplares de 200 tareas, creados a partir de ejemplares SALBP de 100 tareas de Otto et al. (2013).
- Grupo 4 (G4): 60 ejemplares de 2000 tareas, creados a partir de ejemplares SALBP de 1000 tareas de Otto et al. (2013).

Para la generación de los nuevos ejemplares del PALBP-B, únicamente se contemplarán aquellos ejemplares que cumplan los siguientes criterios:

- Ejemplares con grafos *mixed*, de esta forma se asegura la diversidad de los grafos.
- Ejemplares con una distribución de tiempos de proceso ya sea PB o BM. Otto et al. (2013) encuentran que es usual en la industria que la mayoría de los tiempos de tarea sigan una distribución normal en $0.1 \cdot ct$ (PB). Sin embargo, también señalan que, en ocasiones, algunas tareas se suelen agregar en grupos por diferentes motivos. El tiempo de estos grupos se acerca a la mitad del tiempo de ciclo, siendo en estos casos la distribución bimodal (BM) la que mejor refleja estas circunstancias.
- Todos los ejemplares indiferentemente de su OS.
- Todos los ejemplares que no sean clasificados como triviales. En un ejemplar trivial todas las soluciones factibles son óptimas (Otto et al., 2013), incluso con

procedimientos heurísticos que generan soluciones de manera aleatoria. Por lo que estos ejemplares no ayudan a evaluar el rendimiento de un procedimiento.

Aplicando los criterios anteriores, se obtiene que en cada uno de los grupos de Otto et al. (2013) hay 150 ejemplares (de los 525 iniciales) que cumplen los criterios marcados y, por tanto, son candidatos para ser seleccionados y crear los nuevos 60 ejemplares de cada grupo para el PALBP-B.

En los nuevos ejemplares para el PALBP-B, como se ha mencionado anteriormente, se trabajará con 2 líneas. Por lo que cada ejemplar del PALBP-B estará formado por 2 ejemplares, uno para cada línea, seleccionados al azar de entre los 150 ejemplares SALBP disponibles de cada grupo. Al seleccionar el ejemplar de cada línea al azar, se puede dar el caso de tener un ejemplar diferente en cada línea o el mismo ejemplar en ambas.

El tiempo de ciclo de los ejemplares se modificará, ya que al ser el mismo ct =1000 en todos los ejemplares, no se producen lotes. Otto et al. (2013) fijan un tiempo de ciclo ct =1000 para todos los ejemplares y a partir de este valor generan los tiempos de proceso de las tareas de cada ejemplar, siguiendo, como se ha introducido anteriormente, una distribución PB, PM o BM. Para mantener la clasificación (PB, PM y BM) de los ejemplares sin alterar, se aumentará o disminuirá, de manera aleatoria, el tiempo de ciclo de cada línea (con el objetivo de no formar lotes mayores a 15 unidades en alguna línea) y se aumentarán o disminuirán los tiempos de las tareas del ejemplar asignado a dicha línea, en proporción al aumento o disminución del tiempo de ciclo. El rango para generar nuevos tiempos de ciclo estará comprendido entre [$ct \cdot 0.5$, $ct \cdot 1.5$] del tiempo de ciclo ct =1000. Los tiempos de ciclo generados tienen que ser diferentes entre sí, con el fin de generar lotes y, así, la posibilidad de generar buffers. Los nuevos tiempos de tarea obtenidos al incrementar o reducir el tiempo de ciclo, se fijan a números enteros, por lo que, si hay algún tiempo de proceso que no sea entero, dicho tiempo pasará a ser el inmediato entero superior.

5.2.3.2 Experimento computacional

Con el objetivo de evaluar el rendimiento de las 48 heurísticas propuestas en la Subsección 5.2.2, se han realizado 6 experimentos computacionales. Para este fin, en cada experimento se resuelven los 240 ejemplares generados como se muestra en la Subsección 5.2.3.1.

En cada uno de los experimentos se ha fijado el tiempo máximo de cálculo max_time_{Bf} de la metaheurística HB-SA, que se usa para resolver el (sub)problema de dimensionado de buffers, a un valor predeterminado. A raíz de los resultados obtenidos por la metaheurística HB-SA en la Subsección 4.5.3, se observó que no es necesario un tiempo de cálculo max_time_{Bf} elevado para encontrar buenas soluciones. Incluso con un tiempo $max\_time_{Bf} = 0.05$ segundos, es suficiente para obtener el mismo número de soluciones factibles que con un tiempo $max\_time_{Bf} = 3600$. Por este motivo, en este experimento, los valores de tiempo max_time_{Bf} utilizados son (en segundos): 0, 0.05, 0.1, 0.5, 1 e infinito (abreviado *inf*). En el caso del tiempo $max\_time_{Bf} = 0$, solo se ejecuta la primera fase (generación de una solución inicial) de HB-SA.

Se recuerda que la función objetivo es de minimizar el valor $E(S) = K + Bf \cdot 10^{-6}$.

Los resultados obtenidos por las heurísticas OE y OT para el experimento computacional se recogen en el Anexo C.1 (véase en el documento externo “Anexos”).

En la Tabla 15, Tabla 16, Tabla 17 y Tabla 18, se muestran, a modo de resumen, los resultados obtenidos por las diferentes heurísticas OE-1-x, OE-2-x, OT-1-x y OT-2-x. En cada tabla, para cada una de las 12 heurísticas de un tipo (representada en una columna) y para cada valor de max_time_{Bf} se muestra: el promedio del total del número de estaciones de todos los ejemplares $\overline{K}$, el promedio del total del tamaño de los buffers de todos los ejemplares $\overline{Bf}$, el promedio del total del tiempo $\overline{CPU}$ (en segundos), y la desviación media relativa (DMR) de las heurísticas respecto a la cota inferior LB_T (suma de la cota inferior del número de estaciones LB y la cota inferior del tamaño de los buffers LB_{Bf}, que se calcula: $LB_T = \left\lceil \sum_{h \in H} \left(\frac{\sum_{i \in TP_h} t_{hi}}{c_h} \right) \right\rceil + LB_{Bf}$ y se asume que la cota $LB_{Bf} = 0$; de este modo una solución S será óptima si $E(S) = LB_T$, es decir, el número de estaciones $K = LB$ y el tamaño de los buffers $Bf = 0$). Además, para cada valor de tiempo max_time_{Bf}, se resalta en negro el mejor (menor) valor $\overline{K}$.

De los resultados de la Tabla 15, Tabla 16, Tabla 17 y Tabla 18, se observa que la heurística que obtiene el mejor promedio total $\overline{K}$, para cualquier valor de max_time_{Bf}, es OE-2-3, seguido de cerca por OE-2-10. Además, se puede observar que las siete heurísticas que obtienen un mejor promedio $\overline{K}$, para cualquier valor de max_time_{Bf}, están basadas en el algoritmo OE-2, siendo estas: OE-2-2, OE-2-3, OE-2-5, OE-2-7, OE-2-8, OE-2-10 y OE-2-11. La heurística con mejor valor DMR es OE-2-5, seguido por OE-2-7 y OE-2-10. Esto demuestra la superioridad del algoritmo OE-2 frente a los demás presentados. De los resultados se observa que, a pesar de ser OE-2-3 el mejor

algoritmo en términos del valor $\overline{K}$, no es el que obtiene el mejor valor DMR. Esto es debido a que obtener resultados ligeramente peores en ejemplares pequeños penaliza porcentualmente más que obtener resultados ligeramente peores en ejemplares grandes (que es lo que ocurre); es por este motivo que OE-2-3 obtiene un peor resultado DMR que OE-2-5.

Por otra parte, también se constata que, como era de esperar, al aumentar el valor del tiempo max_time_{Bf}, se tienden a mejorar las soluciones obtenidas en la mayoría de los procedimientos. Sin embargo, en algunas heurísticas, como por ejemplo OE-1-1, al aumentar el valor de max_time_{Bf} no se obtienen un mejor promedio del valor $\overline{K}$, sino que el valor de $\overline{K}$ tiende a empeorar ligeramente para algunos valores de max_time_{Bf}. Se pone de manifiesto que no siempre un mayor valor de max_time_{Bf}, se traduce en un mejor valor de la solución $E(S)$. Este empeoramiento del valor $\overline{K}$ para algunos valores de max_time_{Bf} ocurre debido al carácter aleatorio de la metaheurística HB-SA.

Además, se observa en varias heurísticas (p.ej. OE-1-1, OE-2-3, OT-1-4, OT-2-6) que, a partir de cierto valor de tiempo max_time_{Bf} ya no se mejora el valor de $\overline{K}$ y únicamente se mejora el valor $\overline{Bf}$. Por ejemplo, escogiendo como referencia la heurística OE-2-3 con $max\_time_{Bf} = 0$, se puede visualizar que entre el valor de DMR de éste y el de OE-2-3 con $max\_time_{Bf} = inf$, hay una mejora del 4.26% a favor del último, pero con un aumento del tiempo de $\overline{CPU}$ de un 1269%. Sin embargo, la mejora de DMR que hay entre $max\_time_{Bf} = 0$ y $max\_time_{Bf} = 0.05$, es de un 2.29% con un aumento en el tiempo de $\overline{CPU}$ de un 25.4%; y la mejora de DMR que hay entre $max\_time_{Bf} = 0$ y $max\_time_{Bf} = 0.5$ es del 4.26% (mismo resultado que $max\_time_{Bf} = inf$), con un aumento del tiempo de $\overline{CPU}$ de un 261.8%. Asimismo, se observa que el valor DMR de OE-2-3 con $max\_time_{Bf} = 0.1$ y $max\_time_{Bf} = inf$ es el mismo DMR=1.211. Lo que demuestra que un valor grande de max_time_{Bf} no necesariamente se traduce siempre en mejores soluciones.

Aunque lo expuesto anteriormente no es una norma general para todas las heurísticas, ya que hay algunas que obtienen un mejor o peor porcentaje de mejora de DMR dependiendo del tiempo max_time_{Bf}, si se puede asumir que, en general, y teniendo en cuenta que el valor de $Bf \cdot 10^{-6}$ es muy pequeño, el valor de DMR no tiende a mejorar a partir de cierto tiempo max_time_{Bf}, que en la mayoría de los casos esta entre 0.1 y 0.5 segundos.

Tabla 15. Resumen de los resultados obtenidos por las diferentes heurísticas OE-1-x al fijar el tiempo max_time_{Bf} de la metaheurística HB-SA a diferentes valores.

max_time_{Bf} (s)	Resultado	LB_T	OE-1-1	OE-1-2	OE-1-3	OE-1-4	OE-1-5	OE-1-6	OE-1-7	OE-1-8	OE-1-9	OE-1-10	OE-1-11	OE-1-12
0	$\overline{K}$	107.279	108.763	108.758	108.875	109.667	109.075	109.950	108.713	110.033	**108.658**	108.892	108.979	109.967
	$\overline{Bf}$	0	0.875	1.033	2.225	3.346	2.242	1.692	2.104	0.542	1.042	2.175	1.967	0.554
	$\overline{CPU}$	-	1.038	0.977	4.647	0.983	1.402	1.179	1.408	1.673	1.331	4.907	5.211	4.913
	DMR %	-	1.844	1.832	1.905	2.880	1.982	3.092	1.663	2.626	1.774	1.849	1.989	2.609
0.05	$\overline{K}$	107.279	108.738	108.725	108.829	109.629	109.025	109.908	**108.650**	110.025	108.671	108.821	108.933	109.954
	$\overline{Bf}$	0	1.013	1.350	3.350	4.283	2.842	2.438	2.838	0.663	1.308	2.958	2.638	0.633
	$\overline{CPU}$	-	1.420	1.724	6.019	1.592	2.343	1.656	2.456	1.997	1.901	6.534	6.686	5.654
	DMR %	-	1.771	1.723	1.882	2.859	1.910	3.022	1.511	2.549	1.744	1.803	1.956	2.598
0.1	$\overline{K}$	107.279	108.742	108.729	108.829	109.617	109.029	109.904	**108.646**	110.017	108.679	108.821	108.929	109.954
	$\overline{Bf}$	0	0.904	1.258	3.192	4.242	2.825	2.454	2.813	0.654	1.379	2.929	2.550	0.604
	$\overline{CPU}$	-	1.787	2.618	7.117	2.302	3.195	2.210	3.666	2.334	2.512	7.563	7.706	5.930
	DMR %	-	1.772	1.675	1.882	2.835	1.911	3.021	1.510	2.547	1.747	1.803	1.955	2.598
0.5	$\overline{K}$	107.279	108.742	108.725	108.825	109.613	109.021	109.904	**108.646**	110.025	108.675	108.821	108.917	109.954
	$\overline{Bf}$	0	0.888	1.321	3.088	3.913	2.617	2.258	2.667	0.583	1.238	2.763	2.346	0.596
	$\overline{CPU}$	-	5.246	9.433	15.467	7.648	10.646	6.193	12.280	4.598	7.148	15.561	14.908	8.140
	DMR %	-	1.772	1.661	1.881	2.834	1.909	3.021	1.510	2.549	1.746	1.803	1.941	2.598
1	$\overline{K}$	107.279	108.742	108.725	108.825	109.613	109.021	109.904	**108.646**	110.025	108.675	108.821	108.917	109.954
	$\overline{Bf}$	0	0.871	1.175	2.983	3.717	2.546	2.288	2.617	0.583	1.196	2.667	2.317	0.563
	$\overline{CPU}$	-	9.218	17.482	25.330	13.736	19.445	10.769	22.677	7.392	12.642	24.919	23.533	10.826
	DMR %	-	1.772	1.661	1.881	2.834	1.909	3.021	1.497	2.549	1.746	1.803	1.941	2.598
inf	$\overline{K}$	107.279	108.742	108.725	108.825	109.613	109.021	109.904	**108.646**	110.025	108.679	108.821	108.917	109.954
	$\overline{Bf}$	0	0.817	1.150	2.888	3.571	2.425	2.183	2.508	0.554	1.142	2.538	2.208	0.550
	$\overline{CPU}$	-	30.063	61.947	67.085	34.353	59.085	30.290	69.458	23.191	40.991	65.206	62.298	25.941
	DMR %	-	1.772	1.661	1.881	2.834	1.909	3.021	1.497	2.549	1.768	1.803	1.941	2.598

Tabla 16. Resumen de los resultados obtenidos por las diferentes heurísticas OE-2-x al fijar el tiempo max_time_{Bf} de la metaheurística HB-SA a diferentes valores.

max_time_{Bf} (s)	Resultado	LB_T	OE-2-1	OE-2-2	OE-2-3	OE-2-4	OE-2-5	OE-2-6	OE-2-7	OE-2-8	OE-2-9	OE-2-10	OE-2-11	OE-2-12
0	$\overline{K}$	107.279	108.763	108.342	**107.983**	108.746	108.021	109.342	108.008	108.592	108.708	107.992	107.992	108.692
	$\overline{Bf}$	0	0.904	1.113	2.296	3.479	2.150	1.596	2.092	0.613	0.954	2.250	2.058	0.692
	$\overline{CPU}$	-	1.176	1.044	5.201	1.120	1.660	1.401	1.650	2.150	1.584	5.699	6.232	5.726
	DMR %	-	1.844	1.470	1.265	1.850	1.194	2.753	1.252	1.858	1.804	1.243	1.208	1.720
0.05	$\overline{K}$	107.279	108.746	108.313	**107.946**	108.692	107.950	109.304	107.963	108.579	108.704	107.954	107.963	108.671
	$\overline{Bf}$	0	1.050	1.450	2.638	4.046	2.629	2.258	2.708	1.083	1.404	2.650	2.246	0.833
	$\overline{CPU}$	-	1.488	2.406	6.522	1.968	2.911	1.981	2.996	2.910	2.129	7.057	7.258	6.380
	DMR %	-	1.773	1.263	1.236	1.823	1.080	2.685	1.144	1.759	1.790	1.199	1.190	1.715
0.1	$\overline{K}$	107.279	108.750	108.304	**107.942**	108.688	107.963	109.300	107.954	108.600	108.704	107.950	107.954	108.675
	$\overline{Bf}$	0	1.121	1.367	2.454	3.850	2.600	2.142	2.558	1.029	1.400	2.554	2.208	0.767
	$\overline{CPU}$	-	1.896	3.746	7.786	2.763	4.028	2.570	4.186	3.704	2.668	8.345	8.313	7.029
	DMR %	-	1.774	1.260	1.211	1.811	1.084	2.684	1.143	1.775	1.774	1.198	1.187	1.715
0.5	$\overline{K}$	107.279	108.746	108.308	**107.942**	108.683	107.954	109.292	107.954	108.583	108.700	**107.942**	107.950	108.667
	$\overline{Bf}$	0	0.971	1.267	2.313	3.596	2.258	2.042	2.483	0.975	1.417	2.396	2.054	0.717
	$\overline{CPU}$	-	5.620	14.448	17.548	8.839	12.812	6.993	13.776	10.256	6.996	18.406	16.661	11.965
	DMR %	-	1.773	1.261	1.211	1.786	1.081	2.658	1.143	1.760	1.789	1.164	1.186	1.714
1	$\overline{K}$	107.279	108.746	108.308	**107.942**	108.679	107.954	109.288	107.954	108.583	108.700	**107.942**	107.954	108.667
	$\overline{Bf}$	0	0.925	1.208	2.263	3.463	2.108	1.958	2.433	0.958	1.346	2.471	1.996	0.654
	$\overline{CPU}$	-	9.278	26.234	27.738	14.824	22.366	11.505	23.848	17.653	11.646	29.242	25.410	17.688
	DMR %	-	1.773	1.261	1.211	1.785	1.081	2.657	1.143	1.760	1.789	1.164	1.187	1.714
inf	$\overline{K}$	107.279	108.742	108.308	**107.942**	108.679	107.954	109.288	107.954	108.583	108.700	**107.942**	107.950	108.667
	$\overline{Bf}$	0	0.871	1.196	2.242	3.383	2.067	1.929	2.496	0.942	1.300	2.404	1.967	0.663
	$\overline{CPU}$	-	27.323	86.490	71.238	36.054	70.026	32.096	67.161	56.063	33.717	75.507	131.424	90.891
	DMR %	-	1.772	1.261	1.211	1.785	1.081	2.657	1.143	1.760	1.789	1.164	1.186	1.714

Tabla 17. Resumen de los resultados obtenidos por las diferentes heurísticas OT-1-x al fijar el tiempo max_time_{Bf} de la metaheurística HB-SA a diferentes valores.

max_time_{Bf} (s)	Resultado	LB_T	OT-1-1	OT-1-2	OT-1-3	OT-1-4	OT-1-5	OT-1-6	OT-1-7	OT-1-8	OT-1-9	OT-1-10	OT-1-11	OT-1-12
0	$\overline{K}$	107.279	110.092	109.229	108.875	109.667	109.075	110.021	**108.721**	109.967	110.004	108.892	108.983	109.871
	$\overline{Bf}$	0	0.367	0.542	2.225	3.338	2.242	1.754	2.208	0.621	0.417	2.175	1.963	0.583
	$\overline{CPU}$	-	1.215	1.269	4.918	1.308	1.717	1.567	1.712	1.933	1.647	5.350	5.601	5.411
	DMR %	-	1.766	1.905	2.880	1.982	3.187	1.613	2.473	2.922	1.849	1.990	2.534	1.766
0.05	$\overline{K}$	107.279	110.088	109.221	108.829	109.608	109.025	109.996	**108.679**	109.950	110.000	108.817	108.950	109.863
	$\overline{Bf}$	0	0.425	0.700	3.279	4.263	2.892	2.479	2.950	0.892	0.404	2.963	2.671	0.696
	$\overline{CPU}$	-	1.753	2.158	7.249	2.359	3.174	2.465	3.343	2.700	2.129	7.681	7.946	6.842
	DMR %	-	1.763	1.882	2.817	1.910	3.180	1.551	2.452	2.921	1.801	1.950	2.487	1.763
0.1	$\overline{K}$	107.279	110.088	109.221	108.829	109.613	109.017	109.996	**108.667**	109.950	110.000	108.821	108.950	109.863
	$\overline{Bf}$	0	0.446	0.679	3.246	4.146	2.746	2.504	2.825	0.883	0.400	2.988	2.554	0.704
	$\overline{CPU}$	-	2.256	3.146	8.408	3.477	4.407	3.436	4.811	3.545	2.792	8.792	9.002	7.231
	DMR %	-	1.763	1.882	2.834	1.908	3.180	1.547	2.452	2.921	1.803	1.961	2.487	1.763
0.5	$\overline{K}$	107.279	110.088	109.221	108.825	109.613	109.017	109.992	**108.667**	109.950	110.000	108.821	108.946	109.863
	$\overline{Bf}$	0	0.392	0.642	3.071	3.933	2.608	2.329	2.663	0.817	0.379	2.738	2.400	0.667
	$\overline{CPU}$	-	3.013	7.424	16.647	8.688	11.633	7.456	13.132	5.820	3.662	16.783	16.207	9.412
	DMR %	-	1.763	1.881	2.834	1.908	3.179	1.547	2.452	2.921	1.803	1.949	2.487	1.763
1	$\overline{K}$	107.279	110.088	109.221	108.825	109.613	109.017	109.992	**108.667**	109.950	110.000	108.821	108.946	109.863
	$\overline{Bf}$	0	0.383	0.633	3.008	3.671	2.558	2.271	2.604	0.796	0.375	2.679	2.346	0.629
	$\overline{CPU}$	-	3.935	12.553	26.518	14.745	20.140	12.249	23.193	8.641	4.749	26.060	24.885	12.040
	DMR %	-	1.763	1.881	2.834	1.908	3.179	1.547	2.452	2.921	1.803	1.949	2.487	1.763
inf	$\overline{K}$	107.279	110.088	109.221	108.825	109.613	109.017	109.992	**108.667**	109.950	110.000	108.821	108.946	109.863
	$\overline{Bf}$	0	0.371	0.613	2.883	3.525	2.442	2.138	2.496	0.758	0.346	2.538	2.175	0.608
	$\overline{CPU}$	-	9.032	38.631	66.366	34.674	58.139	33.246	69.866	24.956	10.018	65.685	63.245	26.436
	DMR %	-	1.763	1.881	2.834	1.908	3.179	1.547	2.452	2.921	1.803	1.949	2.487	1.763

Tabla 18. Resumen de los resultados obtenidos por las diferentes heurísticas OT-2-x al fijar el tiempo max_time_{Bf} de la metaheurística HB-SA a diferentes valores.

max_time_{Bf} (s)	Resultado	LB_T	OT-2-1	OT-2-2	OT-2-3	OT-2-4	OT-2-5	OT-2-6	OT-2-7	OT-2-8	OT-2-9	OT-2-10	OT-2-11	OT-2-12
0	$\overline{K}$	107.279	110.925	109.325	108.729	109.696	108.933	110.133	**108.671**	109.692	110.825	108.775	108.850	109.663
	$\overline{Bf}$	0	0.363	0.567	2.333	3.554	2.513	1.746	2.196	0.625	0.421	2.350	2.100	0.533
	$\overline{CPU}$	-	3.962	4.020	5.115	1.369	1.804	1.618	1.817	2.000	1.691	5.358	5.625	5.364
	DMR %	-	3.688	2.101	1.751	3.081	2.057	3.428	1.705	2.733	3.701	1.809	2.144	2.875
0.05	$\overline{K}$	107.279	110.917	109.321	108.671	109.638	108.863	110.129	**108.625**	109.688	110.825	108.713	108.792	109.646
	$\overline{Bf}$	0	0.479	0.633	3.096	4.771	2.833	2.575	2.917	0.900	0.488	3.067	2.817	0.783
	$\overline{CPU}$	-	2.263	2.872	7.337	2.826	3.778	2.894	3.731	3.323	2.734	7.772	7.935	7.189
	DMR %	-	3.709	2.100	1.708	3.016	1.950	3.426	1.646	2.695	3.710	1.782	2.054	2.872
0.1	$\overline{K}$	107.279	110.917	109.321	108.667	109.633	108.863	110.121	**108.625**	109.696	110.829	108.717	108.788	109.646
	$\overline{Bf}$	0	0.450	0.663	3.050	4.375	2.863	2.571	2.879	0.846	0.479	2.913	2.779	0.763
	$\overline{CPU}$	-	2.450	3.676	8.425	3.529	4.850	3.408	4.799	3.719	2.953	8.956	8.918	7.691
	DMR %	-	3.709	2.100	1.707	3.015	1.950	3.424	1.646	2.698	3.711	1.783	2.052	2.827
0.5	$\overline{K}$	107.279	110.917	109.321	108.658	109.629	108.863	110.121	**108.625**	109.692	110.825	108.713	108.792	109.646
	$\overline{Bf}$	0	0.483	0.608	2.954	4.029	2.725	2.442	2.854	0.854	0.475	3.113	2.758	0.775
	$\overline{CPU}$	-	3.845	9.921	16.570	8.880	12.759	7.405	12.932	6.927	4.624	17.719	16.417	11.574
	DMR %	-	3.709	2.100	1.704	3.014	1.950	3.424	1.646	2.697	3.710	1.781	2.077	2.872
1	$\overline{K}$	107.279	110.917	109.321	108.658	109.629	108.863	110.121	**108.625**	109.692	110.821	108.713	108.783	109.642
	$\overline{Bf}$	0	0.471	0.575	2.854	3.913	2.613	2.413	2.663	0.771	0.375	2.875	2.504	0.704
	$\overline{CPU}$	-	5.573	17.549	26.389	14.856	22.269	12.164	22.739	10.961	6.670	28.019	25.277	16.297
	DMR %	-	3.709	2.100	1.704	3.014	1.950	3.424	1.646	2.697	3.708	1.781	2.051	2.825
inf	$\overline{K}$	107.279	110.917	109.321	108.658	109.629	108.863	110.121	**108.625**	109.692	110.821	108.713	108.783	109.642
	$\overline{Bf}$	0	0.429	0.567	2.700	3.633	2.492	2.292	2.538	0.733	0.350	2.788	2.408	0.658
	$\overline{CPU}$	-	15.835	60.536	68.714	34.647	68.309	33.177	68.552	34.652	18.800	74.728	66.949	47.957
	DMR %	-	3.709	2.100	1.704	3.014	1.950	3.424	1.646	2.697	3.708	1.781	2.051	2.825

En la Tabla 19 se muestra, para los diferentes valores de tiempo max_time_{Bf} (primera columna), el número de veces (en porcentaje) que una de las 12 heurísticas de un tipo (representada en las columnas) obtiene la mejor solución para el total de ejemplares. Se conoce como "mejor solución" a aquella que obtiene el menor valor $E(S)$ de entre todas las soluciones generadas por las 48 heurísticas. La mejor solución para un ejemplar puede ser obtenida por una o más heurísticas de entre las 48 disponibles. En la tabla, para cada valor de tiempo max_time_{Bf}, se resalta en negro el mayor valor. De los resultados de la Tabla 19 se observa que las heurísticas OE-2 son las que aportan más veces la mejor solución. En especial, OE-2-3 y OE-2-11 son los que aportan en más ocasiones (alrededor del 80% de las veces) la mejor solución.

En la Tabla 20 se muestra, para los diferentes valores de tiempo max_time_{Bf} (primera columna), el número de veces (en porcentaje) que una de las 12 heurísticas de un tipo (representada en las columnas) obtiene una solución óptima confirmada (al obtener un valor igual al de la cota LB_T) para el total de ejemplares. En la tabla, para cada valor de max_time_{Bf}, se resalta en negro el mayor valor. De los resultados se observa que los procedimientos que más soluciones óptimas confirmadas obtienen para los diferentes valores de max_time_{Bf} son los procedimientos OE-2. En especial, destacan las heurísticas OE-2-3, OE-2-5, OE-2-7, OE-2-10 y OE-2-11, que en la mayoría de los casos obtienen un resultado igual o mayor al 50% de soluciones óptimas confirmadas.

Del análisis de los resultados de la Tabla 15 a la Tabla 20, se puede concluir que los procedimientos orientados a estaciones, y en especial los OE-2, obtienen un mejor rendimiento promedio que los orientados a tareas. Siendo los procedimientos OE-2-3 y OE-2-5, los que obtienen un mejor rendimiento en términos de $\overline{K}$ y DMR, respectivamente.

Por otra parte, se observa que a partir de un valor de tiempo max_time_{Bf}, el valor de la solución $E(S)$ ya no tiende a mejorar significativamente, mejorando en la mayoría de los casos únicamente el valor de Bf.

Tabla 19. Resumen del número de veces que las heurísticas obtienen la mejor solución para el total de ejemplares.

max_time_{Bf} (s)	Prioridad	1	2	3	4	5	6	7	8	9	10	11	12
0	OE-1	52.92	53.33	49.17	37.08	48.75	39.17	56.67	44.17	53.75	49.58	50.00	46.25
	OE-2	52.92	63.33	77.92	52.50	74.17	41.25	76.25	54.17	53.33	77.08	**79.58**	56.67
	OT-1	38.75	54.17	49.17	37.08	48.75	37.92	55.83	46.25	39.58	49.58	50.00	46.67
	OT-2	35.00	50.00	52.92	35.00	48.75	36.25	55.00	43.75	35.00	51.25	49.58	41.67
0.05	OE-1	52.92	52.08	49.17	35.83	48.33	38.75	55.83	43.33	52.92	49.58	49.17	44.58
	OE-2	52.92	62.50	78.33	52.50	76.25	41.25	76.67	53.33	52.08	78.75	**79.58**	57.08
	OT-1	38.33	53.75	49.58	36.25	48.33	36.67	55.42	45.42	39.58	49.58	50.00	45.00
	OT-2	33.75	50.00	54.17	34.58	48.33	35.00	53.75	42.50	34.17	50.83	49.17	40.00
0.1	OE-1	53.33	51.25	50.00	36.25	48.33	38.75	55.83	43.33	52.92	50.42	49.58	44.58
	OE-2	52.50	61.67	**78.75**	53.33	75.42	41.67	75.83	52.50	52.08	77.92	78.33	56.25
	OT-1	38.33	53.75	49.58	36.25	48.33	36.67	55.42	45.42	40.00	50.42	49.17	45.42
	OT-2	34.17	49.58	53.33	34.58	48.75	35.00	53.75	42.92	34.17	50.83	49.17	40.00
0.5	OE-1	52.92	52.08	50.42	36.67	48.75	39.17	55.83	43.75	52.92	50.42	50.42	44.58
	OE-2	53.33	62.92	**80.83**	53.33	77.50	41.67	76.67	52.50	51.25	78.75	80.42	56.67
	OT-1	38.75	54.58	50.42	36.67	49.17	36.67	55.42	45.42	40.00	50.83	50.42	45.42
	OT-2	33.75	50.00	54.17	35.00	48.75	35.42	53.75	42.92	34.17	51.67	48.75	40.00
1	OE-1	52.50	52.08	50.00	36.67	49.17	38.75	56.67	43.75	52.92	50.42	50.42	44.58
	OE-2	53.33	63.33	80.00	54.17	77.50	41.67	76.25	52.50	50.83	79.17	**80.83**	56.67
	OT-1	38.75	54.58	50.00	36.67	49.17	37.08	55.42	45.83	40.00	50.83	50.42	45.42
	OT-2	33.75	51.25	54.17	35.00	48.75	35.42	53.75	42.92	34.17	51.67	50.00	40.42
inf	OE-1	53.33	52.50	50.42	36.67	49.17	39.17	56.67	43.75	52.50	50.83	50.83	44.58
	OE-2	52.92	63.33	80.00	54.17	77.50	41.67	76.67	52.92	51.67	79.58	**80.42**	57.08
	OT-1	38.75	55.00	50.42	36.67	49.17	37.08	55.42	45.83	40.00	50.83	50.83	45.42
	OT-2	33.75	51.25	54.58	35.00	48.75	35.42	53.75	42.92	34.17	51.67	50.00	40.42

Tabla 20. Resumen del número de veces que las heurísticas obtienen una solución óptima confirmada para el total de ejemplares.

max_time_{Bf} (s)	Prioridad	1	2	3	4	5	6	7	8	9	10	11	12
0	**OE-1**	41.25	42.08	39.58	28.33	37.92	31.25	44.58	35.42	42.08	39.58	39.17	35.83
	OE-2	41.25	47.50	50.00	41.25	49.17	32.08	50.00	42.08	41.67	50.00	**51.25**	43.75
	OT-1	30.00	43.75	39.58	28.33	37.92	30.00	44.17	37.92	31.67	39.58	39.17	35.42
	OT-2	28.33	40.42	43.33	27.92	37.92	28.75	43.33	35.42	27.92	40.83	38.33	31.25
0.05	**OE-1**	42.08	42.50	40.00	28.33	38.75	31.67	45.00	35.00	42.50	40.00	39.17	35.83
	OE-2	41.67	47.92	50.42	42.08	50.83	32.50	50.42	42.50	41.25	50.83	**52.50**	45.00
	OT-1	30.83	44.58	40.42	28.75	38.75	29.58	44.58	37.50	32.50	40.00	40.00	35.42
	OT-2	27.50	41.25	44.58	28.33	38.75	27.92	43.33	35.00	27.92	41.25	38.75	31.25
0.1	**OE-1**	42.50	41.67	40.83	28.75	38.75	31.67	45.00	35.00	42.50	40.83	39.58	35.83
	OE-2	41.67	48.75	51.25	42.50	51.25	32.92	50.42	42.50	41.67	51.67	**52.50**	44.58
	OT-1	30.83	44.58	40.42	28.75	38.75	29.58	44.58	37.50	32.92	40.83	39.17	35.83
	OT-2	27.50	40.83	43.75	28.33	38.75	27.92	43.33	35.42	27.92	41.25	38.75	31.25
0.5	**OE-1**	42.50	42.50	41.25	29.17	38.75	32.08	45.00	35.42	42.50	40.83	40.42	35.83
	OE-2	42.50	48.75	52.50	42.92	52.08	32.92	50.42	42.50	41.67	52.50	**52.92**	44.58
	OT-1	31.25	45.00	41.25	29.17	39.17	29.58	44.58	37.50	32.92	41.25	40.42	35.83
	OT-2	27.50	41.67	44.58	28.75	38.75	28.33	43.33	35.42	27.92	42.08	38.33	31.25
1	**OE-1**	42.08	42.50	40.83	29.17	39.17	31.67	45.42	35.42	42.50	40.83	40.42	35.83
	OE-2	42.50	49.17	52.50	43.75	52.08	32.92	50.42	42.50	41.25	52.92	**53.33**	45.00
	OT-1	31.25	45.00	40.83	29.17	39.17	30.00	44.58	37.92	32.92	41.25	40.42	35.83
	OT-2	27.50	42.50	44.58	28.75	38.75	28.33	43.33	35.42	27.92	42.08	39.58	31.67
inf	**OE-1**	42.50	42.92	41.25	29.17	39.17	32.08	45.42	35.42	42.08	41.25	40.83	35.83
	OE-2	42.50	49.17	52.50	43.75	52.08	32.92	50.83	42.50	41.67	**53.33**	52.92	45.42
	OT-1	31.25	45.42	41.25	29.17	39.17	30.00	44.58	37.92	32.92	41.25	40.83	35.83
	OT-2	27.50	42.50	44.58	28.75	38.75	28.33	43.33	35.42	27.92	42.08	39.58	31.67

5.3 Cóctel de heurísticas

Como ya se ha introducido anteriormente, el cóctel de heurísticas (CH) es un concepto que consiste en utilizar todas las heurísticas rápidas disponibles para resolver un problema, obteniendo tantas soluciones como heurísticas rápidas disponibles; y de entre todas las soluciones, retener la mejor (García-Villoria et al., 2012).

Con el objetivo de obtener mejores soluciones, en esta sección se presentan 7 cócteles de heurísticas, formado cada uno de ellos por un conjunto de heurísticas OE y OT. En la Subsección 5.3.1 se muestran los cócteles de heurísticas diseñados. En la Subsección 5.3.2 se realiza un experimento computacional para evaluar el rendimiento de los procedimientos presentados.

5.3.1 Diseño de los cócteles de heurísticas

En este apartado se desarrollan 7 cócteles de heurísticas. Estos cócteles se conocerán como CH-1 a CH-7. Los cócteles CH-1 a CH-4 están constituidos de la siguiente forma: el CH-1 está formado por las 12 heurísticas basadas en OE-1; el CH-2 está formado por las 12 heurísticas basadas en OE-2; el CH-3 está formado por las 12 heurísticas basadas en OT-1; y el CH-4 está formado por las 12 heurísticas basadas en OT-2.

El funcionamiento de los CH es el siguiente: para cada ejemplar se ejecutan todas las heurísticas una detrás de otra en el orden en el que su regla de prioridad aparece en la Tabla 14. Si durante el procedimiento una heurística encuentra una solución con un valor igual a la cota inferior LB_T, entonces el procedimiento finaliza con una solución óptima confirmada. En cualquier caso, al finalizar el procedimiento se devuelve la mejor solución encontrada.

Los cócteles CH-5 a CH-7, están constituidos por la combinación de algunos o todos los cócteles de CH-1 a CH-4. El CH-5 está formado por los cuatro CH anteriores y en el siguiente orden: CH-2, CH-1, CH-4 y CH-3. El orden de ejecución de los CH en CH-5 viene determinado por el rendimiento mostrado por CH-1, CH-2, CH-3 y CH-4, en el experimento computacional de la Subsección 5.3.2. El objetivo de ejecutar primero los CH con mejor rendimiento es el de encontrar, si es posible, una solución óptima del ejemplar a resolver en la mayor brevedad, y de esta forma poder reducir el tiempo de cálculo de CH-5. En los cócteles CH-6 y CH-7, se eliminan, respecto a CH-5, los cócteles con peor rendimiento: en CH-6 se excluye el cóctel CH-3; y en CH-7 los cócteles CH-3 y CH-4. Esto se hace con el objetivo de observar si el rendimiento de CH-5 se ve afectado al ir excluyendo los cócteles con peor rendimiento. Por lo tanto, el CH-6 está

formado por tres CH y en el siguiente orden: CH-2, CH-1 y CH-3; y el CH-7 está formado por dos CH y en el siguiente orden: CH-2 y CH-1.

5.3.2 Experimento computacional

Para evaluar el rendimiento de los cócteles CH-1 a CH-7 propuestos en la subsección anterior, se han realizado los mismos 6 experimentos computacionales que en la Subsección 5.2.3. Se recuerda que el tiempo máximo de cálculo max_time_{Bf} de la metaheurística HB-SA se ha fijado a diferentes valores predeterminados (en segundos): 0, 0.05, 0.1, 0.5, 1 e infinito.

Los procedimientos desarrollados han sido codificados y ejecutados en Java SE8, y testeados en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

Los resultados obtenidos por los CH para el experimento computacional se recogen en el Anexo C.2 (véase en el documento externo “Anexos”).

En la Tabla 21 se muestra, a modo de resumen, los resultados obtenidos por los cócteles CH-1 a CH-7: el promedio del total del número de estaciones de todos los ejemplares $\overline{K}$, el promedio del total del tamaño de los buffers de todos los ejemplares $\overline{Bf}$, el promedio del total del tiempo $\overline{CPU}$, y la desviación media relativa (DMR) de los procedimientos respecto a la cota inferior LB_T. En esta tabla de resultados también se incluye la cota inferior LB_T y la heurística OE-2-3 (elegida por ser la heurística greedy con mejor promedio $\overline{K}$). Para cada valor de tiempo max_time_{Bf} (primera columna), se resalta en negro el mejor (menor) promedio total $\overline{K}$.

Los resultados de la Tabla 21 reflejan que la heurística OE-2-3 obtiene un mejor resultado $\overline{K}$ que el cóctel CH-1, CH-3 y CH-4, pero un peor resultado que CH-2, CH-5, CH-6 y CH-7. El CH-5, CH-6 y CH-7 son las heurísticas que obtienen mejores resultados, obteniendo todos ellos el mismo resultado $\overline{K}$ y DMR en todos los experimentos. Pero, siendo el CH-5 el que obtienen la mejor solución $E(S)$ en la mayoría de los experimentos, ya que es el que obtiene un valor $\overline{Bf}$ menor. Sin embargo, CH-5 es el cóctel con mayor tiempo $\overline{CPU}$, incrementando en algunos casos un 35% el tiempo del segundo cóctel con mejor valor $E(S)$, es decir CH-6.

Tabla 21. Resumen de los resultados obtenidos por el CH1 a CH7 al fijar el tiempo max_time_{Bf} de la metaheurística HB-SA a diferentes valores.

max_time_{Bf} (s)	Resultado	LB_T	OE-2-3	CH-1	CH-2	CH-3	CH-4	CH-5	CH-6	CH-7
0	$\overline{K}$	107.279	107.983	108.446	107.888	108.625	108.554	**107.883**	**107.883**	**107.883**
	$\overline{Bf}$	0	2.296	0.854	1.367	1.783	1.442	1.354	1.354	1.363
	$\overline{CPU}$	-	5.201	29.609	33.315	33.592	39.675	131.588	99.506	61.672
	DMR %	-	1.265	1.109	0.781	1.190	1.185	0.758	0.758	0.758
0.05	$\overline{K}$	107.279	107.946	108.425	107.854	108.600	108.513	**107.850**	**107.850**	**107.850**
	$\overline{Bf}$	0	2.638	1.054	1.796	2.483	2.038	1.788	1.796	1.813
	$\overline{CPU}$	-	6.522	39.569	44.005	49.417	54.305	180.562	133.480	81.817
	DMR %	-	1.236	1.082	0.742	1.184	1.138	0.719	0.719	0.719
0.1	$\overline{K}$	107.279	107.942	108.438	107.854	108.596	108.513	**107.850**	**107.850**	**107.850**
	$\overline{Bf}$	0	2.454	1.038	1.629	2.446	1.917	1.613	1.625	1.646
	$\overline{CPU}$	-	7.786	48.190	53.988	60.727	62.762	212.735	156.639	98.677
	DMR %	-	1.211	1.085	0.742	1.183	1.138	0.718	0.718	0.718
0.5	$\overline{K}$	107.279	107.942	108.438	107.854	108.596	108.508	**107.850**	**107.850**	**107.850**
	$\overline{Bf}$	0	2.313	0.863	1.454	2.204	2.046	1.442	1.450	1.467
	$\overline{CPU}$	-	17.548	113.364	133.424	116.907	126.214	458.352	351.970	236.793
	DMR %	-	1.211	1.085	0.742	1.183	1.160	0.718	0.718	0.718
1	$\overline{K}$	107.279	107.942	108.438	107.854	108.596	108.504	**107.850**	**107.850**	**107.850**
	$\overline{Bf}$	0	2.263	0.858	1.392	2.167	1.871	1.383	1.392	1.404
	$\overline{CPU}$	-	27.738	191.020	217.914	183.821	201.945	741.249	575.133	391.012
	DMR %	-	1.211	1.085	0.742	1.183	1.135	0.718	0.718	0.718
inf.	$\overline{K}$	107.279	107.942	108.438	107.850	108.596	108.504	**107.846**	**107.846**	**107.846**
	$\overline{Bf}$	0	2.242	0.838	1.383	2.092	1.813	1.375	1.379	1.392
	$\overline{CPU}$	-	71.238	542.957	672.555	340.820	568.432	1927.519	1638.548	1141.170
	DMR %	-	1.211	1.085	0.740	1.183	1.135	0.717	0.717	0.717

También se observa que a mayor valor de tiempo max_time_{Bf} se suele obtener un mejor valor $E(S)$, pero en algunos casos como en CH-1, CH-3 y CH-4, a un mayor valor de tiempo max_time_{Bf} no se traduce en una mejora del valor $\overline{K}$ y únicamente se mejora el valor $\overline{Bf}$. Destaca, por ejemplo, la comparación de CH-5 con $max\_time_{Bf} = 0$ y CH-5 con $max\_time_{Bf} = inf$, donde la mejora de DMR para el último es de un 5.4%, pero con un incremento en el tiempo $\overline{CPU}$ de 1364%. Y también la comparación entre de CH-5 con $max\_time_{Bf} = 0.1$ y CH-5 con $max\_time_{Bf} = inf$, donde la mejora de DMR para el último es de un 0.13%, pero con un incremento en el tiempo $\overline{CPU}$ de 806%.

Nótese que CH-2 obtiene resultados muy cercanos a CH-5, CH-6 y CH-7, pero en un tiempo menor de cálculo. Por ejemplo, una comparación entre CH-2 con $max\_time_{Bf} = 0$ y CH-7 con $max\_time_{Bf} = 0$, el primero obtiene un valor de DMR un 3% peor, pero en un tiempo de cálculo un 74% menor. Y, en el caso de CH-2 con $max\_time_{Bf} = inf$ y CH-7 con $max\_time_{Bf} = inf$, el primero obtiene un valor de DMR un 3.2% peor, pero en un tiempo de cálculo un 72% menor.

La Tabla 22 muestran el número de veces (en porcentaje) que cada cóctel de heurísticas obtiene una solución óptima confirmada para los diferentes valores de tiempo max_time_{Bf}. En la tabla, para cada valor de tiempo max_time_{Bf}, se resalta en negro el mayor valor alcanzado.

De los resultados de la Tabla 22, se observa que el CH-5, CH-6 y CH-7, son los que obtienen un mayor número de veces una solución óptima confirmada (siendo este valor siempre más del 60%), seguidos de cerca por CH-2.

Tabla 22. Número de soluciones óptimas confirmadas obtenidas por el CH1 a CH7 para los diferentes valores de tiempo max_time_{Bf} de la metaheurística HB-SA.

max_time_{Bf} (s)	CH-1	CH-2	CH-3	CH-4	CH-5	CH-6	CH-7
0	53.33	59.17	50.42	51.67	**60.83**	**60.83**	60.42
0.05	53.75	60.42	50.83	51.67	**61.25**	60.83	60.83
0.1	53.75	61.25	50.83	51.67	**62.08**	61.67	61.67
0.5	54.58	61.67	50.83	52.08	**62.50**	**62.50**	**62.50**
1	54.17	61.67	51.25	53.33	**62.50**	62.08	62.08
inf.	54.58	61.67	51.25	53.33	**62.50**	**62.50**	**62.50**

La Tabla 23 muestra el porcentaje de mejora (valor positivo) o no (valor negativo) del valor DMR de CH1 a CH7 respecto al valor de DMR de la heurística OE-2-3, para los diferentes valores de tiempo max_time_{Bf}.

De los resultados de la Tabla 23, se observa que todos los CH obtienen una mejora respecto a OE-2-3, siendo CH-2, CH-5, CH-6 y CH-7 los que obtienen un mayor

incremento de mejora. Estos tres últimos cócteles de heurísticas obtienen el mismo porcentaje de mejora respecto a OE-2-3.

Tabla 23. Porcentaje de mejora de DMR de CH1 a CH7 respecto al valor de DMR de OE-2-3.

max_time_{Bf} (s)	CH-1	CH-2	CH-3	CH-4	CH-5	CH-6	CH-7
0	12.269	38.208	5.869	6.293	40.039	40.039	40.039
0.05	12.446	39.960	4.186	7.957	41.833	41.833	41.833
0.1	10.381	38.760	2.346	6.068	40.672	40.672	40.672
0.5	10.407	38.778	2.374	4.265	40.689	40.689	40.689
1	10.407	38.778	2.374	6.288	40.689	40.689	40.689
inf.	10.407	38.873	2.374	6.288	40.784	40.784	40.784

En general, de los resultados se concluye el buen rendimiento de la heurística OE-2-3, que solo es superado por los CH que la incluyen dentro de su cóctel de heurísticas. Por otra parte, CH-5, CH-6 y CH-7, muestran un rendimiento superior a los demás presentados, pero con un incremento importante en el tiempo $\overline{CPU}$ (cercano al 100% en el caso de CH-7 respecto a CH-5). Se observa que estos tres procedimientos obtienen el mismo resultado $\overline{K}$, pero con diferencias en el valor $\overline{Bf}$. Por ejemplo, entre el mejor cóctel, CH-5, y el tercer mejor cóctel, CH-7, hay un incremento en CH-7 del valor de $\overline{Bf}$ entorno al 0.6% - 2% (dependiendo del valor max_time_{Bf}), pero a su vez una reducción del tiempo $\overline{CPU}$ entorno al 46%-55% (dependiendo del valor max_time_{Bf}).

Se recuerda que CH-2 obtiene unos resultados de $\overline{K}$ y DMR cercanos a CH-5, CH-6 y CH-7, y en un tiempo $\overline{CPU}$ bastante inferior. Sin embargo, para que los resultados de CH-2 puedan igualar a los obtenidos por CH-5, CH-6 y CH-7, el tiempo max_time_{Bf} que necesita CH-2 incrementa de forma significativa. Por ejemplo, CH-7 con un $max\_time_{Bf} = 0.05$ obtiene unos resultados $\overline{K} = 107.850$ y un tiempo $\overline{CPU} = 81.817$, para que CH-2 pueda igualar estos resultados se necesita un valor de $max\_time_{Bf} = inf$, con el que obtiene un valor de $\overline{K} = 107.850$ y un tiempo $\overline{CPU} = 672.555$.

Por tanto, la elección de uno u otro CH dependerá del tiempo disponible para la resolución de los ejemplares y de la calidad de solución deseada.

Además, respecto al valor de max_time_{Bf}, parece necesario tenerlo en cuenta para obtener una buena relación entre el tiempo CPU y el valor $E(S)$, ya que, como se ha podido observar, no siempre un gran esfuerzo computacional en tiempo se traduce en una mejora considerable del valor $E(S)$.

5.4 Metaheurística basada en recocido simulado

Las metaheurísticas son procedimientos no exactos ampliamente utilizados para solucionar problemas de optimización complejos, como son los problemas ALB y PALB.

Con el objetivo de mejorar las soluciones obtenidas por los procedimientos anteriormente propuestos, en la Subsección 5.4.1 se desarrolla una metaheurística basada en el recocido simulado y su experimento computacional se presenta en la Subsección 5.4.2.

5.4.1 Recocido simulado (SA)

Como ya se introdujo anteriormente, el recocido simulado (SA) es un algoritmo que se inspira en el proceso físico de enfriamiento de los metales, el cual ha sido ampliamente utilizado para resolver problemas de optimización combinatoria. El SA es un procedimiento de búsqueda local, que permite aceptar peores soluciones que la solución en curso, con el objetivo de evitar quedar atrapado en óptimos locales.

En la Figura 42 se presenta el procedimiento SA desarrollado para PALBP-B.

El SA propuesto se divide en dos fases. En la primera fase, se genera una solución inicial S_0 al utilizar un cóctel de heurísticas. Si la solución aportada por el cóctel de heurísticas es óptima, se finaliza el procedimiento SA y se devuelve la solución obtenida. En caso contrario, se pasa a la segunda fase (fase de búsqueda), a partir del punto 5 (Figura 42) del algoritmo.

En el SA propuesto, al igual que en el SA presentado en la Sección 4.5, se utiliza una lista de prioridad (PL) donde se consideran todas las tareas de todas las líneas. A cada tarea de la PL se le asigna (sin repetir) un valor de prioridad, el cual, posteriormente, se usa para ordenar las tareas considerando su valor de prioridad. De este modo, se obtiene una secuencia de tareas que se utiliza para construir una solución. La variación de los valores de prioridad (de la PL) de una solución dada, proporciona, de esta forma, una solución vecina.

Sea $E(S)$ el valor de la función objetivo de la solución S.
Sea LB_T la cota inferior de la suma del número de estaciones y del tamaño de los buffers.
Sea $N(S)$ el vecindario de la solución S.
Sea t la temperatura del sistema.
Sea max_time el tiempo máximo disponible de computación.
0. Inicializar los parámetros:
    t_0 temperatura inicial
    t_{min} temperatura mínima
    itt número de iteraciones donde la temperatura permanece igual
    α factor de enfriamiento
    p probabilidad de movimiento
    $t \coloneqq t_0$
1. Generación de una solución inicial (S_0)
2. ***Si*** ($E(S_0) = LB_T$) *entonces*
3. $S_B \coloneqq S_0$;
4. Ir al paso 28;
5. $S_C \coloneqq S_0$; $S_B \coloneqq S_C$;
6. Construcción de la lista de prioridad PL.
7. ***Mientras*** ($t \geq t_{min}$ y $E(S_B) > LB_T$ y no se cumpla el tiempo max_time) *hacer*
8. $iter \coloneqq 0$
9. ***Mientras*** ($iter < itt$) *hacer*
10. ***Si*** ($aleatorio[0,1) < p$) *entonces*
11. Movimiento *swap* en la lista PL.
12. ***En caso contrario*** movimiento *insert* en la lista PL
13. Generar la solución vecina S_N de $N(S_c)$.
14. Sea $\Delta = E(S_N) - E(S_c)$.
15. ***Si*** ($\Delta \leq 0$) *entonces*
16. $S_C \coloneqq S_N$
17. ***Si*** ($E(S_C) < E(S_B)$) *entonces*
18. $S_B \coloneqq S_C$;
19. ***Si*** ($E(S_B) = LB_T$) *entonces*
20. Ir al Paso 27.
21. ***En caso contrario***
22. ***Si*** ($aleatorio[0,1) < e^{-\Delta/t}$) *entonces*
23. $S_C \coloneqq S_N$
24. $iter \coloneqq iter + 1$
25. ***Fin Mientras***
26. $t \coloneqq \alpha \cdot t$
27. ***Fin Mientras***
28. Devuelve la mejor solución encontrada S_B.

Figura 42. Procedimiento basado en la metaheurística SA para resolver el PALBP-B.

A continuación se especifican los elementos del SA propuesto en la Figura 42.

Generación de una solución inicial

En la primera fase del SA se genera una solución inicial S_0 que es la solución aportada por el cóctel CH-7. A partir de los resultados obtenidos del experimento computacional en la Subsección 5.3.2, se ha observado que el cóctel CH-7 es el que aporta mejores

resultados en términos de calidad y tiempo de cálculo. Por este motivo se decide utilizar dicho procedimiento para la generación de la solución S_0.

Si el valor de la solución inicial $E(S_0)$ es igual a la cota inferior LB_T, entonces la solución S_0 es óptima y se finaliza el procedimiento. En caso contrario, se pasa a la segunda fase.

Inicio de la segunda fase

Al no ser el valor de la solución inicial $E(S_0)$ igual a la cota inferior LB_T, se ejecuta la segunda fase del procedimiento SA. En la segunda fase (punto 5, Figura 42), la solución S_0 pasa a ser la solución en curso S_c y a su vez la mejor solución conocida S_B.

Construcción de una lista de prioridad (PL) de las tareas

La lista de prioridades (PL) se representa como una lista de las tareas de todas las líneas (Figura 43) de longitud N, donde $N = \sum_{h \in H} card(TP_h)$, siendo TP_h el conjunto de tareas pertenecientes al producto de la línea h ($h \in H$). En la PL las tareas tienen una posición y a cada posición se le asigna, sin repetir, un valor de prioridad φ. Siendo el rango del valor de prioridad $\varphi = N, \dots, 1$.

Para la construcción inicial de la PL, a cada tarea se le asigna un valor de prioridad φ siguiendo y respetando la secuencia de tareas de línea 1 y línea 2 obtenida en la construcción de la solución inicial S_0. Donde, a la primera tarea en ser secuenciada (ya sea de la línea 1 o línea 2) se le asigna el valor de prioridad más grande, es decir N, y así sucesivamente, hasta que a la última tarea en ser asignada (ya sea de la línea 1 o línea 2) se le asigna el valor 1.

Tarea i[h]	1[1]	2[1]	3[1]	4[1]	1[2]	2[2]	3[2]	4[2]	5[2]
φ (i[h])	9	7	6	4	8	5	3	2	1

Figura 43. Ejemplo de una lista de prioridad (PL).

A continuación se explica cómo se genera una solución a partir de una PL. El uso de los datos de la PL para cualquier procedimiento presentado en la Sección 5.2 y 5.3 se realiza de la siguiente forma. Se recuerda que los procedimientos presentados (Sección 5.2 y 5.3) utilizan un conjunto de tareas candidatas (CTC) donde las tareas se ordenar de acuerdo a una regla de prioridad predeterminada. En el SA propuesto, las tareas de CTC se ordenan de acuerdo a su valor de prioridad φ de la tabla PL. De esta forma, se introduce una nueva regla de prioridad, donde las tareas candidatas se ordenan de forma decreciente a su valor de prioridad φ.

Generación de una solución vecina

En el SA propuesto se generan dos nuevas soluciones vecinas S_N al cambiar el valor de prioridad φ de dos o más tareas de la PL, y de esta forma crear una nueva PL. Posteriormente, se construyen dos nuevas soluciones vecinas al utilizar la nueva PL (que proporciona la lista ordenada de tareas a secuenciar) y los procedimientos OE-1 (Figura 38) y OE-2 (Figura 39). La elección de estos dos procedimientos es debido al buen rendimiento mostrado en el experimento computacional de la Subsección 5.2.3.

En el SA propuesto se utilizan dos formas, *swap* e *insert* (que se utilizan con probabilidad *p* y 1-*p* (donde *p* es un parámetro a calibrar y aquí conocido como: probabilidad de movimiento), respectivamente), de cambiar el valor φ de las tareas y, así, crear nuevas PL: 1) movimiento swap, intercambia los valores de prioridad de dos tareas al azar (ver Figura 44a); y 2) movimiento *insert*, selecciona dos tareas al azar (*rnt1* y *rnt2*), y desplaza el valor de prioridad de todas las tareas comprendidas entre estas a la derecha; el valor en *rnt1* también se desplaza a la derecha y el de *rnt2* a la posición de *rnt1* (ver Figura 44b).

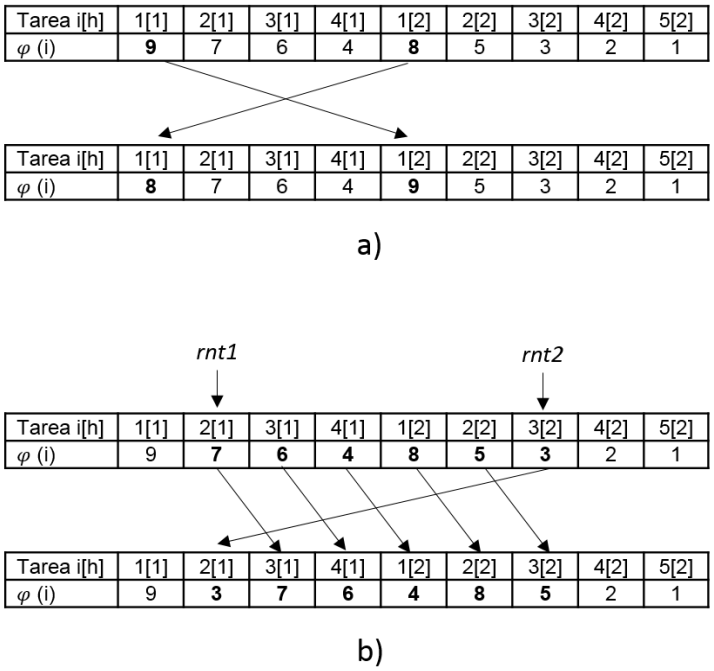

Figura 44. a) ejemplo de movimiento *swap*, b) ejemplo de movimiento *insert*.

A partir de la nueva PL creada, ya sea por el movimiento *swap* o *insert*, se construyen dos soluciones al utilizar los procedimientos OE-1 y OE-2. Como se ha comentado anteriormente, en estos procedimientos, las tareas de CTC son ordenadas de acuerdo con una regla de prioridad predeterminada; sin embargo, ahora, se ordenan de acuerdo con su valor de prioridad φ de la nueva PL. De esta forma, se introduce una nueva regla de prioridad, donde las tareas candidatas se ordenan de forma decreciente a su valor

de prioridad φ. La incorporación de la PL en los OE-1 y OE-2, implica un mínimo cambio en los procedimientos y se realiza de la siguiente forma.

En el procedimiento OE-1, en el punto 6 (Figura 38), las tareas de CTC se ordenan de acuerdo con una regla de prioridad predeterminada (de la Tabla 14). En el caso actual, las tareas de CTC se ordenan (en orden decreciente) de acuerdo con su valor de prioridad φ en la nueva lista PL. En OE-2 se sigue el mismo procedimiento que en OE-1, en el punto 6 (Figura 39), las tareas de CTC se ordenan (en orden decreciente), ahora, de acuerdo con su valor de prioridad φ en la nueva lista PL. De esta forma se incorpora la lista PL a los procedimientos OE-1 y OE-2.

La mejor solución obtenida entre el procedimiento OE-1 y OE-2 será la solución vecina S_N.

Reducción de la temperatura

La temperatura t del algoritmo SA influye en la probabilidad de aceptar peores soluciones vecinas que la actual solución en curso S_c. La temperatura t se irá reduciendo gradualmente cada determinado número de iteraciones itt. Una de las formas más habituales en la literatura para realizar esta reducción es la reducción geométrica, es decir, $t = \alpha \cdot t$, donde α es el factor de enfriamiento: $0 < \alpha < 1$. El valor del parámetro α, así como la temperatura inicial t_0, la temperatura mínima t_{min} y el número de iteraciones donde la temperatura permanece igual itt, son parámetros del algoritmo que deben ser calibrados.

Función objetivo

La función objetivo del problema es minimizar, de forma jerárquica, el número de estaciones utilizadas y el tamaño de los buffers, por lo tanto: F.Obj. [MIN] $E(S) = K + Bf \cdot 10^{-6}$; donde: $E(S)$ es el valor de la solución S; K es el número total de estaciones utilizadas; y Bf es la suma del tamaño de todos los buffers.

Criterio de parada

El SA propuesto finaliza cuando se cumple al menos uno de los tres siguientes criterios de parada: 1) cuando el valor de una solución $E(S)$ es igual a la cota LB_T; 2) cuando se alcanza la temperatura mínima t_{min}; o 3) cuando se excede un tiempo de cálculo prestablecido max_time.

5.4.2 Experimento computacional

En este apartado se presenta la calibración de los parámetros del SA (Subsección 5.4.2.1) y el experimento computacional realizado (Subsección 5.4.2.2).

El procedimiento SA desarrollado ha sido codificado y ejecutado en Java SE8, y testeado en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

5.4.2.1 Calibración de los parámetros del SA

Los valores de los parámetros t_{min}, t_0, α, p y itt del SA, y el parámetro del tiempo max_time_{Bf} de la metaheurística HB-SA (Sección 4.5), han sido calibrados utilizando el algoritmo Nelder and Mead (N&M) (Nelder and Mead, 1965).

Se recuerda que los ejemplares generados del PALBP-B se pueden clasificar en 4 grupos respecto a su tamaño. Un conjunto de entrenamiento formado por 160 ejemplares (40 ejemplares de cada grupo) son generados como se explica en la Subsección 5.2.3.1, y se utilizan para calibrar los parámetros del SA. Estos ejemplares son resueltos fijando el tiempo de cálculo máximo del SA $max\_time = infinito$.

Se han realizado dos experimentos de calibración diferentes. En el primer experimento, los parámetros del SA se han calibrado, exclusivamente, para cada uno de los grupos de ejemplares (G1, G2, G3 y G4), utilizando los 40 ejemplares de entramiento generados de cada grupo. Este tipo de calibración se denomina “SA calibración individual”.

En el segundo experimento los parámetros del SA se han calibrado considerando conjuntamente ejemplares de los cuatro grupos (10 ejemplares de cada grupo, 40 en total). La calibración del segundo experimento se denomina “SA calibración conjunta”.

Los valores obtenidos de los parámetros del SA para el primer experimento son los siguientes:

Grupo 1 (G1): $t_0 = 81; t_{min} = 0.004; \alpha = 0.54; itt = 121; p = 0.59; max\_time_{Bf} = 0.002.$

Grupo 2 (G2): $t_0 = 104; t_{min} = 0.123; \alpha = 0.74; itt = 84; p = 0.56; max\_time_{Bf} = 0.01.$

Grupo 3 (G3): $t_0 = 297; t_{min} = 0.011; \alpha = 0.482; itt = 8; p = 0.48; max\_time_{Bf} = 0.035.$

Grupo 4 (G4): $t_0 = 18; t_{min} = 0.091; \alpha = 0.538; itt = 8; p = 0.51; max\_time_{Bf} = 0.03.$

Los valores obtenidos para los parámetros del SA para el segundo experimento son los siguientes:

Grupo todos: $t_0 = 98$; $t_{min} = 0.184$; $\alpha = 0.489$; $p = 0.57$; $itt = 5$; $max\_time_{Bf} = 0.037$.

5.4.2.2 Experimento computacional

Con el fin de evaluar el rendimiento del SA propuesto, se ha realizado un experimento computacional similar al realizado en la Subsección 5.2.3.2, pero en este caso, el tiempo max_time_{Bf} es un valor ya calibrado. El experimento ha sido ejecutado 5 veces de manera independiente y el promedio de los resultados $\overline{K}$, $\overline{Bf}$ y tiempo $\overline{CPU}$ obtenidos para cada ejemplar, se recoge en el Anexo C.3 (documento externo "Anexos").

La Tabla 24 muestra el valor promedio de los resultados $\overline{K}$, $\overline{Bf}$ y tiempo $\overline{CPU}$ del total de ejemplares de cada grupo (G1, G2, G3, G4). La fila "Calibración individual" muestra los resultados obtenidos por el SA al utilizar los valores de los parámetros calibrados de manera individual para cada grupo. La fila "Calibración conjunta", muestra los resultados obtenidos por el SA al utilizar los valores de los parámetros al calibrar todos los grupos de manera conjunta. La columna "Total" muestra el promedio de los resultados de los grupos G1, G2, G3 y G4.

Tabla 24. Resumen y comparación de los resultados obtenidos por el SA respecto a la cota LB_T y la heurística CH-5 con $max\_time_{Bf} = inf$.

Procedimiento		Resultado	G1	G2	G3	G4	Total
$\mathbf{LB_T}$		$\overline{\mathbf{K}}$	8.267	18.650	35.717	366.483	107.279
CH-5 (inf)		$\overline{\mathbf{K}}$	8.317	18.800	36.067	368.200	107.846
		$\overline{\mathbf{Bf}}$	0.033	0.067	0.017	5.383	1.375
		$\overline{\mathbf{CPU}}$	18.193	154.033	276.369	7261.480	1927.519
SA	Calibración individual	$\overline{\mathbf{K}}$	8.300	18.770	36.067	368.187	107.831
		$\overline{\mathbf{Bf}}$	0.000	0.000	0.040	5.653	1.423
		$\overline{\mathbf{CPU}}$	0.447	5.854	11.706	1843.729	465.434
	Calibración conjunta	$\overline{\mathbf{K}}$	8.310	18.800	36.070	368.223	107.851
		$\overline{\mathbf{Bf}}$	0.033	0.060	0.070	4.587	1.188
		$\overline{\mathbf{CPU}}$	0.752	2.602	6.803	1922.366	483.131

De los resultados de la Tabla 24, se observa que al calibrar los parámetros del SA para cada grupo en específico (calibración individual) se obtienen soluciones, en términos de $\overline{K}$, ligeramente mejores que al calibrar los parámetros considerando todos los grupos (calibración conjunta) y también mejores que el cóctel de heurísticas CH-5 con $max\_time_{Bf} = inf$. También se observa que el SA con "calibración conjunta" obtiene unos resultados ligeramente peores que el cóctel de heurísticas CH-5. Sin embargo, el SA en sus dos tipos de calibración, reduce considerablemente el tiempo de cálculo empleado por CH-5. Siendo esta mejora en promedio de un 76% en el caso del SA calibración individual y de un 75% para el SA calibración conjunta.

La Tabla 25, muestra el número total de soluciones óptimas confirmadas (es decir, el número de ejemplares resueltos por el SA para los cuales se cumple: $E(S) = LB_T$, por lo menos para 1 de las 5 ejecuciones) y la desviación media relativa (DMR) del SA respecto a la cota LB_T.

Tabla 25. Número de soluciones óptimas confirmadas y DMR obtenidos por el SA respecto a la cota LB_T, y comparación de los resultados respecto a la heurística CH-5 con $max\_time_{Bf} = inf$.

Procedimiento		Resultado	G1	G2	G3	G4	Total
CH-5 (inf)		N.º soluciones óptimas	55	49	40	6	150
		DMR %	0.682	0.840	0.905	0.442	0.717
SA	Calibración individual	N.º soluciones óptimas	58	53	46	5	162
		DMR %	0.444	0.642	0.905	0.439	0.607
	Calibración conjunta	N.º soluciones óptimas	58	51	40	6	155
		DMR %	0.587	0.840	0.914	0.450	0.698

De los resultados obtenidos, se confirma que el SA "calibración individual" obtiene un mayor número de soluciones óptimas confirmadas y un menor valor DMR que el SA "calibración conjunta" y que la heurística CH-5. En concreto, el SA "calibración individual" obtiene una solución óptima confirmada en el 67.5% de los ejemplares, el SA "calibración conjunta" en el 64.5%, y el CH-5 en el 62.5%.

Respecto al valor DMR se observa que el SA en sus dos tipos de calibración, obtiene mejores resultados que la heurística CH-5. Mejorando, ambos SA, "calibración individual" y "calibración conjunta", el valor DMR de CH-5 en un 15.3% y 2.65%, respectivamente. Y, obteniendo el SA "calibración individual" un valor DMR un 13% menor que el SA "calibración conjunta".

5.5 Metaheurística basada en optimización por colonia de hormigas

5.5.1 Optimización por colonia de hormigas (ACO)

El algoritmo de optimización por colonias de hormigas (*Ant Colony Optimization* (ACO)) fue propuesto por Doringo (1992). El ACO es una metaheurística que se inspira en la forma en la que las hormigas buscan la manera más rápida para llegar a la fuente de comida. En la naturaleza las hormigas recorren diferentes caminos para llegar a un mismo punto, y en cada camino dejan un rastro de feromonas. Las hormigas posteriores se guían por el rastro de feromonas depositados en los caminos, y tienden a seguir aquellos con una mayor intensidad de feromonas.

En el algoritmo ACO cada hormiga sigue un camino de manera estocástica, con una probabilidad que depende del nivel de feromona del camino y de una información heurística, y depositan un rastro de feromonas en cada camino que realizan. En cada etapa, el nivel de feromona de cada camino es actualizado iterativamente de acuerdo con la calidad del recorrido (solución) aportada por las hormigas.

En la literatura del PALBP se observa que los mejores resultados han sido obtenidos por la metaheurística ACO, y en especial por el ACO con múltiples sub-colonias (mc-ACO) propuesto por Özbakir et al. (2011). En el mc-ACO, los autores utilizan 3 sub-colonias, donde cada sub-colonia tiene asignada una regla de prioridad diferente, que es utilizada por las hormigas de dicha sub-colonia para seguir un camino que las lleva a construir una solución S.

A partir de lo expuesto anteriormente, se propone un procedimiento basado en la metaheurística ACO con múltiples sub-colonias (Figura 45), con el objetivo de resolver el PALBP-B.

A continuación, se da una introducción al procedimiento ACO desarrollado en esta sección.

En el ACO propuesto durante un número predeterminado de iteraciones $iter$ (valor a calibrar), se crea un número C de sub-colonias, con M hormigas cada una. Previamente, a todas las hormigas de una sub-colonia, se le asigna sin repetir una regla de prioridad (extraída de la Tabla 14), la cual es utilizada por las hormigas de dicha sub-colonia como información heurística. La información heurística, indicada como η_i, proporciona el valor de prioridad de la tarea i. Cabe destacar que se dispondrá de 12 sub-colonias, una por cada regla de prioridad de la Tabla 14. De esta forma, el número de sub-colonias será igual al número de reglas de prioridad utilizadas.

Cada hormiga a de cada sub-colonia construye una solución factible al utilizar el procedimiento descrito y explicado posteriormente en la Figura 46. En el ACO propuesto, una hormiga a utiliza diferentes estrategias (exploración y explotación) para, iterativamente, seleccionar y asignar una tarea i de CTC (conjunto de tareas candidatas; una tarea es candidata si: no ha sido asignada; sus predecesoras ya se han asignado; y su tiempo de proceso no es mayor al tiempo muerto de la estación en curso) a la estación k en curso, de acuerdo con la información heurística η_i y el rastro de feromonas $\tau_{i,k}$ de las tareas candidatas. El rastro de feromonas indica que tan bueno o deseable es asignar la tarea i a la estación k. Esta política de rastro de feromonas, conocida como par tarea-estación, es utilizada en Bautista and Pereira (2007) y Özbakir et al. (2011).

La asignación de una tarea i a una estación k se conoce como camino (i, k) y cada camino tiene asociado un rastro de feromonas $\tau_{i,k}$. Cuando ya no quedan más tareas por asignar, la hormiga a terminar su recorrido y devuelve una solución S_a.

Una particularidad del ACO propuesto es que el rastro de feromonas $\tau_{i,k}$ que utilizan las hormigas está compuesto por: un rastro de feromonas local y un rastro de feromonas global. Cada una de las sub-colonias dispone de su propia matriz de feromonas local. Después de que todas las hormigas de una sub-colonia terminan su recorrido, la cantidad de feromona de todos los pares tarea-estación de la matriz local se actualizan, y en los pares tarea-estación explorados por las hormigas se deposita una cantidad de feromonas. Una vez que todas las hormigas de todas las sub-colonias finalizan su recorrido, todos los pares tarea-estación de la matriz de feromonas global se actualiza y en los pares tares-estación explorados por la mejor solución aportada por las sub-colonias S_{SC} se deposita una cantidad de feromonas.

Para inicializar las matrices de feromonas local y global, en cada par tarea-estación se deposita una cantidad de feromonas inicial aleatoria comprendida entre dos valores a calibrar. Con el objetivo de prevenir una convergencia prematura del procedimiento, después de cada iteración se deposita una cantidad constante (a calibrar) de feromonas en todos los pares tarea-estación de las matrices locales y globales. Además, cada determinado número de iteraciones se inicializa de nuevo la cantidad de feromonas (valor aleatorio comprendido entre dos valores a calibrar) en todos los pares tarea-estación en las matrices locales y globales.

Al finalizar el procedimiento, se devuelve la mejor solución encontrada S_G.

Sea $E(S)$ el valor de la función objetivo de la solución S.
Sea LB_T la cota inferior de la suma del número de estaciones y del tamaño de los buffers.
Sea C el número de sub-colonias.
Sea M el número de hormigas pertenecientes a una sub-colonia.
Sea R el número de iteraciones para reiniciar los valores de las feromonas.
Sea max_time el tiempo máximo disponible de computación.
0. Inicializar:
    $iter$ número de iteraciones
    cte_fer constante de feromona a depositar en el par tarea-estación
    $E(S_G) \coloneqq UB + 1$ (UB: cota superior del número de estaciones)
1. Asignar una cantidad de feromona en todos los pares tarea-estación, en todas las matrices locales y global. La cantidad de feromona a depositar es un valor aleatorio entre (τ_0,τ_F).
2. $i \coloneqq 0$
3. $r \coloneqq 0$
4. Asignar una regla de prioridad a ser utilizada en cada sub-colonia $subC$.
5. ***Mientras*** ($i < iter$ y $E(S_G) > LB_T$ y no se cumpla el tiempo max_time) *hacer*
6. $subC \coloneqq 1$; $E(S_{SC}) \coloneqq UB + 1$;
7. ***Mientras*** ($subC \leq C$) *hacer*
8. $a \coloneqq 1$;
9. ***Mientras*** ($a \leq M$) *hacer*
10. Construir la solución S_a de la hormiga a.
11. ***Si*** ($E(S_a) < E(S_{SC})$) *entonces*
12. $S_{SC} \coloneqq S_a$
13. ***Si*** ($E(S_{SC}) < E(S_G)$) *entonces*
14. $S_G \coloneqq S_{SC}$
15. ***Si*** ($E(S_G) = LB_T$) *entonces*
16. Ir al Paso 29.
17. $a \coloneqq a + 1$
18. ***Fin Mientras***
19. Actualizar matriz de feromonas local de $subC$.
20. $subC \coloneqq subC + 1$
21. ***Fin Mientras***
22. Actualizar matriz de feromonas global a partir de S_{SC}.
23. $i \coloneqq i + 1$
24. $r \coloneqq r + 1$
25. ***Si*** ($R < r$) *entonces*
26. Asignar aleatoriamente una cantidad de feromona (τ_0,τ_F) entre todos los pares tarea-estación, para todas las matrices locales y global.
27. $r \coloneqq 0$
28. ***En caso contrario*** deposita una cantidad de feromona cte_fer entre todos los pares tarea-estación, para todas las matrices locales y global.
29. ***Fin Mientras***
30. Devuelve la mejor solución S_G encontrada.

Figura 45. Procedimiento basado en la metaheurística ACO para resolver el PALBP-B.

Construcción de una solución factible por una hormiga

Cada hormiga construye una solución S_a siguiendo un procedimiento orientado a estaciones (OE-ACO, ver Figura 46) y utilizando la regla de prioridad asignada a su sub-colonia. El procedimiento OE-ACO propuesto está basado en el procedimiento OE-2

(Subsección 5.2.2.1), ya que como se ha podido observar el procedimiento OE-2 generalmente proporciona buenos resultados.

Sea lcm el tiempo de ciclo de las líneas.
Sea k el número de la estación.
Sea h la línea que pertenece al conjunto H.
Sea CTA el conjunto del total de tareas por asignar.
Sea CTC el conjunto de tareas candidatas de todas las líneas.
Sea WL_k la carga de trabajo de la estación k.
Sea t'_{hi} el tiempo de proceso normalizado al lcm de la tarea i de la línea h.
Sea max_time_{Bf} el tiempo de cálculo máximo de la metaheurística HB-SA.
0. Inicializar:
    $k \coloneqq 0$
    CTA conjunto formado por todas las tareas de todas las líneas
    $max\_time_{Bf} \coloneqq$ tiempo predeterminado
1. ***Mientras*** ($CTA \neq \emptyset$) *hacer*
2. Abrir una nueva estación $k \coloneqq k + 1$ y $WL_k \coloneqq 0$
3. Crear CTC. Para todas las tareas i de la línea h y para todas las líneas $h \in H$, una tarea i_h será candidata si sus predecesoras ya han sido asignadas, pertenece a CTA y $t'_{hi} + WL_k \leq lcm$.
4. ***Si*** ($CTC = \emptyset$) *entonces*
5. Ir al Paso 1.
6. ***Para todo*** ($i_h \in CTC$) *hacer*
7. ***Si*** ($t'_{hi} = lcm - WL_k$) *entonces*
8. La tarea i_h se asigna a la estación k.
9. Ir al Paso 21.
10. ***Fin Para***
11. ***Si*** ($aleatorio[0,1) < q_0$) *entonces*
12. Utilizar estrategia I_1 (ec. 20) para seleccionar una tarea i_h de CTC.
13. ***En caso contrario***
14. Utilizar estrategia I_2 (ec. 21) para seleccionar una tarea i_h de CTC.
15. ***Para todo*** ($j_h \in CTC | i_h \neq j_h$) *hacer*
16. ***Si*** ($t'_{hi} + t'_{hj} + WL_k \leq lcm$) *entonces*
17. Ir al Paso 20.
18. ***Fin Para***
19. Seleccionar la tarea i_h de CTC con el tiempo de proceso más grande.
20. Asignar la tarea i_h de CTC a la estación k.
21. ***Si*** (la estación k es multilínea) *entonces*
22. Ejecutar metaheurística HB-SA (max_time_{Bf}) y obtener una solución
23. ***Si*** (la solución aportada por la metaheurística *HB-SA=no factible*) *entonces*
24. La tarea i_h asignada se elimina de la estación k y de CTC.
25. ***Si*** ($CTC = \emptyset$) *entonces*
26. Ir al Paso 2.
27. ***En caso contrario*** ir al Paso 6.
28. Eliminar la tarea asignada i_h de CTA y actualizar: $WL_k \coloneqq WL_k + t'_{hi}$.
29. Ir al Paso 3.
30. ***Fin Mientras***
31. Devuelve la solución S_a encontrada por la hormiga.

Figura 46. Procedimiento OE-ACO utilizado por una hormiga del ACO para la construcción de una solución para el PALBP-B.

En el procedimiento OE-ACO, en cada paso en la construcción de una solución (punto 6 al 10, Figura 46) se comprueba si hay alguna tarea de CTC cuyo tiempo sea igual al tiempo restante de la estación k; de ser así, se asigna a la estación k la tarea que cumpla esta condición. De no cumplirse la condición anterior, se seleccionará una tarea de CTC dependiendo de: 1) el nivel de feromona entre la tarea y la estación k; y 2) la información heurística de cada tarea. En el ACO propuesto, una hormiga a que en la iteración previa ha seleccionado una tarea j_h, en la iteración actual seleccionará una tarea i_h al utilizar uno de los dos tipos de estrategias: explotación (ec. 20) y exploración (ec. 21). La selección de la estrategia se realiza de manera aleatoria y q_0 es un parámetro a calibrar. En la estrategia de explotación, se selecciona la tarea i_h con el valor de I_1 más grande entre las tareas que componen el CTC. En la estrategia de exploración, una tarea i_h de CTC se selecciona con una probabilidad $p(i_h, k)$.

$$I_1 = arg\ max_{i_h \epsilon CTC} \left\{ \left[\tau_{(i_h,k)}\right]^{\alpha} \left[\eta_{(i_h)}\right]^{\beta} \right\} \quad si\ \ aleatorio[0,1) < q_0 \tag{20}$$

$$I_2 = p(i_h, k) = \frac{\left[\tau_{(i_h,k)}\right]^{\alpha} \left[\eta_{(i_h)}\right]^{\beta}}{\sum_{u_h \epsilon CTC} \left[\tau_{(u_h,k)}\right]^{\alpha} \left[\eta_{(u_h)}\right]^{\beta}} \quad si\ \ aleatorio[0,1) \geq q_0 \tag{21}$$

Donde: $\tau_{(i_h,k)}$ es el rastro de feromonas que hay entre la tarea i_h y la estación k; $\eta_{(i_h)}$ es la información heurística relacionada a la tarea i_h (debido a la amplitud del rango de valores que puede adquirir la información heurística $\eta_{(i_h)}$, ésta es linealmente normalizada entre 1 y $card\{CTC\}$); CTC es el conjunto actual de tareas candidatas; y α y β son dos parámetros (a calibrar) que ponderan la importancia del rastro de feromonas y de la información heurística, respectivamente.

La ecuación 22 muestra la composición del rastro de feromonas $\tau_{(i_h,k)}$.

$$\tau_{(i_h,k)} = (1 - w) \cdot \tau_{local(i_h,k)} + w \cdot \tau_{global(i_h,k)} \tag{22}$$

Donde: $\tau_{local(i_h,k)}$ es el rastro de feromonas que hay entre la tarea i_h y la estación k en la matriz de feromonas local de la sub-colonia; $\tau_{global(i_h,k)}$ es el rastro de feromonas que hay entre la tarea i_h y la estación k en la matriz de feromonas global; y $w \in [0,1]$ es un parámetro (a calibrar) que representa la importancia de la feromona global.

Con el objetivo de incrementar al máximo la carga de trabajo de la estación k en curso, en el punto 15 al 18 (Figura 46) se comprueba si en la actual estación k se puede asignar, por tiempo, alguna otra tarea de CTC además de la tarea ya seleccionada (por explotación o exploración). De no ser esto posible, entonces la tarea seleccionada anteriormente (por explotación o exploración) se sustituirá por la tarea de CTC con el tiempo de proceso más grande (punto 19, Figura 46).

Actualización del rastro de feromonas

Inicialmente, la cantidad de feromona en cada par tarea-estación de las matrices locales y global, es un valor aleatorio entre los parámetros (τ_0,τ_F). Esta aleatoriedad inicial permite la diversificación al inicio del procedimiento. Los parámetros τ_0 y τ_F tienen que ser calibrados posteriormente.

Una vez que todas las hormigas de una sub-colonia terminan su recorrido, se deposita una cantidad de feromona en todos los caminos (par tarea-estación) recorridos por las hormigas (ecuación 23-25) y se actualiza la matriz de feromonas local de la sub-colonia. La ecuación 23, actualiza el rastro de feromonas del camino (i_h, k) utilizado por la hormiga a en su recorrido. $E(S_a)$ es el valor de la solución obtenida por la hormiga a. La ecuación 24 representa el incremento del nivel de feromona entre la tarea i_h y la estación k, si esa asignación tarea-estación es utilizada por las hormigas de la sub-colonia. La ecuación 25 representa el valor actualizado del rastro de feromonas para el camino (i_h, k) en la matriz local, donde ρ representa el ratio de evaporación de la feromona (parámetro a calibrar).

$$\Delta\tau^{a}_{local(i_h,k)} = \frac{1}{E(S_a)} \tag{23}$$

$$\Delta\tau_{local(i_h,k)} = \sum_{a=1}^{M} \Delta\tau^{a}_{local(i_h,k)} \tag{24}$$

$$\tau_{local(i_h,k)} = (1-\rho)\cdot\tau_{local(i_h,k)} + \Delta\tau_{local(i_h,k)} \tag{25}$$

Cuando todas las hormigas de todas las sub-colonias han terminado su recorrido (es decir, se ha realizado una iteración), se actualiza la matriz de feromonas global con la mejor solución aportada por las sub-colonias S_{SC} utilizando la ecuación 26.

$$\tau_{global(i_h,k)} = (1-\rho)\cdot\tau_{global(i_h,k)} + \frac{1}{E(S_{SC})} \tag{26}$$

Después de completar una iteración, adicionalmente, se suma una constante de feromonas cte_fer (parámetro a calibrar) a todos los pares tarea-estación, en todas las matrices local y global.

Cada R número de iteraciones (parámetro a calibrar), se reinicia a un valor aleatorio el rastro de feromonas $\tau_{(i_h,k)}$ (donde $\tau_{(i_h,k)} = aleatorio(\tau_0,\tau_F)$) entre todos los pares tarea-estación, en todas las matrices local y global.

Función objetivo

La función objetivo del problema es minimizar, de forma jerárquica, el número de estaciones utilizadas y el tamaño de los buffers, por lo tanto: F.Obj. [MIN] $E(S) = K +$

$Bf \cdot 10^{-6}$; donde: $E(S)$ es el valor de la solución S; K es el número total de estaciones utilizadas; y Bf es la suma del tamaño de todos los buffers.

Criterio de parada

El ACO propuesto finaliza cuando se cumple al menos uno de los tres siguientes criterios de parada: 1) cuando el valor de una solución $E(S)$ es igual a la cota LB_T; 2) cuando se realizan un número de iteraciones $iter$ preestablecidas; o 3) cuando se excede un tiempo de cálculo prestablecido max_time.

5.5.2 Experimento computacional

En este apartado se presenta la calibración de los parámetros del ACO (Subsección 5.5.2.1) y el experimento computacional realizado (Subsección 5.5.2.2).

El ACO desarrollado ha sido codificado y ejecutado en Java SE8, y testeado en un ordenador personal equipado con un Intel Core i7-9700K CPU 3.6 GHz con 32Gb de RAM.

5.5.2.1 Calibración de los parámetros del ACO

Los valores de los parámetros $iter$, a, R, α, β, ρ, w, q_0, τ_0, τ_F y cte_fer del ACO y el parámetro del tiempo max_time_{Bf} de la metaheurística HB-SA (Sección 4.5) han sido calibrados utilizando el algoritmo Nelder and Mead (N&M).

Los parámetros del ACO se han calibrado usando los mismos 160 ejemplares utilizados para calibrar los parámetros del SA (Sección 5.4). Estos ejemplares son resueltos fijando el tiempo de cálculo máximo del ACO $max\_time = infinito$.

Del mismo modo que en el SA (Sección 5.4), se han realizado dos experimentos de calibración diferentes. En el primer experimento, los parámetros del ACO se han calibrado, exclusivamente, para cada uno de los grupos de ejemplares (G1, G2, G3 y G4). En el segundo experimento, los parámetros del ACO se han calibrado utilizando los mismos 40 ejemplares (10 ejemplares de cada grupo) que en el SA calibración conjunta.

Los valores obtenidos para los parámetros del "ACO calibración individual" en el primer experimento son los siguientes:
Grupo 1 (G1): $iter = 41$; $a = 5$; $R = 33$; $\alpha = 5.54$; $\beta = 8.25$; $\rho = 0.612$; $w = 0.50$; $q_0 = 0.48$; $\tau_0 = 2.1$; $\tau_F = 3.8$; $cte\_fer = 0.54$; $max\_time_{Bf} = 0.005$.

Grupo 2 (G2): $iter = 51$; $a = 6$; $R = 30$; $\alpha = 7.94$; $\beta = 12.37$; $\rho = 0.63$; $w = 0.53$; $q_0 = 0.51$; $\tau_0 = 1.1$; $\tau_F = 3.4$; $cte\_fer = 0.45$; $max\_time_{Bf} = 0.004$.

Grupo 3 (G3): $iter = 68$; $a = 3$; $R = 27$; $\alpha = 10.68$; $\beta = 15.93$; $\rho = 0.61$; $w = 0.6$; $q_0 = 0.54$; $\tau_0 = 1$; $\tau_F = 3$; $cte\_fer = 0.4$; $max\_time_{Bf} = 0.002$.

Grupo 4 (G4): $iter = 8$; $a = 2$; $R = 8$; $\alpha = 13.2$; $\beta = 18.11$; $\rho = 0.81$; $w = 0.73$; $q_0 = 0.63$; $\tau_0 = 1.2$; $\tau_F = 4.7$; $cte\_fer = 0.42$; $max\_time_{Bf} = 0.04$.

Los valores obtenidos para los parámetros del "ACO calibración conjunta" en el segundo experimento son los siguientes:
Grupo todos: $iter = 6$; $a = 3$; $R = 6$; $\alpha = 11.56$; $\beta = 17.03$; $\rho = 0.79$; $w = 0.62$ $q_0 = 0.55$; $\tau_0 = 2.3$; $\tau_F = 4.7$; $cte\_fer = 0.17$; $max\_time_{Bf} = 0.028$.

5.5.2.2 Experimento computacional

Para evaluar el rendimiento del ACO propuesto, se ha realizado el mismo experimento computacional que en la Subsección 5.4.2.2. El experimento ha sido ejecutado 5 veces de manera independiente y el promedio de los resultados $\overline{K}$, $\overline{Bf}$ y tiempo $\overline{CPU}$ obtenidos para cada ejemplar, se recogen en el Anexo C.4 (documento externo "Anexos").

La Tabla 26 (que presenta la misma información que la Tabla 24), muestra que la calibración individual de los parámetros del ACO permite obtener resultados ligeramente mejores que el ACO calibración conjunta, tanto en número de estaciones $\overline{K}$ como en el tiempo de cálculo $\overline{CPU}$. Además, tanto la calibración individual como la conjunta, obtienen mejores resultados del número de estaciones $\overline{K}$ que los conseguidos por el SA y el CH-5. Aunque el SA, en general, necesita un menor tiempo de cálculo promedio.

Tabla 26. Resumen y comparación de los resultados obtenidos por el ACO respecto a la cota LB_T, la heurística CH-5 con $max\_time_{Bf} = inf$ y el SA.

Procedimiento		Resultado	G1	G2	G3	G4	Total
$\mathbf{LB_T}$		$\overline{\mathbf{K}}$	8.267	18.650	35.717	366.483	107.279
CH-5 (inf)		$\overline{\mathbf{K}}$	8.317	18.800	36.067	368.200	107.846
		$\overline{\mathbf{Bf}}$	0.033	0.067	0.017	5.383	1.375
		$\overline{\mathbf{CPU}}$	18.193	154.033	276.369	7261.480	1927.519
SA	Calibración individual	$\overline{\mathbf{K}}$	8.300	18.770	36.067	368.187	107.831
		$\overline{\mathbf{Bf}}$	0.000	0.000	0.040	5.653	1.423
		$\overline{\mathbf{CPU}}$	0.447	5.854	11.706	1843.729	465.434
	Calibración conjunta	$\overline{\mathbf{K}}$	8.310	18.800	36.070	368.223	107.851
		$\overline{\mathbf{Bf}}$	0.033	0.060	0.070	4.587	1.188
		$\overline{\mathbf{CPU}}$	0.752	2.602	6.803	1922.366	483.131
ACO	Calibración individual	$\overline{\mathbf{K}}$	8.300	18.727	35.967	368.030	107.756
		$\overline{\mathbf{Bf}}$	0.000	0.073	0.000	5.427	1.375
		$\overline{\mathbf{CPU}}$	0.848	13.324	32.466	2587.670	658.577
	Calibración conjunta	$\overline{\mathbf{K}}$	8.303	18.750	36.023	368.073	107.788
		$\overline{\mathbf{Bf}}$	0.043	0.130	0.003	5.783	1.490
		$\overline{\mathbf{CPU}}$	1.472	4.666	14.765	3093.652	778.639

En la Tabla 27, que muestra la misma información que la Tabla 25, se observa que el ACO "calibración individual" obtiene un mayor número de soluciones óptimas confirmadas (en el 70% de ejemplares) que el ACO "calibración conjunta" (en el 67.9% de ejemplares). Además, se observa que el ACO "calibración individual" es el procedimiento, de todos los diseñados, que obtiene un menor valor DMR, seguido del ACO "calibración conjunta". El ACO "calibración individual" mejora en un 15.5% el valor DMR obtenido por el ACO "calibración conjunta", en un 20.4% el valor DMR obtenido por el SA "calibración individual", y en un 32.6% el valor DMR obtenido por CH-5. Se observa que los resultados (de la Tabla 26 y Tabla 27) obtenidos por el ACO y el SA, ambos con "calibración individual", para el grupo de ejemplares G1 (ejemplares de 40 tareas), son los mismos. Sin embargo, el mejor rendimiento del ACO se hace patente al aumentar el tamaño de los ejemplares a resolver, llegando a una mejora en DMR de aproximadamente el 30% respecto al SA para el grupo G2 y G3, y de un 6.8% para el grupo G4.

Tabla 27. Número de soluciones óptimas confirmadas y DMR obtenidos por el ACO respecto a la cota LB_T, y comparación de los resultados respecto a la heurística CH-5 con $max\_time_{Bf} = inf$ y el SA.

Procedimiento		Resultado	G1	G2	G3	G4	Total
CH-5 (inf)		N.º soluciones óptimas	55	49	40	6	150
		DMR %	0.682	0.840	0.905	0.442	0.717
SA	Calibración individual	N.º soluciones óptimas	58	53	46	5	162
		DMR %	0.444	0.642	0.905	0.439	0.607
	Calibración conjunta	N.º soluciones óptimas	58	51	40	6	155
		DMR %	0.587	0.840	0.914	0.450	0.698
ACO	Calibración individual	N.º soluciones óptimas	58	55	46	9	168
		DMR %	0.444	0.448	0.641	0.398	0.483
	Calibración conjunta	N.º soluciones óptimas	57	54	45	7	163
		DMR %	0.492	0.586	0.800	0.409	0.572

De los resultados obtenidos se puede establecer que la metaheurística ACO, con una calibración individual de los parámetros, es el procedimiento que mejor rendimiento obtiene de entre todos los desarrollados, considerando el valor de $\overline{K}$ y DMR. Aunque el SA en general obtiene un menor tiempo $\overline{CPU}$. Se recuerda, que la función objetivo establecida es una función multiobjetivo jerárquica, en la que el objetivo principal es minimizar el número de estaciones utilizadas y se desempata con el tamaño total de los buffers.

5.6 Conclusiones

En este capítulo se han presentado varios procedimientos no exactos para resolver el problema PALB-B. El objetivo de los procedimientos desarrollados es el de minimizar, de forma jerárquica, el número de estaciones utilizadas y el tamaño de los buffers. Para el objetivo secundario, (sub)problema de dimensionado de buffers en las estaciones multilínea, se ha utilizado la metaheurística HB-SA basada en SA (Sección 4.5).

De las 48 heurísticas greedy de un solo paso desarrolladas (Subsección 5.2.2), se observa que los procedimientos orientados a estaciones (OE) obtienen, de manera global, mejores soluciones que los orientados a tareas (OT). Siendo las heurísticas denominadas OE-2-3 y OE-2-5 las que aportan mejores soluciones. Por otra parte, también se visualiza que un tiempo max_time_{Bf} elevado, no se traduce generalmente en mejores soluciones. En la mayoría de los casos las soluciones no tienden a mejorar a partir de valores de max_time_{Bf} de entre 0.1 y 0.5 segundos. Inclusive, en algunos casos, las soluciones empeoran debido al carácter aleatorio de la metaheurística HB-SA.

De forma similar, se observa que los cócteles de heurísticas basados en procedimientos OE, obtienen mejores soluciones que los basados en procedimientos OT, en especial el cóctel denominado CH-7. Además, del mismo modo que ocurre en las heurísticas greedy, un tiempo max_time_{Bf} elevado no siempre mejora sustancialmente la calidad de las soluciones. Generalmente, en los CH, un tiempo max_time_{Bf} entre 0.05 y 0.1 segundos es suficiente para aportar soluciones de calidad sin llegar a incrementar drásticamente el tiempo de cálculo.

A raíz de los resultados obtenidos por las heurísticas greedy y los procedimientos CH, parece necesario ajustar, mediante calibración, el valor del tiempo max_time_{Bf}.

El desarrollo y posterior análisis de los resultados de las metaheurísticas SA y ACO demuestran, como era de esperar, que estos procedimientos mejoran los resultados obtenidos por los demás procedimientos presentados. Siendo el ACO el procedimiento, de todos los presentados, que obtiene mejores soluciones. De los resultados se constata que, al calibrar los parámetros de las metaheurísticas exclusivamente para cada uno de los grupos de ejemplares (se recuerda que los ejemplares se agrupan según su tamaño, resultando en 4 grupos), se obtienen mejores soluciones que al calibrar los parámetros considerando todos los grupos juntos. Además, se observa que en ningún caso, al calibrar el parámetro max_time_{Bf}, su valor supera los 0.04 segundos. Un resultado que

permite reforzar lo dicho anteriormente, que para obtener buenas soluciones no parece necesario un valor grande de max_time_{Bf}.

Para finalizar, se hace mención que se está preparando una publicación con los procedimientos heurísticos y metaheurísticos presentados en este capítulo: ***Heuristic and metaheuristic procedures for solving the parallel assembly lines balancing problem with multi-line stations and different cycle times (en preparación/enviado).***

Capítulo 6. Conclusiones finales, futuras vías de investigación y contribuciones

6.1 Conclusiones

Tras una revisión exhaustiva de la literatura del PALBP, se pudo observar que una parte importante de los trabajos presentados consideran en sus procedimientos de resolución la posibilidad que las líneas de montaje trabajen con tiempos de ciclo diferentes. En estos casos, para poder resolver el problema, éste se reduce al caso de trabajar con el mismo tiempo de ciclo para todas las líneas. Los procedimientos utilizados para este fin introducen la producción en lotes en las líneas. De la revisión de la literatura se constata que en ningún trabajo presentado se manifiesta el uso de buffers en el caso de trabajar con lotes. Mediante un ejemplo ilustrativo se ha demostrado que no considerar la existencia y el dimensionado de buffers en las estaciones multilínea, puede hacer que una solución académica del PALBP no pueda ser aplicada en la industria.

En esta tesis doctoral se ha introducido, definido y formalizado, el problema de dimensionado de buffers en los problemas PALB con estaciones multilínea y líneas con tiempos de ciclo diferentes. Este nuevo problema de estudio se ha nombrado como PALBP-B.

Una de las premisas del PALBP-B es que las unidades de producto fluyen de una estación a otra cada tiempo de ciclo de la línea y de una en una. Lo que significa que, si hay más de una unidad de producto en un puesto de trabajo, el resto de las unidades deberán depositarse en un buffer.

El problema abordado en esta tesis se compone de dos (sub)problemas que idealmente se deberían de resolver conjuntamente: 1) (sub)problema de equilibrado de líneas (definiendo las estaciones regulares y multilínea); 2) (sub)problema de dimensionado de buffers en una estación multilínea. La resolución simultánea de ambos (sub)problemas implica un gran nivel de dificultad, ya que es sabido que el PALBP es de naturaleza NP-difícil.

La metodología de investigación se ha dividido en dos etapas.

En la primera etapa se define y formaliza el (sub)problema de dimensionado de buffers. Los procedimientos presentados están diseñados para resolver el (sub)problema de: dado un conjunto de tareas en una estación multilínea, secuenciar y programar las

tareas de la estación con el objetivo de minimizar el tamaño total de los buffers en ambas líneas.

Para la formalización y resolución óptima del (sub)problema de dimensionado de buffers, se presenta un modelo de PLEM (conocido como modelo V1), a raíz del cual se ha desarrollado una variante (modelo V2) y una modificación en un parámetro del software de resolución para el modelo V1 (conocido como modelo V3). El experimento computacional realizado muestra que el modelo V2 (el cual considera la misma secuencia de tareas en todas las unidades que se han de procesar en una estación en un tiempo de ciclo lcm) es el que obtiene un mejor rendimiento, tanto en calidad de soluciones como en tiempo de cálculo. Sin embargo, como se demostró, V2 no siempre garantiza la obtención de una solución óptima de un ejemplar. No obstante, se observa que en varios ejemplares del experimento computacional los modelos desarrollados no han sido capaces de obtener una solución factible dentro de un tiempo máximo de cálculo. A raíz de un segundo experimento se demuestra que ir variando el valor de la variable a_h (de cada línea h) de cada ejemplar hasta conseguir una solución óptima, no parece ser una buena estrategia. Porque, aunque en varios ejemplares se consigue una solución en un tiempo de cálculo relativamente pequeño, en otros se sigue necesitando un tiempo de cálculo de más de 50 segundos.

Las heurísticas HB-1A y HB-1B, y el cóctel de heurísticas HB-1AB, muestran que obtienen un mayor número de soluciones factibles respecto a los modelos de PLEM y necesitan un tiempo de cálculo mucho menor. Siendo el tiempo promedio de cálculo para las heurísticas de 0.4 milisegundos y para HB-1AB de 0.7 milisegundos, en comparación a los 144 segundos de V2. Sin embargo, V2 obtiene en promedio unas soluciones un 40% mejor que HB-1A y que HB-1AB.

En este contexto y con el objetivo de mejorar la calidad de las soluciones obtenidas, se ha desarrollado una metaheurística basada en SA (HB-SA). Se observa que HB-SA es el procedimiento que obtiene un mayor número de soluciones factibles, con una calidad de las soluciones bastante cerca a los conseguidos por los modelos V1 y V3, pero V2 obtiene un promedio de soluciones un 32% mejor. Sin embargo, el tiempo de cálculo promedio obtenido por HB-SA es un 98% menor que el tiempo promedio de V2. Además, un segundo experimento, manifiesta que limitar el tiempo de cálculo de HB-SA no afecta, en gran medida, a la calidad de las soluciones obtenidas. Donde, con un tiempo de cálculo máximo entre 0.5 y 1 (segundos) se puede obtener una calidad de soluciones cercanas a las obtenidas al no limitar el tiempo de cálculo, con una reducción de tiempo promedio de cálculo entre un 96% y 92%, respectivamente.

Después del análisis de los resultados de los procedimientos propuestos, todo parece indicar que HB-SA es el procedimiento que mantiene un mejor compromiso entre número de soluciones factibles, calidad de las soluciones y tiempo de cálculo. Aunque, V2 es el procedimiento que obtiene un mejor valor promedio de las soluciones, el tiempo de cálculo requerido parece inviable dentro de un contexto industrial. En esta parte, los procedimientos no exactos y en especial HB-SA, parecen ser de mayor aplicabilidad y en el caso del último, ofrece una buena calidad de soluciones en tiempos relativamente pequeños.

En la segunda etapa, se han desarrollado diferentes procedimientos no exactos, los cuales utilizan el procedimiento HB-SA para resolver el (sub)problema de dimensionado de buffers en las estaciones multilínea. Los procedimientos desarrollados tienen una función multiobjetivo, minimizar el número de estaciones utilizadas y el tamaño total de los buffers. La segunda parte de la función objetivo, minimizar el tamaño total de los buffers, se utiliza para desempatar entre dos soluciones.

El desarrollo de 48 heurísticas greedy de un solo paso, basadas cada una de ellas en una regla de prioridad de la literatura, permite observar que los mejores resultados son obtenidos por heurísticas basadas en procedimientos orientados a estaciones (OE) y no en las basadas en procedimientos orientado a tareas (OT). Además, los resultados del experimento computacional permiten constatar que, generalmente, la calidad de las soluciones no tiende a mejorar significativamente al incrementar el tiempo de cálculo máximo permitido del procedimiento HB-SA (max_time_{Bf}). Incluso, en algunos casos, se observa que un mayor tiempo max_time_{Bf} tiende a empeorar las soluciones obtenidas, debido a la naturaleza estocástica de HB-SA.

El diseño de 7 cócteles de heurísticas basados en varias de las 48 heurísticas greedy, permite constatar que los mejores resultados son obtenidos por los cócteles de heurísticas basados principalmente en procedimientos OE. Además, como era de esperar, las soluciones no tienden a mejorar significativamente a un mayor tiempo max_time_{Bf}. Estando comprendido este valor max_time_{Bf} entre 0.05 y 0.1 segundos. Por tanto, parece necesario tener en cuenta el valor de max_time_{Bf}, para obtener una buena relación entre el tiempo CPU y el valor de la solución.

El desarrollado dos metaheurísticas, SA y ACO, permite mejorar los resultados obtenidos por los demás procedimientos presentados. Siendo el ACO el procedimiento, de todos los presentados, que obtiene mejores soluciones. De los resultados se verifica que, al calibrar los parámetros de las metaheurísticas exclusivamente para cada uno de los grupos de ejemplares (calibración individual), se obtienen mejores soluciones que al

calibrar los parámetros considerando todos los grupos juntos (calibración conjunta). De los resultados se constata que el ACO con calibración individual, es el procedimiento que obtiene mejores soluciones, mejorando en un 15.56% el valor de DMR obtenido por el ACO “calibración conjunta” (segundo mejor procedimiento), y en un 20.4% el valor de DMR obtenido por el SA “calibración individual” (tercer mejor procedimiento).

Por último, se observa que en ambas metaheurísticas el valor de max_time_{Bf}, el cual es un parámetro a calibrar, en ningún caso supera los 0.04 segundos. Lo que demuestra, una vez más, que no es necesario tiempos de max_time_{Bf} muy grandes.

Finalmente, se puede concluir que, se han logrado todos los objetivos propuestos en esta tesis. A continuación, se detallan los objetivos logrados:

1. Un actualizado y extenso análisis del estado del arte del PALBP, exponiendo diferentes vías de investigación en el campo.
2. Se ha demostrado la necesidad del uso de buffers en las estaciones multilínea cuando los sistemas PALB trabajan con líneas con tiempos de ciclo diferentes.
3. Se ha presentado y definido el PALBP-B.
4. Definición y formalización del (sub)problema de dimensionado de buffers en una estación multilínea. Desarrollo de un modelo de PLEM y diferentes variantes, para su resolución exacta. Propuesta de dos heurísticas, un cóctel de heurísticas y una metaheurística para la resolución aproximada del (sub)problema de dimensionado de buffers.
5. Generación de una serie de ejemplares para el (sub)problema de dimensionado de buffers.
6. Realización de varios experimentos computacionales para evaluar el rendimiento de los métodos propuestos para el (sub)problema de dimensionado de buffers en una estación multilínea.
7. Desarrollo de 48 heurísticas greedy, 7 cócteles de heurísticas, una metaheurística basada en SA y otra basada en ACO para resolver el problema PALB-B.
8. Generación de una serie de ejemplares para el PALBP-B.
9. Realización de varios experimentos computacionales para evaluar el rendimiento de los métodos propuestos para PALBP-B.

6.2 Futuras vías de investigación

El PALBP-B, es un problema que se puede presentar en la mayoría de las variantes del PALBP presentes en la literatura. Como se ha podido constatar en el estado del arte del PALBP, el trabajar con líneas con tiempos de ciclo diferentes no solo se contempla para el caso del PALB de líneas rectas y modelo único. Existen otras variantes como el PALBP de líneas de dos lados y en forma de U, donde también se considera el caso de trabajar con líneas con tiempos de ciclo diferentes.

A continuación se detallan las futuras vías de investigación abiertas a raíz de la realización de la presente tesis:

Procedimientos no exactos para resolver el (sub)problema de dimensionado de buffers.

Nuevas heurísticas pueden ser desarrolladas para obtener un mayor número de soluciones factibles del problema. También, se pueden desarrollar y comparar el rendimiento de diferentes metaheurísticas como podrían ser: GA, ACO, TS, etc.

Un punto interesante puede ser la combinación del modelo de PLEM y una heurística/metaheurística, trabajando ambos métodos de manera colectiva, lo que resultaría en una metaheurística. Por ejemplo, primero ejecutando una metaheurística durante un breve periodo de tiempo con el objetivo de obtener una solución factible buena. Posteriormente, utilizando la solución factible aportada por la metaheurística como una cota superior en el modelo de PLEM.

Procedimientos no exactos para resolver el PALBP-B.

El desarrollo de procedimientos heurísticos greedy, con reglas de prioridad combinadas, puede ser un punto interesante para obtener soluciones factibles de mayor calidad. Por ejemplo, Martino and Pastor (2010) y Otto and Otto (2014), presentan la combinación de varias reglas de prioridad elementales, es decir, aquella reglas que se basan en un único atributo de la tarea. En las reglas de prioridad combinadas usualmente se pondera el peso de cada regla elemental. El peso de cada regla se puede calibrar utilizando el procedimiento EAGH (*Empirically Adjusted Greedy Heuristics*) (Corominas , 2005), como en Martino and Pastor (2010). Otto and Otto (2014) recomiendan utilizar reglas elementales con diferentes tipos de información (p.ej. relación de precedencia y tiempos de procesos). Los resultados expuestos en ambos trabajos demuestran que las reglas de prioridad combinadas ayudan a obtener mejores soluciones. Las soluciones así obtenidas pueden ser utilizadas, posteriormente, como una solución inicial en diferentes metaheurísticas, como en el SA desarrollado en esta tesis.

Por otra parte, queda abierto el desarrollo de otros procedimientos metaheurísticos como GA, TS, BA, etc., o hiperheurísticas, para resolver el PALBP-B. Los procedimientos anteriormente mencionados ya han sido utilizados con éxito en el PALBP. En el caso de las hiperheurísticas solo han sido utilizadas en Özbakır and Seçme (2020). Las hiperheurísticas son métodos de búsqueda o mecanismo de aprendizaje para seleccionar o generar heurísticas para resolver problemas de búsqueda computacionalmente difíciles (Burke et al., 2010). Una forma de utilizar una hiperheurística para resolver el PALBP-B sería utilizar un enfoque similar al propuesto por Özbakır and Seçme (2020). Los autores proponen que la selección de una tarea del conjunto de tareas candidatas (CTC; usualmente, una tarea es candidata si: la tarea no se ha asignado a ninguna estación; y sus predecesoras ya han sido asignadas) se realice de acuerdo con una regla de prioridad. De esta forma se va generando una secuencia de tareas y paralelamente se genera una secuencia de heurísticas del mismo tamaño. En cada posición en la secuencia de heurísticas se guarda la regla de prioridad utilizada para seleccionar una tarea de CTC. Al final, cada posición de la secuencia de tareas tiene asociada una regla de prioridad, que ocupa la misma posición, pero en la secuencia de heurísticas. La generación de diferentes soluciones se realiza al cambiar la posición de dos o más reglas de prioridad en la secuencia de heurísticas, y posteriormente construir una nueva solución al ir asignando las tareas de acuerdo con la nueva secuencia de heurísticas.

Como se ha observado, la calibración de los parámetros de las metaheurísticas SA y ACO juega un papel importante a la hora de obtener mejores soluciones, ya que afectan al comportamiento del algoritmo. Un aspecto importante en el diseño de los algoritmos (como en el SA y el ACO) es evitar que converjan rápido y queden atrapados en un óptimo local. El control de ciertos parámetros puede ayudar a evitar o ralentizar la convergencia total del algoritmo a través de la variación dinámica de ciertos parámetros en función del estado de la búsqueda. Por ejemplo, una forma de evitar o reducir la convergencia rápida del algoritmo ACO, es la incorporación de parámetros dinámicos para mejorar el equilibrio entre exploración y explotación, de esta forma evitar la convergencia prematura y que el algoritmo deje regiones del espacio de búsqueda sin explorar. El uso de operadores dinámicos es un punto interesante, ya que pueden ser aplicados en varios algoritmos como los presentados en esta tesis.

Considerar otras características del problema.

El problema presentado en esta tesis se centra en el PALBP-B más básico, en términos de restricciones y características de la línea de montaje. Este problema puede ser

extendido a diferentes tipos de *layout*, ya presentados en la literatura del PALBP, como pueden ser líneas de dos lados y líneas en forma de U. Además, el problema puede ser extendido a diferentes tipos de línea: líneas mixtas y multi-modelo. En este caso, sería necesario adaptar el (sub)problema de dimensionado de buffers al problema de líneas mixtas y multi-modelo, considerando la secuenciación de los modelos en las líneas. Así mismo, parece interesante incrementar la practicidad del problema y considerar características que pueden surgir en casos reales: tiempos de proceso estocásticos, operarios con diferentes niveles de habilidades, tiempo de desplazamientos entre líneas (en el caso de las estaciones multilínea), tiempos de *setup*, líneas paralelas mixtas (conjunto de diferentes tipos de líneas, por ejemplo: una línea modelo único y una línea modelos mixtos). Otras funciones objetivo, como minimizar el tiempo de ciclo o el coste (considerando estaciones, operarios y buffers) del sistema, pueden ser consideradas.

6.3 Contribuciones

A continuación se presentan los trabajos derivados, hasta el momento, de esta tesis doctoral:

1. Aguilar, H., García-Villoria, A., & Pastor, R. (2020). A survey of the parallel assembly lines balancing problem. Computers & Operations Research, 124, 105061. https://doi.org/10.1016/j.cor.2020.105061.
2. Pastor, R., Aguilar, H., & García-Villoria, A. (2021). Existence and sizing of buffers in parallel assembly lines with multi-line workstations and different cycle times. DYNA MANAGEMENT, 9(1), [10 p.]-[10 p.]. https://doi.org/10.6036/MN10070.
3. Aguilar, H., García-Villoria, A., & Pastor, R. (2021). Mathematical models for buffer sizing problems in parallel assembly lines with multi-lines stations and different cycle times. DYNA, 96(6), 563–563. https://doi.org/10.6036/10223.
4. Heuristic and metaheuristic procedures for buffer sizing problems in parallel assembly lines with multi-line stations and different cycle times (en preparación/enviado).
5. Heuristic and metaheuristic procedures for solving the parallel assembly lines balancing problem with multi-line stations and different cycle times (en preparación/enviado).

Referencias

Ağpak, K., & Zolfaghari, S. (2015). Mathematical models for parallel two-sided assembly line balancing problems and extensions. *International Journal of Production Research*, *53*(4), 1242–1254. https://doi.org/10.1080/00207543.2014.955218

Aguilar, H., García-Villoria, A., & Pastor, R. (2020). A survey of the parallel assembly lines balancing problem. *Computers & Operations Research*, *124*, 105061. https://doi.org/10.1016/j.cor.2020.105061

Aguilar, H., García-Villoria, A., & Pastor, R. (2021). Mathematical models for buffer sizing problems in parallel assembly lines with multi-line stations and different cycle times. *DYNA*, *96*(6), 563–563. https://doi.org/10.6036/10223

Andrés, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, *187*(3), 1212–1223. https://doi.org/10.1016/j.ejor.2006.07.044

Araújo, F. F. B., Costa, A. M., & Miralles, C. (2015). Balancing parallel assembly lines with disabled workers. *European J. of Industrial Engineering*, *9*(3), 344–365. https://doi.org/10.1504/EJIE.2015.069343

Bartholdi, J. J. (1993). Balancing two-sided assembly lines: a case study. *International Journal of Production Research*, *31*(10), 2447–2461. https://doi.org/10.1080/00207549308956868

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, *142*(2), 259–277. https://doi.org/10.1016/j.ijpe.2012.10.020

Bautista, J., & Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, *177*(3), 2016–2032. https://doi.org/10.1016/j.ejor.2005.12.017

Baybars, I. (1986). Survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, *32*(8), 909–932. https://doi.org/10.1287/mnsc.32.8.909

Baykasoğlu, A., Özbakir, L., Görkemli, L., & Görkemli, B. (2009). Balancing Parallel Assembly Lines via Ant Colony Optimization. *2009 International Conference on Computers and Industrial Engineering*, 506–511. https://doi.org/10.1109/ICCIE.2009.5223867

Baykasoğlu, A., Özbakir, L., Görkemli, L., & Görkemli, B. (2012). Multi-colony ant algorithm for parallel assembly line balancing with fuzzy parameters. *Journal of Intelligent and Fuzzy Systems*, *23*(6), 283–295. https://doi.org/10.3233/IFS-2012-0520

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, *168*(3), 694–715. https://doi.org/10.1016/j.ejor.2004.07.023

Benzer, R., Gökçen, H., Çetinyokus, T., & Çerçioglu, H. (2007). A Network Model for Parallel Line Balancing Problem. *Mathematical Problems in Engineering*, *2007*, 1–12. https://doi.org/10.1155/2007/10106

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, *183*(2), 674–693. https://doi.org/10.1016/j.ejor.2006.10.010

Boysen, N., Fliedner, M., & Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics*, *111*(2), 509–528. https://doi.org/10.1016/j.ijpe.2007.02.026

Boysen, N., Schulze, P., & Scholl, A. (2021). Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2021.11.043

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A Classification of Hyper-heuristic Approaches BT - Handbook of Metaheuristics. In M. Gendreau & J.-Y. Potvin (Eds.) (pp. 449–468). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4419-1665-5_15

Buxey, G. M. (1974). Assembly Line Balancing with Multiple Stations. *Management Science*, *20*(6), 1010–1021. https://doi.org/10.1287/mnsc.20.6.1010

Calleja, G., Corominas, A., García-Villoria, A., & Pastor, R. (2013). A MILP model for the Accessibility Windows Assembly Line Balancing Problem (AWALBP). *International Journal of Production Research*, *51*(12), 3549–3560. https://doi.org/10.1080/00207543.2012.751514

Capacho, L., Pastor, R., Dolgui, A., & Guschinskaya, O. (2009). An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*, *15*(2), 109–132. https://doi.org/10.1007/s10732-007-9063-x

Çerçioğlu, H., Özcan, U., Gökçen, H., & Toklu, B. (2009). A Simulated Annealing Approach for Parallel Assembly Line Balancing Problem. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *24*(2), 331–341.

Chutima, P., & Jirachai, P. (2020). Parallel U-shaped assembly line balancing with adaptive MOEA/D hybridized with BBO. *Journal of Industrial and Production Engineering*, *37*(2–3), 97–119. https://doi.org/10.1080/21681015.2020.1735544

Chutima, P., & Yothaboriban, N. (2017). Multi-objective mixed-model parallel assembly line balancing with a fuzzy adaptive biogeography-based algorithm. *International Journal of Industrial and System Engineering*, *26*(1), 90–132. https://doi.org/10.1504/IJISE.2017.083182

Çil, Z. A., Mete, S., Özceylan, E., & Ağpak, K. (2017). A Beam Search Approach for Solving Type II Robotic Parallel Assembly Line Balancing Problem, *61*, 129–138. https://doi.org/10.1016/j.asoc.2017.07.062

Corominas, A., García-Villoria, A., & Pastor, R. (2013). Technical note: A systematic procedure based on CALIBRA and the Nelder & Mead algorithm for fine-tuning metaheuristics. *Journal of the Operational Research Society*, *64*(2), 276–282. https://doi.org/10.1057/jors.2012.51

Corominas, A. (2005). Empirically Adjusted Greedy Algorithms (EAGH): A new approach to solving combinatorial optimisation problems. *Working Paper IOC-DT-P-2005-22*, Universitat Politècnica de Catalunya, Spain. https://doi.org/Available at http://hdl.handle.net/2117/510

Corominas, A., Ferrer, L., & Pastor, R. (2011). Assembly line balancing: General resource-constrained case. *International Journal of Production Research*, *49*(12), 3527–3542. https://doi.org/10.1080/00207543.2010.481294

Doringo, M. (1992). *Optimization, learning and natural algorithms*. *Ph.D. Thesis, Politecnico di Milano*. Retrieved from http://ci.nii.ac.jp/naid/10027800670/en/

Dowsland, K. A., & Adenso-Díaz, B. (2003). Heuristic design and fundamentals of the Simulated Annealing. *Inteligencia Artificial.*, *7*(19), 93–102.

Erel, E., & Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, *9*(5), 414–434. https://doi.org/10.1080/095372898233902

Esmaeilian, G. R., Ghanbarian, F., & Hamedi, M. (2015). Multi-objective balancing of Parallel Mixed-Model assembly lines by Considering Machine and Worker Constraints. *Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management*, 2021–2028.

Esmaeilian, G. R., Ismail, N., Sulaiman, S., Hamedi, M., & Ahmad, M. M. H. M. (2009). Allocating and Balancing of Mixed Model Production Through the Parallel Assembly Lines. *European Journal of Scientific Research*, *31*(4), 616–631.

Esmaeilian, G. R., Sulaiman, S., Ismail, N., Hamedi, M., & Ahmad, M. M. H. M. (2011). A tabu search approach for mixed-model parallel assembly line balancing problem (type II). *International Journal of Industrial and Systems Engineering*, *8*(4), 407–431. https://doi.org/10.1504/IJISE.2011.041803

Faaland, B. H., Klastorin, T. D., Schmitt, T. G., & Shtub, A. (1992). Assembly Line Balancing with Resource Dependent Task Times. *Decision Sciences*, *23*(2), 343–364. https://doi.org/10.1111/j.1540-5915.1992.tb00393.x

García-Villoria, A., Corominas, A., & Pastor, R. (2012). Cóctel de heurísticas para resolver problemas difíciles. *DYNA INGENIERIA E INDUSTRIA*, *87*(3), 275–278. https://doi.org/10.6036/4508

García-Villoria, A., Corominas, A., & Pastor, R. (2010). Solving the response time variability problem by means of the cross-entropy method. *International Journal of Manufacturing Technology and Management*, *20*(1–4), 316–330. https://doi.org/10.1504/ijmtm.2010.032904

García-Villoria, A., & Pastor, R. (2009). Introducing dynamic diversity into a discrete particle swarm optimization. *Computers and Operations Research*, *36*(3), 951–966. https://doi.org/10.1016/j.cor.2007.12.001

Gen, M., & Cheng, R. (2000). *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons.

Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of Metaheuristics: International Series in Operations research e Management Science*.

Ghosh, S., & Gagnon, R. J. (1989). A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, *27*(4), 37–41. https://doi.org/10.1080/00207548908942574

Gökçen, H., Ağpak, K., & Benzer, R. (2006). Balancing of parallel assembly lines. *International Journal of Production Economics*, *103*(2), 600–609. https://doi.org/10.1016/j.ijpe.2005.12.001

Grangeon, N., & Norre, S. (2012). Extending metaheuristics based on bin packing for SALBP to PALBP. *European J. of Industrial Engineering*, *6*(6), 713–732.

Guo, Q., & Tang, L. (2009). A scatter search based heuristic for the balancing of parallel assembly lines. *Proceedings of the 48h IEEE Conference on Decision and Control*

(CDC) Held Jointly with 2009 28th Chinese Control Conference, 6256–6261. https://doi.org/10.1109/CDC.2009.5399991

Hackman, S. T., Magazine, M. J., & Wee, T. S. (1989). Fast, Effective Algorithms for Simple Assembly Line Balancing Problems, *37*(6), 916–924. Retrieved from https://www.jstor.org/stable/171473

Hamzadayi, A., & Yildiz, G. (2013). A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. *Computers and Industrial Engineering*, *66*(4), 1070–1084. https://doi.org/10.1016/j.cie.2013.08.008

Helgeson, W. B., Salveson, M. E., & Smith, W. W. (1954). How to balance an assembly line. *Management Report*, *7*.

Hwang, R. K., Katayama, H., & Gen, M. (2008). U-shaped assembly line balancing problem with genetic algorithm. *International Journal of Production Research*, *46*(16), 4637–4649. https://doi.org/10.1080/00207540701247906

Kara, Y., & Atasagun, Y. (2013). *Assembly Line Balancing with Resource Dependent Task Times: An Application to Parallel Assembly Lines*. *IFAC Proceedings Volumes* (Vol. 46). Saint Petersburg: IFAC. https://doi.org/https://doi.org/10.3182/20130619-3-RU-3018.00333

Kara, Y., Gökçen, H., & Atasagun, Y. (2010). Balancing parallel assembly lines with precise and fuzzy goals. *International Journal of Production Research*, *48*(6), 1685–1703. https://doi.org/10.1080/00207540802534715

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kucukkoc, I., & Zhang, D. Z. (2013). *Balancing parallel two-sided assembly lines via a genetic algorithm based approach*. *43rd International Conference on Computers and Industrial Engineering (CIE43)*.

Kucukkoc, I., & Zhang, D. Z. (2014a). An Agent Based Ant Colony Optimisation Approach for Mixed-Model Parallel Two-Sided Assembly Line Balancing Problem. *Pre-Prints of the Eighteenth International Working Seminar on Production Economics,* 313–328.

Kucukkoc, I., & Zhang, D. Z. (2014b). Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Economics*, *158*, 314–333. https://doi.org/10.1016/j.ijpe.2014.08.010

Kucukkoc, I., & Zhang, D. Z. (2014c). Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Economics*, *52*(12), 3665–3687. https://doi.org/10.1016/j.ijpe.2014.08.010

Kucukkoc, I., & Zhang, D. Z. (2015a). A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem. *Production Planning & Control*, *26*(11), 874–894. https://doi.org/10.1080/09537287.2014.994685

Kucukkoc, I., & Zhang, D. Z. (2015b). Balancing of parallel U-shaped assembly lines. *Computers and Operations Research*, *64*, 233–244. https://doi.org/10.1016/j.cor.2015.05.014

Kucukkoc, I., & Zhang, D. Z. (2015c). Coping with model variations on parallel u-shaped assembly line configurations. *IFAC-PapersOnLine*, *28*(3), 2030–2035.

https://doi.org/10.1016/j.ifacol.2015.06.387

Kucukkoc, I., & Zhang, D. Z. (2015d). Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. *Computers and Industrial Engineering*, *84*, 56–69. https://doi.org/10.1016/j.cie.2014.12.037

Kucukkoc, I., & Zhang, D. Z. (2016a). Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines. *International Journal of Advanced Manufacturing Technology*, *82*(1–4), 265–285. https://doi.org/10.1007/s00170-015-7320-y

Kucukkoc, I., & Zhang, D. Z. (2016b). Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach. *Computers and Industrial Engineering*, *97*, 58–72. https://doi.org/10.1016/j.cie.2016.04.001

Kucukkoc, I., & Zhang, D. Z. (2017). Balancing of mixed-model parallel U-shaped assembly lines considering model sequences. *International Journal of Production Research*, *55*(20), 5958–5975. https://doi.org/10.1080/00207543.2017.1312586

Kucukkoc, I., Zhang, D. Z., & Keedwell, E. C. (2013). Balancing parallel two-sided assembly lines with ant colony optimisation algorithm. *Proceedings of the 2nd Symposium on Nature-Inspired Computing and Applications, NICA 2013 - AISB Convention 2013*, 21–28.

Lusa, A. (2008). A survey of the literature on the multiple or parallel assembly line balancing problem. *European J. of Industrial Engineering*, *2*(1), 50–72. https://doi.org/10.1504/EJIE.2008.016329

Martino, L., & Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, *48*(6), 1787–1804. https://doi.org/10.1080/00207540802577979

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Miltenburg, J. (1998). Balancing U-lines in a multiple U-line facility. *European Journal of Operational Research*, *109*, 1–23.

Miltenburg, J., & Wijngaard, J. (1994). The U-line Line Balancing Problem. *Management Science*, *40*(10), 1378–1388. https://doi.org/10.1287/mnsc.40.10.1378

Nelder, J. A., & Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, *7*(4), 308–313. https://doi.org/10.1093/comjnl/7.4.308

Otto, A., & Otto, C. (2014). How to design effective priority rules: Example of simple assembly line balancing. *Computers and Industrial Engineering*, *69*(1), 43–52. https://doi.org/10.1016/j.cie.2013.12.013

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research*, *228*(1), 33–45. https://doi.org/10.1016/j.ejor.2012.12.029

Özbakir, L., Baykasoğlu, A., Görkemli, B., & Görkemli, L. (2011). Multiple-colony ant algorithm for parallel assembly line balancing problem. *Applied Soft Computing*, *11*(3), 3186–3198. https://doi.org/10.1016/j.asoc.2010.12.021

Özbakır, L., & Seçme, G. (2020). *A hyper-heuristic approach for stochastic parallel assembly line balancing problems with equipment costs*. *Operational Research*. Springer Berlin Heidelberg. https://doi.org/10.1007/s12351-020-00561-x

Özcan, U. (2018). Balancing stochastic parallel assembly lines. *Computers & Operations Research*, *99*, 109–122. https://doi.org/10.1016/j.cor.2018.05.006

Özcan, U. (2019). Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. *International Journal of Production Economics*, *213*, 81–96. https://doi.org/10.1016/j.ijpe.2019.02.023

Özcan, U., Çerçioğlu, H., Gökçen, H., & Toklu, B. (2009). A tabu search algorithm for the Parallel Assembly Line Balancing Problem. *Grazi University Journal of Science*, 313–323. https://doi.org/10.1007/s00170-008-1753-5

Özcan, U., Çerçioğlu, H., Gökçen, H., & Toklu, B. (2010a). Balancing and sequencing of parallel mixed-model assembly lines. *International Journal of Production Research*, *48*(17), 5089–5113. https://doi.org/10.1080/00207540903055735

Özcan, U., Gökçen, H., & Toklu, B. (2010b). Balancing parallel two-sided assembly lines. *International Journal of Production Research*, *48*(16), 4767–4784. https://doi.org/10.1080/00207540903074991

Pastor, R., Aguilar, H., & García-Villoria, A. (2021). Existence and sizing of buffers in parallel assembly lines with multi-line workstations and different cycle times. *DYNA MANAGEMENT*, *9*(1), [10 p.]-[10 p.]. https://doi.org/10.6036/MN10070

Pastor, R., Andrés, C., & Miralles, C. (2010). Corrigendum to “Balancing and scheduling tasks in assembly lines with sequence-dependent setup” [European Journal of Operational Research 187 (2008) 1212-1223] (DOI:10.1016/j.ejor.2006.07.044). *European Journal of Operational Research*, *201*(1), 336. https://doi.org/10.1016/j.ejor.2009.02.019

Pastor, R., Chueca, I., & García-Villoria, A. (2012). A heuristic procedure for solving the Lexicographic Bottleneck Assembly Line Balancing Problem (LB-ALBP). *International Journal of Production Research*, *50*(7), 1862–1876. https://doi.org/10.1080/00207543.2011.578164

Pinto, P., Dannenbring, D. G., & Khumawala, B. M. (1975). A branch and bound algorithm for assembly line balancing with paralleling. *International Journal of Production Research*, *13*(2), 183–196. https://doi.org/10.1080/00207547508942985

Salveson, M. E. (1955). The assembly line balancing problem. *Journal of Industrial Engineering*, *6*(3), 18–25.

Scholl, A. (1993). Data of Assembly line Balaning Problems.

Scholl, A. (1999). *Balancing and sequencing of assembly lines* (2nd ed. He). Heidelberg: Physica-Verlag.

Scholl, A., & Boysen, N. (2009). Designing parallel assembly lines with split workplaces: Model and optimization procedure. *International Journal of Production Economics*, *119*(1), 90–100. https://doi.org/10.1016/j.ijpe.2009.01.011

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, *73*(9–12), 1665–1694. https://doi.org/10.1007/s00170-014-5944-y

Sparling, D. (1998). Balancing Just-In-Time Production Units: The N U-Line Balancing

Problem. *INFOR: Information Systems and Operational Research*, *36*(4), 215–237. https://doi.org/10.1080/03155986.1998.11732360

Talbot, F. B., Patterson, J. H., & Gehrlein, W. V. (1986). Comparative Evaluation of Heuristic Line Balancing Techniques. *Management Science*, *32*(4), 430–454. https://doi.org/10.1287/mnsc.32.4.430

Tapkan, P., Özbakir, L., & Baykasoğlu, A. (2016). Bee algorithms for parallel two-sided assembly line balancing problem with walking times. *Applied Soft Computing Journal*, *39*, 275–291. https://doi.org/10.1016/j.asoc.2015.11.017

Yadav, A., Kulhary, R., Nishad, R., & Agrawal, S. (2020). Parallel two-sided assembly line balancing with tools and tasks sharing. *Assembly Automation*, *40*(6), 833–846. https://doi.org/10.1108/AA-02-2018-025

Zhang, D. Z., & Kucukkoc, I. (2013). Balancing Mixed-Model Parallel Two-Sided Assembly Lines. *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IEEE-IESM'2013), École Mohammadia d'Ingénieurs de Rabat (EMI)*, 391–401.